RoboReason: Boosting Robotics-Relevant VLM Reasoning Through Scalable Data Generation

Sagnik Anupam*, **Utkarsh Lal***, **Aneesh Edara**, **Botong Zhang**, **Yao Tang**, **Jiatao Gu**, **Dinesh Jayaraman**†, **Osbert Bastani**†
University of Pennsylvania

Abstract

Vision-Language Models (VLMs) have demonstrated impressive performance when integrated into robot manipulation pipelines, but real-world evaluation of VLM capabilities on downstream robotics tasks is both difficult and expensive to scale. We propose that spatial and temporal ordering tasks using images from stereo and wrist cameras attached to real-world robots are particularly well-suited for evaluating visual reasoning capabilities, as they assess skills directly applicable to robots. We evaluate VLM reasoning abilities on three questions: ordering shuffled frames conditioned on a language description of the task, ordering objects depthwise based on distance from the camera, and ordering objects from left-to-right in pixel coordinates. The complexity of these tasks can also be modulated by changing the size of the set to sort. Our experiments show that several state-of-the-art vision-language models struggle on ordering tasks. To fix this, we propose a scalable autonomous training data generation pipeline, and show that both supervised finetuning and reinforcement-learning on the generated data can significantly improve VLM reasoning performance both on spatiotemporal ordering tasks as well as downstream robotics and visual reasoning tasks.

1 Introduction

Vision-Language Models (VLMs) have been adapted for diverse roles in robot manipulation pipelines as their capabilities have increased, serving as task planner components (Yang et al., 2025b), reward models (Rocamonde et al., 2024), action annotators (Hancock et al., 2026), and code policy generators (Liang et al., 2023). Effectively performing these tasks requires sophisticated reasoning over large collections of images. Consider using a VLM to predict task progress based on recent observations; the VLM must be able to reason about spatial relationships between objects both within a frame and across frames—e.g., to measure whether the gripper moved closer to a target object to be grasped.

As a consequence, there is an urgent need to study and improve the fundamental capabilities of VLMs at spatiotemporal reasoning. While there has been significant recent interest in this problem (Wang et al., 2025), much of it focuses on general settings that do not capture the specific challenges that arise in robotics settings: (1) cluttered scenes from robot-centric viewpoints, (2) reasoning about object relationships across time, and (3) integrating information across multiple viewpoints.

We first conduct a systematic evaluation of multiple large VLMs on realistic spatiotemporal ordering tasks under a variety of settings, like having access to computer vision model outputs or having structured test-time compute. Our ordering problems are constructed from robot demonstration videos in the DROID dataset (Khazatsky et al., 2024) (examples in Figure 1). Our results indicate that leading VLMs underperform on our dataset of real-world robotics images when compared to previously reported results. To address this performance gap, we present RoboReason, a scalable data generation pipeline for generating spatiotemporal reasoning training data from robot manipulation trajectories. We train much smaller models on a small amount (700 samples) of pipeline-generated data using Supervised Finetuning (SFT) and Group Relative Policy Optimization (GRPO) (Shao et al., 2024) and show that model performance increases significantly, even matching the performance of much larger models on some tasks. We also train small models on larger pipeline-generated training datasets of 8.8K sam-

Correspondence to co-first author: sanupam<at>engineering.upenn.edu. Equal authorship is indicated with * while † indicates equal advising. Code and data available at https://sagnikanupam.com/papers/roboreason.

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 54156–54182
October 24-29, 2026 ©2026 Association for Computational Linguistics

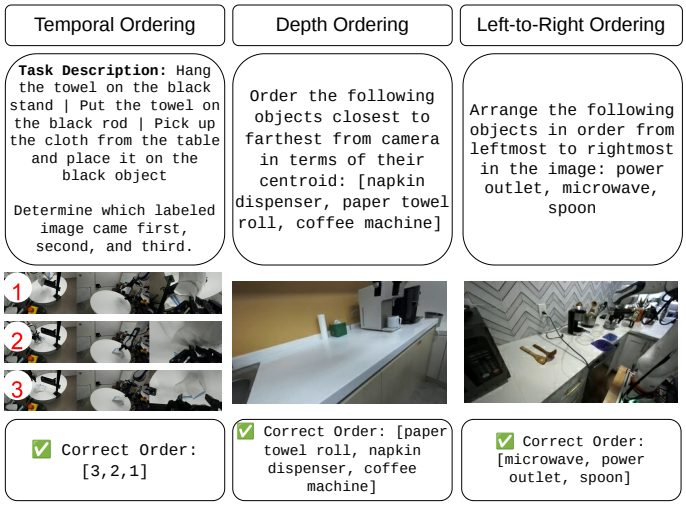


Figure 1: Examples of spatiotemporal ordering tasks in our dataset.

ples and find that model performance also improves significantly on downstream spatial and robotics reasoning tasks, demonstrating the potential for our approach to improve the efficacy of VLMs targeting robotics tasks.

Our main findings are as follows:

- **VLMs that perform well on existing spatial reasoning benchmarks struggle on multi-object ordering tasks in real-world robot manipulation images.** We evaluate state-of-the-art embodied reasoning VLMs like Gemini Robotics-ER 1.5, GPT-5, and Qwen3-VL-235B on our tasks and discover that their performance decreases sharply as we increase the number of items to order. Additionally, GPT-5 and Gemini-ER-1.5 are unable to achieve more than 80% accuracy on the easiest base setting of ordering two objects in any of our spatiotemporal tasks, despite reporting higher results in several other spatial reasoning benchmarks (Ray et al., 2024; Tong et al., 2024; Team et al., 2025).

- **VLMs improve their performance in spatial ordering tasks with outputs from computer vision models.** We show that using a specialized vision model to predict segmentation maps or bounding boxes and then providing this information to the VLM mostly improves its accuracy on our left-to-right and depth-wise spatial ordering tasks. These results suggest that VLMs struggle with basic capabilities such as identifying objects in cluttered environments, compared to specialized vision models.

- **VLMs do not improve performance on temporal ordering tasks when provided with structured test-time compute, but do improve performance on spatial ordering tasks when provided with the same compute.** We study if explicitly structuring test-time compute by applying merge sort over VLM-generated pairwise comparisons aids performance in high reasoning settings. Our results show that merge sort performs worse or at a comparable level to providing all set elements to the VLM at once on temporal ordering tasks, but does improve spatial ordering performance. These results suggest that VLMs successfully reason over more than two frames to solve our temporal ordering tasks.

- **Both reinforcement learning and supervised finetuning (SFT) on pipeline-generated data improve small model performance on downstream reasoning tasks.** We scale our data construction pipeline to construct 8857 training samples for each ordering task, and finetune Qwen3VL-8B-Instruct on generated data. We observe that training on generated data improves performance on multiple downstream tasks. However, no single approach improves performance across all downstream tasks, and, in some cases, performance degrades after training on generated data.

- **Reinforcement learning yields limited benefits on temporal ordering but offers large boosts on spatial ordering on training with a small set of generated data.** We perform reinforcement-learning finetuning experiments on Qwen2-VL-

2B-Instruct and SmolVLM2-2.2B-Instruct using GRPO to maximize ordering performance on our tasks can improve its performance. Performance improves significantly on both 2D and depth spatial ordering tasks (with Qwen2-VL outperforming all larger VLMs on some tasks), while we observe more limited performance improvements on our temporal ordering task. Small model performance also increases proportional to the size of the RoboReason-generated training dataset.

Related work. While several large-scale robot manipulation datasets have been released with associated robot camera images (O'Neill et al., 2024; Khazatsky et al., 2024), attempts to utilize such trajectories for evaluating VLM reasoning have focused on trajectory-specific spatial orientation and robot-object interactions (Chen et al., 2025). Alternatively, most approaches to evaluation VLM spatiotemporal reasoning use question-answer benchmarks with high-resolution images and answer options (Hudson and Manning, 2019; Antol et al., 2015; Wang et al., 2025), which means VLM reasoning results can be influenced by guesswork, selection bias, and lack of visual noise or cluttered scenes (Balepur et al., 2024; Zheng et al., 2024; Song et al., 2025). Our approach instead leverages tasks using real-world robot camera images and trains models on multi-object spatiotemporal ordering tasks, decreasing the evaluation bias. We discuss related work in detail in Appendix A.

2 RoboReason

Our pipeline comprises of methods to construct three ordering tasks. In each task, the VLM is provided with a shuffled set of items alongside any task-specific context, and is tasked with reproducing the original ordering of the items. We use the pipeline task outputs as-is for our training experiments, but our test set has been checked by humans. We formalize the three ordering tasks as follows:

1. **Temporal ordering:** We assume we are provided with a set of video frames $f_1, f_2, \ldots, f_n$ with associated timestamps $t_1, t_2, \ldots, t_n$ and a language description l describing the task being performed. After providing only the frames and the language description to the VLM, we ask the VLM to output a permutation of the frames $VLM(f_1, \ldots, f_n; l) = i_1, \ldots, i_n$ such that $t_{i_1} < t_{i_2} < \cdots < t_{i_n}$. In the canonical task setting, each video frame comprises three images (left, right, and wrist camera) stitched into a single image.

2. **Depth ordering:** This task provides a single video frame f alongside a list of the names of objects present in f specified via their names. We denote the list of objects as $o_1, o_2, \ldots, o_n$, and their associated distances from the camera $d_1, \ldots, d_n$. The VLM is then prompted to output a permutation of the objects $VLM(o_1, \ldots, o_n; f) = i_1, \ldots, i_n$ such that $d_{i_1} < d_{i_2} < \cdots < d_{i_n}$.

3. **Left-to-right ordering:** This task is constructed in a similar way as depth ordering, except $d_1, \ldots, d_n$ now measure the distance between the left edge of frame f and the centroid of the object in 2D image space.

2.1 Data Generation Pipeline

Temporal Ordering. We randomly sample trajectories from the DROID dataset (Khazatsky et al., 2024) that have language annotations. We divide each selected trajectory into 5 equal segments, and select one frame at random from each segment. We label the five images 1-5 after sorting in increasing temporal order, and for the corresponding 3 image setting, we select images 1, 3 and 5. Ground truth orderings are calculated using the timestamps of the trajectory provided in the original DROID trajectory. Recall that these are expert demonstration trajectories, so frames sampled from later in a trajectory will typically exhibit higher task progression than those from earlier in the trajectory. It is the ability to assess this robotics task progression that our ordering task is designed to evaluate.

Depth Ordering: Using the trajectories sampled for the temporal ordering task, we use the left camera view of a randomly selected video frame from each trajectory for the depth ordering task, since the wrist camera view is often directly above an object and does not show multiple objects at different depths. We identify objects present in the image with GPT-5, and construct bounding boxes using GroundingDINO (Liu et al., 2025) for all identified objects. Then, we filter out all trajectories where fewer than 5 objects with non-overlapping intersection over union mask, bounding boxes, and containment are visible in the scene. We impose a mask intersection over union (IoU) threshold of 0.01, a bounding box IoU threshold of 0.3, and a bounding box containment threshold of 0.5. If more than 5 such objects are present in a trajectory,

we uniformly sample objects to initialize our set of 5 objects. For ground truth depth ordering, we generate depth maps for each image using Marigold (Ke et al., 2025), a monocular depth estimation model. We then generate segmentation masks of each object using Grounded SAM 2 (Ren et al., 2024) (using the GPT-5 generated object names), and compute the mean depth of each object over all pixels in the segmentation mask.

Left-to-Right Ordering: For our left-to-right ordering task, we use the left-view camera images from the depth task, each from a unique DROID trajectory. For each image, similar to depth ordering above, we use the list of objects identified by GPT-5, and infer object bounding boxes for each with GroundingDINO. We first remove any objects we classify as scene objects (objects with $\texttt{width}_{\texttt{bbox}} > 0.7 \times \texttt{width}_{\texttt{image}}$). Then, we find the largest cluster of objects lying along a similar horizontal level, enforcing a maximum centroid y-coordinate difference of 0.5. Finally, in this cluster, we remove any objects that almost fully overlap with another object, i.e., have bounding boxes with IOU > 0.7. If we have less than 5 valid objects at this point, we discard the trajectory. The ground truth left-to-right ordering is constructed for valid trajectories using the x-coordinate of the centroid of the bounding box encompassing each object.

For the left-to-right and depth datasets, once we have filtered trajectories from the original temporal dataset, we continue sampling trajectories until we achieve the desired training set size.

2.2 Reinforcement Learning Finetuning

To examine if reinforcement learning can be used as a viable method for improving model performance on our spatiotemporal ordering tasks, we train Qwen2-VL-2B-Instruct and SmolVLM2-2.2B-Instruct using GRPO (Shao et al., 2024) on our three ordering tasks using data generated via our pipeline. We use the Kendall rank correlation coefficient, defined as $\tau = 1 - \frac{2(\text{number of discordant pairs})}{\binom{n}{2}}$, between the LLM generated rank and the ground truth rank as a reward function for all our tasks.

3 Experiments

We conduct two sets of experiments. First, we evaluate the spatiotemporal reasoning ability of current state-of-the-art VLMs (Section 3.2). Second, we evaluate our training data generation pipeline by human validation of the pipeline outputs and evaluating small VLMs finetuned on the pipeline-generated data (Section 3.3).

Our experiments seek to answer the following questions with respect to large VLM performance:

1. How does VLM spatiotemporal ability vary across different reasoning levels?
2. Does structuring test-time compute by performing pairwise comparisons and using merge sort help improve model performance?
3. At what rate does model performance decline as more objects/images are added to the ordering list, and how does this rate vary for different models?
4. Does overlaying segmentation maps and bounding boxes on images help improve VLM spatiotemporal reasoning performance?

We attempt to answer the following questions to evaluate RoboReason-generated training data:

1. How accurate are the pipeline generated outputs with respect to human annotations?
2. Does finetuning on our ordering tasks improve downstream robotics or visual reasoning task performance?
3. How does finetuned model performance on spatiotemporal ordering change with i) training task type and ii) amount of pipeline-generated data used?

3.1 Experimental Setup

Since we seek to evaluate models under different reasoning levels, we set output token limits for low, medium, and high reasoning settings in our tasks (described in Table 11), based on the maximum output tokens generated by GPT-5 in a given OpenAI reasoning setting (low, medium, high) for a particular ordering task. This helps ensure a consistent comparison with other models, which require specifying reasoning or output budgets in tokens. For the merge sort format of the tasks (which we only evaluate in the high reasoning setting), the token counts are determined separately using the maximum token count produced by GPT-5 while performing pairwise comparisons.

For our three ordering tasks, we present results from 3 settings: the base image setting (which only provides the VLM with the original DROID

Task Format	Model Name	Base Image			Segmentation Map Image			Bounding Box Image		
		Low	Medium	High	Low	Medium	High	Low	Medium	High
5 Image Set	GPT-5	15.67	**19.67**	18.67	11.00	**12.00**	11.33	8.33	8.67	**10.00**
	GPT-5-Nano	0.67	2.33	1.00	2.00	2.33	0.67	1.00	3.33	2.33
	Gemini-ER-1.5	1.33	6.67	7.33	3.67	4.00	5.33	2.33	4.33	5.33
	Qwen3-VL-235B	6.67	9.67	7.67	3.00	4.00	4.33	2.67	2.67	3.33
3 Image Set	GPT-5	**49.33**	47.67	47.33	44.67	**45.00**	41.33	39.67	38.67	**40.67**
	GPT-5-Nano	17.00	23.33	23.67	20.33	20.00	24.67	18.33	22.67	22.00
	Gemini-ER-1.5	38.33	36.67	38.00	32.67	37.33	35.67	32.00	35.00	34.67
	Qwen3-VL-235B	32.33	32.00	32.67	30.33	28.00	28.67	29.67	28.00	27.00

Table 1: Model Accuracies (%) for Temporal Ordering Tasks.

Task Format	Model Name	Base Image			Segmentation Map Image			Bounding Box Image		
		Low	Medium	High	Low	Medium	High	Low	Medium	High
5 Object Set	GPT-5	4.33	4.33	5.00	**9.33**	8.00	6.67	7.67	**10.33**	7.33
	GPT-5-Nano	2.33	4.33	3.00	5.67	3.33	2.67	3.00	3.33	3.67
	Gemini-ER-1.5	3.00	**6.00**	5.67	4.33	8.00	7.33	6.33	9.33	9.33
	Qwen3-VL-235B	2.67	3.00	3.33	5.00	3.00	4.00	5.33	5.00	4.67
3 Object Set	GPT-5	34.00	34.00	32.67	**42.33**	**42.33**	41.33	43.67	41.67	39.00
	GPT-5-Nano	26.00	25.67	25.67	30.67	29.00	26.67	27.00	29.67	26.67
	Gemini-ER-1.5	25.67	**35.67**	**35.67**	31.00	37.33	38.33	38.33	40.00	**44.00**
	Qwen3-VL-235B	29.67	30.00	29.67	31.67	30.67	30.67	34.67	34.00	36.00

Table 2: Model Accuracies (%) for Depth Ordering Tasks

image), the segmentation map setting (which overlays the GroundedSAM 2 segmentation map associated with each object on top of the image), and the bounding box setting (which overlays the GroundingDINO bounding box associated with each object on top of the image). For depth and left-to-right ordering tasks, we only overlay bounding boxes and segmentation maps associated with the objects to be ordered. For the temporal ordering setting, we overlay bounding boxes and segmentation maps on all 3 camera views for objects that GPT-5 identifies as relevant objects from the task description. We present our results for temporal ordering in Table 1, depth ordering in Table 2, and left-to-right ordering in Table 3. In the base image setting, we examine how the number of objects to order affects performance in Table 4, where we randomly sample images/objects from our initial 5-length image/object set to construct the 2 and 4 length versions for all task types. In all our ordering experiments, we calculate accuracy as the percentage of samples where the model outputs exactly the same order as the ground truth values. For our small generated dataset experiments, we construct a dataset of 1000 samples for each task type. A subset of 300 randomly sampled tasks are then selected as a test set, with the remaining being used as our small training dataset. For the depth and left-to-right test sets, since the ground truths are model generated, we validate the pipeline's generated object labels and ordering by having a human correct any ambiguous labels or incorrect orderings. All our ordering task results are from the human-validated, 300 sample test set. For our downstream task validation experiments, we scale the training dataset to over 8.8K samples for each task type.

We evaluate our finetuned checkpoints on visual reasoning tasks using the SAT (Ray et al., 2024) benchmark, as well as two downstream robotics tasks: autoregressive value estimation using the RT-1 (Brohan et al., 2022) and Dobb-E (Shafiullah et al., 2023) datasets (proposed in (Ma et al., 2025)), and robotics failure analysis question-answering using the real-world trajectory question-pairs from the RoboFAC dataset (Ye et al., 2025).

3.2 Zero-Shot VLM Performance

Model Name	Across All 3 and 5 Set Cardinality Ordering Tasks		
	Base	Segmentation Map	Bounding Box
GPT-5	1.83 ± 1.49	2.06 ± 1.02	$\mathbf{3.17} \pm 1.28$
GPT-5-Nano	3.33 ± 2.39	$\mathbf{3.94} \pm 1.39$	3.49 ± 2.37
Gemini-ER-1.5	3.96 ± 3.43	$\mathbf{3.99} \pm 2.37$	3.53 ± 1.06
Qwen3-VL-235B	1.61 ± 1.29	$\mathbf{2.92} \pm 3.11$	2.52 ± 2.14

Table 5: Mean and Standard Deviation of Reasoning Difference

Task Format	Model Name	Temporal Ordering	Depth Ordering	Left-to-Right Ordering
5 Image/Object Set	GPT-5	**8.00**	5.67	**23.00**
	GPT-5-Nano	1.00	5.00	12.67
	Gemini-ER-1.5	1.67	**6.33**	22.33
	Qwen3-VL-235B	0.33	3.33	22.00
3 Image/Object Set	GPT-5	**42.33**	**33.00**	60.00
	GPT-5-Nano	23.00	27.00	47.33
	Gemini-ER-1.5	16.33	31.33	59.00
	Qwen3-VL-235B	20.33	23.00	**61.00**

Table 6: Model Accuracies (%) for High-Reasoning Merge Sort Based Ordering in Base Image Setting

Task Format	Model Name	Base Image			Segmentation Map Image			Bounding Box Image		
		Low	Medium	High	Low	Medium	High	Low	Medium	High
5 Object Set	GPT-5	21.00	21.00	21.00	17.67	19.33	19.67	20.33	18.00	22.67
	GPT-5-Nano	7.00	8.67	11.33	5.33	7.67	10.33	4.41	5.67	7.33
	Gemini-ER-1.5	18.79	18.06	19.38	17.58	15.71	18.37	28.77	**30.64**	27.21
	Qwen3-VL-235B	**22.33**	22.00	19.00	14.33	17.67	**23.53**	20.67	18.33	21.00
3 Object Set	GPT-5	58.00	57.00	**60.00**	56.33	55.00	**57.00**	49.67	52.00	52.67
	GPT-5-Nano	39.67	43.00	44.67	36.67	40.67	42.00	33.33	36.00	41.00
	Gemini-ER-1.5	56.00	56.23	59.12	50.84	55.82	56.8	61.67	**64.77**	62.96
	Qwen3-VL-235B	57.00	58.33	58.67	50.00	50.00	48.33	56.00	55.33	61.8

Table 3: Model Accuracies (%) for Left-to-Right Ordering Tasks

Task Format	Model Name	Temporal Ordering Task			Depth Ordering Task			Left-to-Right Ordering Task		
		Low	Medium	High	Low	Medium	High	Low	Medium	High
5 Image/Object Set	GPT-5	15.67	**19.67**	18.67	4.33	4.33	5.00	21.00	21.00	21.00
	GPT-5-Nano	0.67	2.33	1.00	2.33	4.33	3.00	7.00	8.67	11.33
	Gemini-ER-1.5	1.33	6.67	7.33	3.00	**6.00**	5.67	18.79	18.06	19.38
	Qwen3-VL-235B	6.67	9.67	7.67	2.67	3.00	3.33	**22.33**	22.00	19.00
4 Image/Object Set	GPT-5	27.00	29.33	**30.00**	9.67	13.00	11.00	32.67	32.00	37.00
	GPT-5-Nano	6.67	8.00	5.33	6.67	9.33	8.33	17.67	23.67	23.67
	Gemini-ER-1.5	16.33	23.67	18.67	11.00	14.67	11.00	**38.41**	34.93	37.25
	Qwen3-VL-235B	18.33	16.00	13.67	17.67	**18.33**	17.00	37.67	38.33	34.33
3 Image/Object Set	GPT-5	**49.33**	47.67	47.33	34.00	34.00	32.67	58.00	57.00	**60.00**
	GPT-5-Nano	17.00	23.33	23.67	26.00	25.67	25.67	39.67	43.00	44.67
	Gemini-ER-1.5	38.33	36.67	38.00	25.67	**35.67**	**35.67**	56.00	56.23	59.12
	Qwen3-VL-235B	32.33	32.00	32.67	29.67	30.00	29.67	57.00	58.33	58.67
2 Image/Object Set	GPT-5	64.00	**67.67**	67.33	52.33	53.67	53.33	78.33	**80.00**	77.67
	GPT-5-Nano	53.33	52.00	52.00	52.00	49.67	49.33	63.33	67.00	69.00
	Gemini-ER-1.5	50.67	58.33	60.33	49.33	54.67	54.33	74.67	78.26	75.59
	Qwen3-VL-235B	60.67	62.33	59.67	70.00	68.00	**71.33**	78.67	78.00	78.67

Table 4: Model Accuracies (%) Across Different Set Cardinalities. All results are in the base setting (i.e. no images had any bounding boxes or segmentation map overlays on them).

Impact of Reasoning Levels: To measure the impact of reasoning levels, we measure the difference in task performance in a particular setting (for example, 3 image base setting temporal ordering) between the best and worst performing reasoning level. We refer to this quantity as the reasoning difference, and report its mean and standard deviations across different tasks in Table 5.

While in most settings, higher reasoning levels are better, in the base setting, we sometimes find that lower reasoning levels outperform higher ones, especially in 2 object/image temporal and depth settings. We hypothesize that this is because the low reasoning level does not allow for enough tokens to reason about ordering a large number of images or objects, while for pairwise comparisons, generating excessive tokens can lead the model to "overthink" and draw inaccurate conclusions.

Impact of Structured Test-Time Compute: Given the relatively high performance of VLMs while ordering pairs of objects, we evaluate the performance of VLMs by scaling their test time compute using an external structure. We implement a merge sort algorithm that calls the VLM to make pairwise comparisons, and our results (included in Table 6) show that in the high reasoning setting on the spatial ordering tasks, VLMs usually perform at least as good as and often better in the structured test-time compute approach than when provided with all objects to order at once. On the other hand, they perform much better on temporal tasks when they are provided with all the frames at once, suggesting that VLMs improve their performance by reasoning over multiple images correctly in the temporal ordering task, and some of this information is lost with pairwise comparisons.

Impact of Set Cardinalities: Model performance on spatiotemporal ordering tasks declines sharply as more elements are added to the set to be ordered. However, different VLMs decline at different rates, and in on certain task settings, two models (for example, GPT-5 and GPT-5-Nano with low reasoning in the depth ordering task) that perform at relatively similar levels when asked to order 2 elements have a much larger gap in performance when asked to order 3 or more elements. It also may be the case that models that perform at a similar or worse level as other models at smaller set cardinalities perform relatively better at larger set cardinalities, as evidenced by GPT-5 in the aforementioned setting when the models are asked to order 3 or more objects. We show some examples

of these occurrences in Figure 2.

Impact of Segmentation Map and Bounding Box Overlays: For our two canonical settings (ordering 5 and 3 images/objects), we study the difference in performance caused by passing images overlaid with segmentation maps and bounding boxes. Bounding boxes achieve comparable (within 2% of original accuracy) or better performance for all models at each reasoning level in the depth ordering task, and improve performance for Gemini-ER-1.5 and Qwen3-VL-235B in some reasoning levels in the left-to-right ordering tasks. Similarly, segmentation maps also achieve comparable or better performance for all models at each reasoning level in the depth ordering task, although they do not perform as well as bounding box overlays in the left-to-right ordering task.

Mean reasoning difference increases for all models on using images with overlaid segmentation maps, and increases for all models except Gemini-ER-1.5 on using overlaid bounding boxes. Additionally, using bounding box and segmentation map overlays also decreases the standard deviation slightly for all models except Qwen3-VL-235B. Our results suggest that using additional tokens when images are overlaid with computer vision model outputs may improve performance more than expending more tokens in the base setting.

Our results demonstrate that large VLMs have limited performance on spatiotemporal ordering tasks, which often falls short of even prior reported results when applied to our robotics-relevant data.

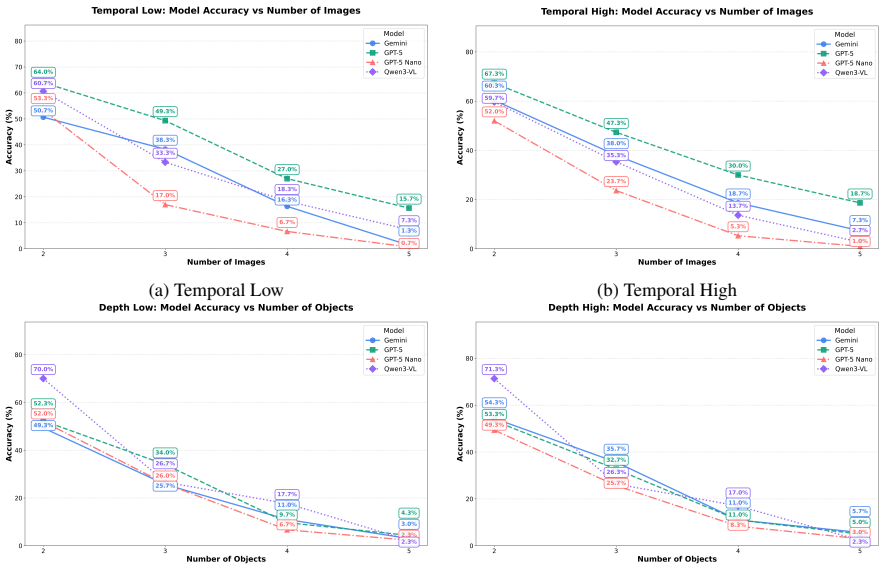
3.3 RoboReason Evaluations

To address these limitations, we generate training data using the scalable RoboReason pipeline. We use it to finetune VLMs, focusing on smaller models due to compute limitations. For evaluating our training data pipeline, we first compare the pipeline outputs to those of a human evaluator. After finetuning small VLMs on the pipeline-generated data, we compare the post-trained models' performance to the original model on both our spatiotemporal ordering tasks and downstream robotics and visual reasoning tasks. Our results show that while the pipeline ground truth orders differ from human evaluator's assessments in a significant percentage of cases, our data generation pipeline still produces training data that provides a strong signal for learning and helps improve small model performance on both ordering and downstream tasks.

Human Evaluation: Since our depth and left-to-right ordering task ground truths are generated by computer vision models, we have a human evaluator validate the 5 object depth and left-to-right test sets prior to evaluating VLMs. The evaluator corrects the ordering if they identify all the objects in the image and determine the pipeline-generated ordering to be incorrect. If any labels are ambiguous, and the evaluator cannot identify an object in the image, they modify the object label prior to updating the ordering. Since the test set trajectories are a random subset of DROID trajectories (and thus drawn from the same distribution as the training set), we present our human evaluator disagreement results as an evaluation of our training data generation pipeline. For the left-to-right tasks, 15.67% of samples in the test set were determined to have at least one object with an incorrect position in the ground truth order, and 29% of samples were determined to have at least one ambiguous label. For the depth tasks, 26.67% of samples in the test set were determined to have at least one object with an incorrect position in the ground truth order, while 22.67% of samples were determined to have at least one ambiguous label.

Impact of Training on Pipeline-Generated Data on Downstream Reasoning Tasks: To investigate if training on RoboReason-generated data can improve performance on downstream reasoning tasks, we finetune a Qwen3VL-8B-Instruct model on 8.8K samples each for our three task types, using both SFT and GRPO. We evaluate each finetuned model on both our ordering tasks, as well as additional visual and robotics reasoning tasks, reporting our results in Table 7 and experimental setup details in Appendix B.

Impact of Task Type and Amount of Data on Finetuning Performance: We show that finetuning on even a small amount of pipeline-generated data significantly improves performance on base image setting ordering tasks in Table 8, enabling the small models to reach the same level of performance as much larger models on the spatial ordering tasks while showing smaller gains on temporal ordering tasks (e.g. Qwen2-VL-2B surpasses the performance of all larger VLMs on the 3-object depth and left-to-right ordering tasks). In general, we find that reinforcement learning improves performance to a greater extent than supervised finetuning on the same task. We also study the impact of the training data amount on the model performance, by training multiple models in each of our 3 Image/Object base image settings with different



(a) Temporal Low (b) Temporal High

(c) Depth Low (d) Depth High

Figure 2: Comparison of how model performance scales with set cardinality in temporal and depth ordering tasks in the base setting. In the temporal high reasoning setting, model performance scales predictably with increase in set cardinality, maintaining a consistent relative order across all set cardinalities. However, the relative order is inconsistent in the remaining settings in the figure.

Model	3 Image/Object Spatiotemporal Ordering			Downstream Reasoning				
	Temporal	Depth	Left-to-Right	SAT	RoboFAC (MCQ)	RoboFAC (Open-Ended)	Value Est. (RT-1)	Value Est. (Dobb-E)
Base	18.00	22.33	53.67	**70.67**	62.55	12.28	0.0066	**0.0123**
SFT (Temporal)	**50.67**	26.00	55.33	**70.67**	60.21	12.92	0.0001	0.0114
SFT (Depth)	22.67	**31.67**	56.67	68.67	60.26	**13.33**	0.0047	0.0015
SFT (Left-to-Right)	23.67	24.67	**61.33**	69.33	58.91	13.16	0.0080	0.0062
GRPO (Temporal)	22.67	26.00	51.67	68.67	**63.13**	11.94	0.0060	0.0036
GRPO (Depth)	22.67	31.33	51.00	68.67	59.84	8.28	**0.0129**	0.0071
GRPO (Left-to-Right)	20.67	17.33	55.33	**70.67**	56.25	11.76	0.0001	0.0000

Table 7: Finetuned Qwen3VL-8B Accuracies on Spatiotemporal Ordering and Downstream Reasoning Tasks

amounts of training data (Table 9). Our results show that model performance scales proportionally to training dataset size, suggesting that scaling training data using our pipeline can help models perform better at spatiotemporal ordering.

Model Name	Training Method	Temporal	Depth	Left-to-Right
Qwen2-VL-2B	Base	17.33	15.66	16.00
	GRPO-300	19.00	37.00	54.00
	GRPO-500	20.00	56.00	63.66
	GRPO-700	**21.67**	**66.00**	**67.66**
SmolVLM2-2.2B	Base	13.00	8.00	12.33
	GRPO-300	15.33	24.33	45.00
	GRPO-500	15.66	30.66	53.33
	GRPO-700	**17.00**	**34.00**	**58.67**

Table 9: 3 Image/Object ordering model accuracies (%) after GRPO under different training data sizes

Model Name	Task Format	Training Method	Temporal	Depth	Left-to-Right
Qwen2-VL-2B	5 Images/Objects	Base	1.00	0.66	0.67
		GRPO	**2.00**	**16.33**	**17.33**
		SFT	0.33	10.33	15.33
	3 Images/Objects	Base	17.33	15.66	16.00
		GRPO	**21.67**	**66.00**	**67.66**
		SFT	16.00	36.00	47.66
SmolVLM2-2.2B	5 Images/Objects	Base	0.33	0.00	2.00
		GRPO	1.33	**2.00**	**16.66**
		SFT	**1.66**	1.00	9.33
	3 Images/Objects	Base	13.00	5.33	12.33
		GRPO	**17.00**	**34.00**	**58.67**
		SFT	12.67	19.33	44.00

Table 8: Model Accuracies (%) for GRPO and SFT

4 Conclusion

We have introduced RoboReason, a scalable task generation pipeline for spatiotemporal reasoning that utilizes real-world robot manipulation images. Experimentally, we first establish the limits of today's VLMs: we demonstrate the impact of reasoning levels, ordering set cardinalities, structured test-time compue and computer vision tools on VLM spatiotemporal reasoning. Next, we demonstrate that by finetuning smaller VLMs on our pipeline-generated data, we can surpass the performance

of much larger VLMs on spatial ordering tasks and improve model performance on downstream reasoning tasks. These results suggest that utilizing reinforcement learning approaches and computer vision model outputs are encouraging directions for future work on real-world spatiotemporal reasoning. We believe that RoboReason presents a promising way forward for improving robotics-relevant VLM capabilities.

Limitations. For the depth and left-to-right ordering tasks, our pipeline can sometimes produce erroneous ground truths due to failures in individual modules such as object detection, bounding box generation, and depth estimation. Additionally, our filtering methods sometimes introduce ambiguity by selecting objects that may overlap or be equidistant from the camera. While using GPT-5 generated object labels may theoretically favor GPT-5 during evaluation due to egocentric bias, we do not see empirical evidence of this occurring, as in several instances, other models outperform it on spatial ordering tasks, and its own performance is sometimes hurt by providing it with images with overlays associated with the objects it had identified. For temporal ordering, it can sometimes be difficult to reason about task progression given certain task descriptions, since the VLM must estimate the direction of motion based only on static images of the scene. Such tasks constitute a minority of tasks in our temporal ordering dataset, and are often solvable using other environment context clues, but our pipeline does not explicitly filter for such ambiguous tasks descriptions. Finally, for temporal ordering tasks, we observe that VLMs struggle to reason across views (as shown in Appendix C) and typically improve performance when restricted to a single camera view. However, since we wish to improve multi-view reasoning during training, we generate multi-view tasks in our temporal setting.

Acknowledgements

This work is funded in part by the UPenn DARPA MOCHA CR PO-0074571/10101353 subcontract of Peraton Labs, DARPA TIAMAT HR0011249042, NSF CAREER 2239301, ONR N00014-22-1-2677, and NSF SLES 2331783. Yao Tang was supported in part by a gift from AWS to Penn Engineering's ASSET Center for Trustworthy AI. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA, NSF, Peraton Labs, or AWS.

References

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433.

Nishant Balepur, Abhilasha Ravichander, and Rachel Rudinger. 2024. Artifacts or abduction: How do LLMs answer multiple-choice questions without the question? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10308–10330, Bangkok, Thailand. Association for Computational Linguistics.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, and 32 others. 2022. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*.

Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. 2017. Matterport3d: Learning from rgb-d data in indoor environments. *arXiv preprint arXiv:1709.06158*.

Kaiyuan Chen, Shuangyu Xie, Zehan Ma, Pannag R Sanketi, and Ken Goldberg. 2025. Robo2vlm: Visual question answering from large-scale in-the-wild robot manipulation datasets. *arXiv preprint arXiv:2505.15517*.

Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. 2017. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2432–2443.

Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Kiana Ehsani, Jordi Salvador, Winson Han, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. 2022. ProcTHOR: Large-scale embodied ai using procedural generation. In *Advances in Neural Information Processing Systems*, volume 35, pages 5982–5994. Curran Associates, Inc.

Mengfei Du, Binhao Wu, Zejun Li, Xuanjing Huang, and Zhongyu Wei. 2024. EmbSpatial-bench: Benchmarking spatial understanding for embodied tasks with large vision-language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 346–355, Bangkok, Thailand. Association for Computational Linguistics.

Hao-Shu Fang, Chenxi Wang, Minghao Gou, and Cewu Lu. 2020. Graspnet-1billion: A large-scale benchmark for general object grasping. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11441–11450.

Xingyu Fu, Yushi Hu, Bangzheng Li, Yu Feng, Haoyu Wang, Xudong Lin, Dan Roth, Noah A. Smith, Wei-Chiu Ma, and Ranjay Krishna. 2025. Blink: Multimodal large language models can see but not perceive. In *Computer Vision – ECCV 2024*, pages 148–166, Cham. Springer Nature Switzerland.

Asher Hancock, Xindi Wu, Lihan Zha, Olga Russakovsky, and Anirudha Majumdar. 2026. Actions as language: Fine-tuning vlms into vlas without catastrophic forgetting. In *International Conference on Learning Representations*, volume 2026, pages 75333–75360.

Drew A. Hudson and Christopher D. Manning. 2019. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6693–6702.

Bingxin Ke, Kevin Qu, Tianfu Wang, Nando Metzger, Shengyu Huang, Bo Li, Anton Obukhov, and Konrad Schindler. 2025. Marigold: Affordable adaptation of diffusion-based image generators for image analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–18.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, Peter David Fagan, Joey Hejna, Masha Itkina, Marion Lepert, Yecheng Jason Ma, Patrick Tree Miller, Jimmy Wu, Suneel Belkhale, Shivin Dass, and 82 others. 2024. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*.

Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, Aniruddha Kembhavi, Abhinav Gupta, and Ali Farhadi. 2017. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*.

Bo Li, Yuanhan Zhang, Liangyu Chen, Jinghao Wang, Fanyi Pu, Joshua Adrian Cahyono, Jingkang Yang, Chunyuan Li, and Ziwei Liu. 2025. Otter: A multi-modal model with in-context instruction tuning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(9):7543–7557.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. 2023. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500.

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. 2025. Grounding dino: Marrying dino with grounded pretraining for open-set object detection. In *Computer Vision – ECCV 2024*, pages 38–55, Cham. Springer Nature Switzerland.

Yecheng Jason Ma, Joey Hejna, Chuyuan Fu, Dhruv Shah, Jacky Liang, Zhuo Xu, Sean Kirmani, Peng Xu, Danny Driess, Ted Xiao, Osbert Bastani, Dinesh Jayaraman, Wenhao Yu, Tingnan Zhang, Dorsa Sadigh, and Fei Xia. 2025. Vision language models are in-context value learners. In *International Conference on Learning Representations*, volume 2025, pages 33984–34009.

Arjun Majumdar, Anurag Ajay, Xiaohan Zhang, Pranav Putta, Sriram Yenamandra, Mikael Henaff, Sneha Silwal, Paul Mcvay, Oleksandr Maksymets, Sergio Arnaud, Karmesh Yadav, Qiyang Li, Ben Newman, Mohit Sharma, Vincent Berges, Shiqi Zhang, Pulkit Agrawal, Yonatan Bisk, Dhruv Batra, and 5 others. 2024. Openeqa: Embodied question answering in the era of foundation models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16488–16498.

NVIDIA, Alisson Azzolini, Junjie Bai, Hannah Brandon, Jiaxin Cao, Prithvijit Chattopadhyay, Huayu Chen, Jinju Chu, Yin Cui, Jenna Diamond, Yifan Ding, Liang Feng, Francesco Ferroni, Rama Govindaraju, Jinwei Gu, Siddharth Gururani, Imad El Hanafi, Zekun Hao, Jacob Huffman, and 34 others. 2025. Cosmos-reason1: From physical common sense to embodied reasoning.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew Wang, Anikait Singh, and 260 others. 2024. Open x-embodiment: Robotic learning datasets and rt-x models : Open x-embodiment collaboration0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903.

Arijit Ray, Jiafei Duan, Ellis Brown, Reuben Tan, Dina Bashkirova, Rose Hendrix, Kiana Ehsani, Aniruddha Kembhavi, Bryan A. Plummer, Ranjay Krishna, Kuo-Hao Zeng, and Kate Saenko. 2024. Sat: Dynamic spatial aptitude training for multimodal language models. *arXiv preprint arXiv:2412.07755*.

Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, Zhaoyang Zeng, Hao Zhang, Feng Li, Jie Yang, Hongyang Li, Qing Jiang, and Lei Zhang. 2024. Grounded sam: Assembling open-world models for diverse visual tasks. *Preprint*, arXiv:2401.14159.

Juan Rocamonde, Victoriano Montesinos, Elvis Nava, Ethan Perez, and David Lindner. 2024. Vision-language models are zero-shot reward models for reinforcement learning. In *International Conference on Learning Representations*, volume 2024, pages 28446–28463.

Nur Muhammad Mahi Shafiullah, Anant Rai, Haritheja Etukuru, Yiqian Liu, Ishan Misra, Soumith Chintala, and Lerrel Pinto. 2023. On bringing robots home. *arXiv preprint arXiv:2311.16098*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Chan Hee Song, Valts Blukis, Jonathan Tremblay, Stephen Tyree, Yu Su, and Stan Birchfield. 2025. Robospatial: Teaching spatial understanding to 2d and 3d vision-language models for robotics. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15768–15780.

Gemini Robotics Team, Abbas Abdolmaleki, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Ashwin Balakrishna, Nathan Batchelor, Alex Bewley, Jeff Bingham, Michael Bloesch, Konstantinos Bousmalis, Philemon Brakel, Anthony Brohan, Thomas Buschmann, Arunkumar Byravan, Serkan Cabi, Ken Caluwaerts, Federico Casarini, and 153 others. 2025. Gemini robotics 1.5: Pushing the frontier of generalist robots with advanced embodied reasoning, thinking, and motion transfer. *Preprint*, arXiv:2510.03342.

Shengbang Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Manoj Middepogu, Sai Charitha Akula, Jihan Yang, Shusheng Yang, Adithya Iyer, Xichen Pan, Austin Wang, Rob Fergus, Yann LeCun, and Saining Xie. 2024. Cambrian-1: A fully open, vision-centric exploration of multimodal llms. In *Advances in Neural Information Processing Systems*, volume 37, pages 87310–87356. Curran Associates, Inc.

Stephen Tyree, Jonathan Tremblay, Thang To, Jia Cheng, Terry Mosier, Jeffrey Smith, and Stan Birchfield. 2022. 6-DoF pose estimation of household objects for robotic manipulation: An accessible dataset and benchmark. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13081–13088.

Xingrui Wang, Wufei Ma, Tiezheng Zhang, Celso M de Melo, Jieneng Chen, and Alan Yuille. 2025. Spatial457: A diagnostic benchmark for 6d spatial reasoning of large multimodal models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 24669–24679.

Dayu Xia, Yue Shi, Yao Mu, Huiting Ji, Chaofan Ma, Yingjie Zhou, Hua Chen, Yang Liu, Jiezhang Cao, and Guangtao Zhai. 2026. Roboprocessbench: Benchmarking process-aware understanding in vision-language robotic manipulation. *arXiv preprint arXiv:2606.13040*.

Jihan Yang, Shusheng Yang, Anjali W. Gupta, Rilyn Han, Li Fei-Fei, and Saining Xie. 2025a. Thinking in space: How multimodal large language models see, remember, and recall spaces. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10632–10643.

Zhutian Yang, Caelan Garrett, Dieter Fox, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. 2025b. Guiding long-horizon task and motion planning with vision language models. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 16847–16853.

Zewei Ye, Weifeng Lu, Minghao Ye, Tao Lin, Shuo Yang, Junchi Yan, and Bo Zhao. 2025. Robofac: A comprehensive framework for robotic failure analysis and correction. *arXiv preprint arXiv:2505.12224*.

Baiqiao Yin, Qineng Wang, Pingyue Zhang, Jianshu Zhang, Kangrui Wang, Zihan Wang, Jieyu Zhang, Keshigeyan Chandrasegaran, Han Liu, Ranjay Krishna, Saining Xie, Manling Li, Jiajun Wu, and Li Fei-Fei. 2025. Spatial mental modeling from limited views. In *Structural Priors for Vision Workshop at ICCV'25*.

Wentao Yuan, Jiafei Duan, Valts Blukis, Wilbert Pumacay, Ranjay Krishna, Adithyavairavan Murali, Arsalan Mousavian, and Dieter Fox. 2024. Robopoint: A vision-language model for spatial affordance prediction for robotics. *arXiv preprint arXiv:2406.10721*.

Chujie Zheng, Hao Zhou, Fandong Meng, Jie Zhou, and Minlie Huang. 2024. Large language models are not robust multiple choice selectors. In *International Conference on Learning Representations*, volume 2024, pages 19426–19454.

Enshen Zhou, Jingkun An, Cheng Chi, Yi Han, Shanyu Rong, Chi Zhang, Pengwei Wang, Zhongyuan Wang, Tiejun Huang, Lu Sheng, and Shanghang Zhang. 2025. Roborefer: Towards spatial referring with reasoning in vision-language models for robotics. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 28404–28481. Curran Associates, Inc.

A Related Works

Robot Manipulation Datasets. Large scale robotics datasets such as Open X-Embodiment (O'Neill et al., 2024) and DROID (Khazatsky et al., 2024) contain robot manipulation trajectory data alongside associated images captured from cameras attached to the robot platform. These have been repurposed into evaluating VLMs for auto-regressive value estimation (Ma et al., 2025) and visual question-answering (Chen et al., 2025). While VLM spatial understanding has been explored using images from DROID (Chen et al., 2025), such tasks have focused on the spatial orientation and the interactions between the robot gripper and the object it interacts with in that particular trajectory, with ground truths sourced from the underlying robot gripper data. Existing benchmarks do not evaluate VLMs on their spatiotemporal understanding of the entire scene as viewed from the perspective of the robot, and we aim to address some of these limitations by analyzing the images associated with the trajectory to construct a spatiotemporal ordering task.

Visual Question Answering Tasks. Both open-ended and MCQ-based visual question answering benchmarks have long been favored for evaluating VLMs on 2D spatial reasoning (Hudson and Manning, 2019; Antol et al., 2015)). Such benchmarks have long focused on relative spatial orientation in 2D image space, and recently, this approach has been extended 3D reasoning by several benchmarks that ask VLMs to reason about how close an object is to the camera (Wang et al., 2025; Tong et al., 2024; Zhou et al., 2025; Du et al., 2024; Fu et al., 2025; Ray et al., 2024).

However, such benchmarks suffer from two primary limitations. Firstly, they often ask questions about the relative depth of pairs of objects, which allows the model to guess its answer from two presented options or exhibit selection bias (Zheng et al., 2024; Balepur et al., 2024). To mitigate this issue, some existing VLM reasoning benchmarks have examined evaluating VLMs using pointing-based questions (Yuan et al., 2024; Song et al., 2025). However, framing reasoning questions as pointing questions requires models to both solve the task at hand and learn to point reliably to the solution in image space, making it difficult to isolate VLM spatiotemporal reasoning capabilities. Secondly, several of these benchmarks primarily use images from large Internet-sourced image datasets that are unlikely for present-day robots to observe in a real-world robot manipulation setting. While some embodied reasoning benchmarks (Majumdar et al., 2024; Zhou et al., 2025; Du et al., 2024; Ray et al., 2024) use images and environments from indoor settings such as Matterport3D (Chang et al., 2017), AI2-Thor (Kolve et al., 2017), Proc-Thor (Deitke et al., 2022) and ScanNet (Dai et al., 2017), as well as tabletop datasets like GraspNET-1B (Fang et al., 2020) and HOPE (Tyree et al., 2022). More recent benchmarks, however, have noted that due to the differences in properties like resolution, lighting variations and real-world robot, as well as the sim-to-real gap, it is difficult to generalize from these datasets to images captured during real-world robot manipulation (Chen et al., 2025; Song et al., 2025). We handle these considerations by ensuring that our tasks use images associated with real-world robot manipulation trajectories in the DROID dataset (Khazatsky et al., 2024) and are formatted as spatiotemporal ordering tasks with a combinatorial number of possible answers.

Spatiotemporal Reasoning Benchmarks: Other approaches for evaluating spatial reasoning in multi-modal LLMs have focused on reasoning from multiple limited views (Yin et al., 2025), multiple perspectives (NVIDIA et al., 2025), and object localization tasks (Majumdar et al., 2024). Approaches to temporal ordering have both focused on ordering video frames (NVIDIA et al., 2025) and reasoning about the order of occurrence of events in videos (Yang et al., 2025a; Li et al., 2025). However, such approaches have not evaluated VLMs on reasoning about real-world robot manipulation videos and images, due to the engineering difficulties associated with retrieving accurate ground-truth data for such questions from low-resolution robot camera images, which has only recently started to be explored in concurrent works to ours (Xia et al., 2026). Our approach instead focuses on spatiotemporal ordering tasks with ground truth information that is easily retrievable from the trajectory video or using traditional computer vision models.

B Downstream Task Evaluations

To evaluate reasoning performance of models fine-tuned on a large RoboReason-generated dataset, we train Qwen3-VL-8B using SFT and GRPO on 8.8K task samples of each type. Training was conducted on three systems: one comprising a single

NVIDIA B300 GPU with 262 GB VRAM for 36 hours, one comprising 4 NVIDIA H200 GPUs with 564GB VRAM for 15 hours, and one comprising 8 NVIDIA RTX A6000 GPUs for 3 hours. For temporal checkpoints, we train on the 3 image temporal prediction task. For depth and left-to-right checkpoints, we train on k-object ordering, where k represents all valid objects and labels that remain after the respective filtering pipelines. The training data was constructed with seeds $s = 12345$ for the temporal task, $s = 54321$ for the depth task, and $s = 2025$ for the left-to-right task. While training the 8B models using SFT and GRPO, the seed $s = 42$ was used.

For evaluating downstream reasoning, we evaluate on 1) 150 question-answer pairs in the SAT test set (Ray et al., 2024), 2) 1.9K multiple-choice and 5.9K open-ended robotics failure reasoning questions (with accuracy measured in terms of exact string match for open-ended questions) in the RoboFAC dataset (Ye et al., 2025), and 3) value estimation tasks on 1000 trajectories each from RT-1 (Brohan et al., 2022) and Dobb-E (Shafiullah et al., 2023), using the value estimation task format proposed in Generative Value Learning (GVL) (Ma et al., 2025). For SAT and RoboFAC, we report task results by measuring the percentage of questions that were answered correctly. For the GVL tasks, we report the absolute average Value-Order Correlation (VOC), where the VOC of an ordering is the rank correlation between predicted values and the original chronological order (Ma et al., 2025). For instances where the VLM did not successfully produce a ranking, we assume VOC to be 0.

C Impact of Camera View on Temporal Reasoning

Model Name	Left View	Right View	Wrist Camera	Multi-View
GPT-5	**50.67**	46.67	47.67	47.33
GPT-5-Nano	**26.33**	21.00	22.33	23.67
Gemini-ER-1.5	42.33	42.00	**45.00**	38.00
Qwen3-VL-235B	**36.00**	28.67	35.33	32.67

Table 10: High Reasoning Model Accuracies (%) for Single-View Image in 3 Image Base Setting for Temporal Ordering

In the temporal ordering task, unlike the spatial reasoning tasks, we provide the model with multiple camera views in the canonical setting. This is primarily because unlike the depth and left-to-right ordering tasks, minute differences between frames are often helpful for human evaluation of the temporal task. In the DROID setup, these differences may sometimes be occluded in specific camera views, necessitating the provision of all 3 camera views.

To evaluate if VLMs perform better when provided with one of the camera views, we run an ablation providing only the left, right, and wrist camera view to the VLM (with the high reasoning setting) in the 3 image ordering version of the task. Since this is simply the base setting of the task, no images had any overlays on them. We present our results in Table 10. For all models and all views (except GPT-5-Nano and Qwen3-VL-235B while using the right-view image), using a single view for temporal ordering performs better than or comparable to (i.e. with a difference in performance smaller than 2%) using an image composited with multiple camera views. This suggests that VLMs struggle to synthesize information from multiple views to reason about the scene in the temporal ordering task. This behavior is hence consistent with results on benchmarks like MindCube (Yin et al., 2025), which show that VLMs find it difficult to answer spatial understanding questions requiring information spread across multiple views.

D Spatiotemporal Ordering Task Construction and Prompts

We design 3 spatiotemporal ordering tasks for evaluating VLM performance. The 2B model finetuning experiments and evaluations are conducted on a machine with 2× Intel(R) Xeon(R) Gold 6342 CPUs (96 cores total) and 8 NVIDIA RTX A6000 GPUs. The underlying robot trajectory images are taken from the DROID dataset (Khazatsky et al., 2024), which has released its data under a CC-BY 4.0 license. We evaluate different input image types on them as follows:

- **Temporal Ordering**: Base Images, Segmentation Map Overlay and Bounding Box mask overlay (in 3-image and 5-image settings across low, medium and high reasoning levels)
- **Depth Ordering**: Base Image, Segmentation Map Overlay and Bounding Box mask overlay (in 3-object and 5-object settings across low, medium and high reasoning levels)
- **Left to Right Ordering**: Base Image, Segmentation Map Overlay and Bounding Box mask

overlay (in 3-object and 5-object settings across low, medium and high reasoning levels)

For each ordering task, we also evaluate models on 2 and 4 image/object settings using the base images across low, medium, and high reasoning levels. Additionally, we also examine if structuring test-time compute helps improve VLM performance by evaluating models in the high reasoning setting using merge sort.

D.1 Temporal Ordering Task

Here the model sees a shuffled sequence of *multiple* images from a robot task sequence and must decide which came first, second, etc. We first detail our algorithm for selecting trajectories from DROID to include in our temporal algorithm task, and then provide prompts used in the temporal ordering tasks in various settings.

D.1.1 Temporal Filtering Algorithm for DROID

Input: Total sequences N, target count K, output directory D
Output: K downloaded sequences with metadata

Generate K random indices: $\mathcal{I} = \{i_1, i_2, \ldots, i_K\}$ where $i_j \in [1, N]$

Load dataset $\mathcal{D}$ from remote storage
Initialize metadata $\mathcal{M} = \{\}$
$count \leftarrow 0$
for each episode $(idx, episode)$ in $\mathcal{D}$ **do**
  if $idx \notin \mathcal{I}$ **then**
    continue ▷ Skip non-target indices
  end if
  $task_desc \leftarrow$ extract task description from $episode$
  $steps \leftarrow$ list of steps in $episode$
  $length \leftarrow |steps|$
  if $length = 0$ **or** $task_desc$ is empty **or** $task_desc =$ "No action" **then**
    continue ▷ Skip invalid sequences
  end if
  $seq_num \leftarrow count + 1$
  Create directory D/sequence_{seq_num}
  for each step s in $steps$ **do**
    Extract image from s.observation
    Save as D/sequence_{seq_num}/step_i.png
  end for
  $\mathcal{M}[seq_num] \leftarrow$ {original_index : idx, task : $task_desc$, length : $length$}
  $count \leftarrow count + 1$
  if $count = K$ **then**
    break ▷ All target sequences downloaded
  end if
end for

Save metadata $\mathcal{M}$ to D/download_metadata.json
return $\mathcal{M}$

n-image Prompt for GPT models

```
if num_images == 2:
        order_desc = "first, second"
        example_order = "[2, 1]"
    elif num_images == 3:
        order_desc = "first, second, third"
        example_order = "[3, 1, 2]"
    elif num_images == 4:
        order_desc = "first, second, third,
     fourth"
        example_order = "[3, 1, 4, 2]"
    else:  # num_images == 5
        order_desc = "first, second, third,
     fourth, fifth"
        example_order = "[3, 1, 5, 2, 4]"

prompt = (
        f'Look at these {num_images} shuffled
     images from a robot task sequence. '
        f'The task is:
     "{task_description}".\n\n'
        f'The red numbers (1..{num_images}) are
     post-shuffle labels - they do NOT reflect
     chronology.\n'
        f"Your task: Determine which labeled
     image came {order_desc}.\n\n"
        f"Respond with ONLY a JSON array of
     numbers in chronological order (e.g.,
     {example_order}).\n"
    )
```

This was used for 2,3,4, and 5 randomly shuffled images when provided to GPT models for temporal ordering.

n-image Prompt for non-GPT models

```
if num_images == 2:
        example_out = "```[2, 1]```"
    elif num_images == 3:
        example_out = "```[3, 1, 2]```"
    elif num_images == 4:
        example_out = "```[3, 1, 4, 2]```"
    else:  # num_images == 5
        example_out = "```[3, 1, 5, 2, 4]```"

    prompt = (
        f"{task_description}\n\n"
        f"These {num_images} images are
     shuffled. Red labels 1..{num_images} are
     just shuffle labels (not
```

```
chronological).\n\n"
    f"Task: Determine the chronological
order of these images. Which labeled image
came first, second, third, etc.?\n\n"
    f"Answer as a bracketed list of labels
inside backticks, earliest to latest.\n"
    f"Example: {example_out}"
)
```

This was used for 2,3,4, and 5 randomly shuffled images when provided to non-GPT models for temporal ordering.

n-image Prompt for VLMs in the Segmentation Map and Bounding Box Settings

When we evaluate VLMs in the bounding box or segmentation map setting, the prompt is modified to include the following line:

```
f'Look at these {num_images} shuffled images
    from a robot task sequence with
    {viz_type_desc}. '
```

where `viz_type_type` could be "Bounding Box" or "Segmentation Map."

3/5-image Prompt for the Merge Sort Setting

```
prompt = (
        f'Look at these 2 images from a robot
    task sequence. '
        f'The task is:
    "{task_description}".\n\n'
        f"These images are from the same
    sequence but at different time points.\n"
        f"Your task: Determine which image came
    FIRST in the sequence.\n\n"
        f"Respond with a JSON object
    containing:\n"
        f'- "first_image": number of the image
    that came first (1 or 2)\n'
        f'- "confidence": your confidence level
    (0.0 to 1.0)\n'
        f'- "reasoning": brief explanation of
    your decision\n\n'
        f'Example: {{"first_image": 1,
    "confidence": 0.8, "reasoning": "The robot
    arm is closer to the object in image 1"}}'
    )
```

D.2 Depth Ordering Task

Here, the model is asked to correctly predict the ordering of objects based on their distance from the cameras. The objects in consideration are provided after filtering out non-overlapping, non-contained objects from the scene using Grounding DINO.

D.2.1 Depth Filtering Algorithm for DROID

Input: Total sequences N, target count K, output directory D, min objects M

Output: K sequences with $\geq M$ non-overlapping objects and depth ordering

Generate K random indices: $\mathcal{I} = \{i_1, i_2, \ldots, i_K\}$ where $i_j \in [1, N]$

Load dataset $\mathcal{D}$ from remote storage
Initialize accepted sequences $\mathcal{A} = \{\}$
$count \leftarrow 0$
for each episode $(idx, episode)$ in $\mathcal{D}$ **do**
  if $idx \notin \mathcal{I}$ **then**
    continue ▷ Skip non-target indices
  end if
  $steps \leftarrow$ list of steps in $episode$
  $length \leftarrow |steps|$
  if $length = 0$ **then**
    continue ▷ Skip empty sequences
  end if

  Step 1: Random Image Selection
  $step_idx \leftarrow$ random integer in $[0, length - 1]$
  $selected_step \leftarrow steps[step_idx]$
  Extract concatenated image I from $selected_step$.observation

  Step 2: DINO Object Detection
  Run DINO detector on image I with prompt
  $\mathcal{B} \leftarrow$ set of detected bounding boxes $\{b_1, b_2, \ldots, b_n\}$
  $\mathcal{L} \leftarrow$ set of object labels $\{l_1, l_2, \ldots, l_n\}$
  if $|\mathcal{B}| < M$ **then**
    continue ▷ Insufficient objects detected
  end if

  Step 3: SAM2 Segmentation
  For each bounding box $b_i \in \mathcal{B}$:
    Run SAM2 segmenter on image I with box b_i
    $m_i \leftarrow$ segmentation mask for object i
  $\mathcal{M} \leftarrow$ set of masks $\{m_1, m_2, \ldots, m_n\}$

  Step 4: Marigold Depth Estimation
  Run Marigold depth estimator on image I
  $D \leftarrow$ depth map for entire image

  Step 5: Depth Extraction and Validation
  For each mask $m_i \in \mathcal{M}$:
    $d_i \leftarrow$ mean depth value in region m_i from depth map D
  $\mathcal{D}_{\text{obj}} \leftarrow$ set of object depths $\{d_1, d_2, \ldots, d_n\}$
  Check pairwise IoU between all masks

$\mathcal{M}_{\text{non-overlap}} \leftarrow$ subset of masks with IoU $< \theta$ for all pairs
$num_objects \leftarrow |\mathcal{M}_{\text{non-overlap}}|$
if $num_objects < M$ **then**
continue ▷ Insufficient non-overlapping objects
end if

Step 6: Depth Ordering
Sort objects by depth: $\mathcal{O}_{\text{sorted}} \leftarrow$ sort $\mathcal{M}_{\text{non-overlap}}$ by d_i (closest to farthest)
Generate depth ordering: ordering $\leftarrow$ list of object labels in depth order

Step 7: Accept and Download
$seq_num \leftarrow count + 1$
Create directory D/sequence_{seq_num}
Save I as D/sequence_{seq_num}/original_image.jpg
Save left portion of I as D/sequence_{seq_num}/left_image.jpg
Save image with bounding boxes as D/sequence_{seq_num}/prompt_image_with_boxes.jpg
Save image with masks as D/sequence_{seq_num}/prompt_image_with_masks.jpg
Save pipeline results as D/sequence_{seq_num}/analysis_results.json
$\mathcal{A}[seq_num] \leftarrow \{$objects : $num_objects$, ordering : ordering, droid_index : $idx\}$
$count \leftarrow count + 1$
if $count = K$ **then**
break ▷ All target sequences downloaded
end if
end for

Save metadata $\mathcal{A}$ to D/split_metadata.json
return $\mathcal{A}$

General VLM Prompt

The general VLM prompt is as follows:

```
"vlm_prompt": "order the following objects
from closest to farthest from camera in
terms of their centroid in the image: obj1,
obj2, obj3, ...",
```

n-object Prompt for GPT Models

```
prompt = (
        f'Look at this image from a robot task
    scene.\n\n'
        f'{vlm_prompt}\n\n'
        f'IMPORTANT: You must use ONLY the
    exact object names provided in the prompt
    above. '
        f'Do not invent new object names or use
    variations. Your response must be a JSON
    array. '
        f'Example: If objects provided are
    elephant, tiger, dragon and correct
    prediction of closest to farthest from
    camera is tiger, dragon, elephant then
    output the json with the order:
    {json.dumps(["tiger", "dragon",
    "elephant"])}. Do not include any other
    text or explanation.'
    )
```

This was used for the setting where 2,3,4, and 5 objects are present in a image provided to GPT-5 and GPT-5-Nano for depth ordering.

n-object Prompt for non-GPT models

```
prompt = (
        f'Look at this image from a robot task
    scene.\n\n'
        f'{vlm_prompt}\n\n'
        f'IMPORTANT: You must use ONLY the
    exact object names provided in the prompt
    above. '
        f'Do not invent new object names or use
    variations. For example '
        f'Example: If objects provided are
    elephant, tiger, dragon and correct
    prediction of closest to farthest from
    camera is tiger, dragon, elephant then
    output the string ```["tiger", "dragon",
    "elephant"]```'
    )
```

n-object Prompt for VLMs in the Segmentation Map and Bounding Box Settings

When we evaluate VLMs in the bounding box or segmentation map setting, the prompt is modified to include the following line:

```
f'The image shows {num_objects} objects
    highlighted with
    {visualization_description} (colored
    annotations).\n\n'
```

where `visualization_description` could be "Bounding Box" or "Segmentation Map."

3/5-object Prompt for the Merge Sort Setting

```
prompt = (
        f'Look at this image from a robot task
    scene.\n\n'
        f'{vlm_prompt}\n\n'
        f'Your task: Compare the two objects
    "{obj1}" and "{obj2}" and determine which
    one is CLOSER to the camera.\n\n'
```

```
    f'IMPORTANT: You must respond with ONLY
one of these exact object names: "{obj1}"
or "{obj2}". '
    f'Do not use variations or invent
names.\n\n'
    f'Respond with a JSON object
containing:\n'
    f'- "closer_object": the name of the
object that is closer (must be exactly
"{obj1}" or "{obj2}")\n'
    f'- "confidence": your confidence level
(0.0 to 1.0)\n'
    f'- "reasoning": brief explanation of
your decision\n\n'
    f'Example: If objects provided are
elephant and tiger, and elephant is closer
to the camera, then output:
{{"closer_object": "elephant",
"confidence": 0.8, "reasoning": "Object
appears larger and more prominent"}}'
)
```

D.3 Left-to-Right Ordering Task

Here, the model is asked to correctly predict the left-to-right ordering of objects.

D.3.1 Left-to-right Filtering Algorithm for DROID

Input: Image directory I, output directory O, min objects M
Output: Dataset of images with left-to-right ordered objects

Parameters:
min_gap ← minimum horizontal gap between objects (default: 0.05)
max_iou ← maximum IoU between objects (default: 0.7)
max_vertical_dist ← maximum vertical distance (default: 0.5)
max_width_ratio ← maximum box width ratio (default: 0.7)

Sample N sequences from I
for each sequence S in sampled sequences **do**
Step 1: Image Selection
$img \leftarrow$ random image from S
Load image dimensions: $(W, H) \leftarrow$ size of img
Extract left portion: $img \leftarrow img[:, : W \times 0.33]$

Step 2: Object Detection
Stage 1: Query GPT-5 with img to identify object categories
Stage 2: Run Grounding DINO with GPT-5 labels as text prompts
Extract results: $\mathcal{L} \leftarrow$ labels, $\mathcal{B} \leftarrow$ bboxes, $\mathcal{S} \leftarrow$ scores
if $|\mathcal{L}| < M$ **then**
continue ▷ Insufficient detections
end if

Step 3: Scene Object Filtering
$\mathcal{L}_{ns} = \{\}, \mathcal{B}_{ns} = \{\}, \mathcal{S}_{ns} = \{\}$
$count_{scene} \leftarrow 0$
for each object $i \in \{1, \ldots, |\mathcal{L}|\}$ **do**
Compute box dimensions: $w_i \leftarrow \mathcal{B}_i[2] - \mathcal{B}_i[0]$, $h_i \leftarrow \mathcal{B}_i[3] - \mathcal{B}_i[1]$
Compute width ratio: $r_w \leftarrow w_i/W$
if $r_w >$ max_width_ratio **then**
$count_{scene} \leftarrow count_{scene} + 1$
continue ▷ Filter large scene objects
end if
$\mathcal{L}_{ns} \leftarrow \mathcal{L}_{ns} \cup \{\mathcal{L}_i\}$
$\mathcal{B}_{ns} \leftarrow \mathcal{B}_{ns} \cup \{\mathcal{B}_i\}$
$\mathcal{S}_{ns} \leftarrow \mathcal{S}_{ns} \cup \{\mathcal{S}_i\}$
end for
if $|\mathcal{L}_{ns}| < M$ **then**
continue ▷ Insufficient objects after scene filtering
end if

Step 4: Clear Ordering Filter
Step 4a: Sort by left edge
Sort by x-coordinate: $idx_{sorted} \leftarrow$ argsort($\mathcal{B}_{ns}[:, 0]$)
$\mathcal{L}_{sorted} \leftarrow [\mathcal{L}_{ns}[i] : i \in idx_{sorted}]$
$\mathcal{B}_{sorted} \leftarrow [\mathcal{B}_{ns}[i] : i \in idx_{sorted}]$
$\mathcal{S}_{sorted} \leftarrow [\mathcal{S}_{ns}[i] : i \in idx_{sorted}]$
Step 4b: Vertical clustering
Compute y-centers: $y_c[i] \leftarrow (\mathcal{B}_{sorted}[i][1] + \mathcal{B}_{sorted}[i][3])/2$ for all i
$best_cluster \leftarrow \emptyset$
$best_size \leftarrow 0$
for each object $i \in \{1, \ldots, |\mathcal{L}_{sorted}|\}$ **do**
$cluster_i \leftarrow \emptyset$
for each object $j \in \{1, \ldots, |\mathcal{L}_{sorted}|\}$ **do**
$y_{diff} \leftarrow |y_c[j] - y_c[i]|$
if $y_{diff}/H \leq$ max_vertical_dist **then**
$cluster_i \leftarrow cluster_i \cup \{j\}$
end if
end for
if $|cluster_i| > best_size$ **then**
$best_cluster \leftarrow cluster_i$
$best_size \leftarrow |cluster_i|$
end if

end for
if $|best_cluster| < M$ **then** ▷ Insufficient vertically-aligned objects
end if
$\mathcal{L}_{clust} \leftarrow [\mathcal{L}_{sorted}[i] : i \in best_cluster]$
$\mathcal{B}_{clust} \leftarrow [\mathcal{B}_{sorted}[i] : i \in best_cluster]$
$\mathcal{S}_{clust} \leftarrow [\mathcal{S}_{sorted}[i] : i \in best_cluster]$
Step 4c: Remove highly overlapping boxes
$to_keep \leftarrow []$ ▷ List of indices to keep (maintains insertion order)
for each object $i \in \{0, \ldots, |\mathcal{L}_{clust}| - 1\}$ **do**
$keep \leftarrow$ True
for each index $j \in to_keep$ **do**
$iou_{ij} \leftarrow$ IoU($\mathcal{B}_{clust}[i], \mathcal{B}_{clust}[j]$)
if $iou_{ij} >$ max_iou **then** ▷ High overlap detected
if $\mathcal{S}_{clust}[i] > \mathcal{S}_{clust}[j]$ **then**
Remove j from to_keep ▷ Current has higher confidence
else
$keep \leftarrow$ False ▷ Existing has higher confidence
break ▷ Skip current object
end if
end if
end for
if $keep$ **then**
Append i to to_keep
end if
end for
$\mathcal{L}_{clear} \leftarrow [\mathcal{L}_{clust}[i] : i \in to_keep]$
$\mathcal{B}_{clear} \leftarrow [\mathcal{B}_{clust}[i] : i \in to_keep]$
$\mathcal{S}_{clear} \leftarrow [\mathcal{S}_{clust}[i] : i \in to_keep]$
if $|\mathcal{L}_{clear}| < M$ **then**
continue ▷ Insufficient unambiguous orderings
end if
Step 5: Remove Duplicate Labels
$keep_idx \leftarrow \{i : |\{j : \mathcal{L}_{clear}[j] = \mathcal{L}_{clear}[i]\}| = 1\}$
$\mathcal{L}_{final} \leftarrow [\mathcal{L}_{clear}[i] : i \in keep_idx]$
$\mathcal{B}_{final} \leftarrow [\mathcal{B}_{clear}[i] : i \in keep_idx]$
$\mathcal{S}_{final} \leftarrow [\mathcal{S}_{clear}[i] : i \in keep_idx]$
if $|\mathcal{L}_{final}| < M$ **then**
continue ▷ Insufficient objects or ambiguous ordering
end if
Save to O_S
end for

General VLM Prompt

The general VLM prompt is as follows:

```
"vlm_prompt": "Arrange the following
objects in order from leftmost to rightmost
in the image: obj1, obj2, obj3, ..."
```

n-object Prompt for all VLMs

`selected_objects` A list of n randomly shuffled detected object labels, e.g., `["table", "chair", "door"]`.

```
prompt = (
    "Arrange the following objects in order
    from leftmost to rightmost in the image: "
    + ", ".join(selected_objects)
    + ".\nReturn only the ordered list in JSON
    format: "
    + '{"ordered_objects": ["obj1", "obj2",
    ...]}'
)
```

These prompts were used for cases with 2, 3, 4, and 5 objects present in an image (or an image overlaid with segmentation maps or bounding boxes), where the VLMs were expected to correctly predict their left-to-right order.

3/5-object Prompt for the Merge Sort Setting

```
prompt = (
    f"Which object is to the LEFT of the
other in the image: '{obj1}' or '{obj2}'?\n"
    f"Return ONLY the name of the leftmost
object in JSON format: "
    f'{{"leftmost_object": "obj_name"}}'
)
```

E Reinforcement Learning Experiments

This section collates the reinforcement learning (RL) details of our experiments: prompts, the final output token limit, and training curves.

E.1 Overview of Experiments

We ran six GRPO RL experiments on 2B models described in Table 8:

- Two runs for **Temporal Ordering** (3-image and 5-image settings)
- Two runs for **Depth Ordering** (3-object and 5-object settings)
- Two runs for **Left-to-Right Ordering** (3-object and 5-object settings)

E.2 Training Configuration

Here we summarise the key configuration choices made during RL-finetuning.

Reward functions. We experimented with a variety of reward signal formulations:

- A fully sparse reward (e.g., +1 for correct ordering, 0 otherwise)
- A rank-based score using Spearman's rank correlation (i.e., penalising the deviation of the predicted ordering from the ground truth in Spearman's rho)
- Ultimately we settled on a Kendall τ based reward (percentage of correct pairwise orderings)

RL algorithm and scale. We used the off-the-shelf GRPO algorithm (Group Relative Policy Optimization) In our setup:

- We trained on 700 unique prompt examples.
- We conducted training for 1,000 gradient/update steps $\sim$ 1.43 epochs.
- Per prompt, we sampled 6 generations at each step (group size = 6).

E.3 Global Settings

Max output tokens: We set the maximum output token limit to **700 tokens** across all 2B RL tasks.

Reasoning first: In all tasks, we nudge the model to write a short English reasoning trace before clearly stating the final ordering. That lets us treat the text as a trajectory for optimization and helps the model overcome any default ordering bias.

E.4 Temporal Ordering Task (with stamping + randomized example output)

Here the model sees a shuffled sequence of *multiple* images (either 3 or 5) from a robot task sequence and must decide which came first, second, etc.

Key implementation details (temporal only).

- **Stamped labels:** We overlay small red numeric markers on each image so the model can unambiguously refer to them (1, 2, 3, . . .). This stamping is *only used* for the temporal task.
- **Randomized example output:** Each training example includes a *randomized* final-answer example like `[3, 1, 2]`. The randomization is deterministic per example ID and helps avoid mode collapse onto a single example format. This randomization is *only used* for the temporal task.

3-image version (system prompt skeleton).

```
Look at these 3 shuffled images from a robot
    task sequence.
The robot task is: "<task description>".

The red numbers (1, 2, 3) are post-shuffle
    labels -
they do NOT reflect chronology.
Your task: Reason about which labeled image
    came first,
second, third.

Feel free to write your process/reasoning in
    English
before outputting your final answer.
Example final answer (for formatting only):
if the correct order is third, first, second,
then at the end output [3, 1, 2]
```

The example final answer is randomized throughout training.

5-image version (system prompt skeleton). Same as above, but for five images and with labels `1..5`.

E.5 Depth Ordering Task

For depth, the model sees a *single* image from the robot workspace and must order objects from closest to farthest .

Prompt.

```
You are a visual reasoner. You will be shown a
    single image
from a robot workspace. You may write notes or
    reason before
answering, but make sure to clearly state the
    final ordering.

Order the following objects from closest to
    farthest from camera
in terms of their centroid in the image:
robot arm, curtain, window, blanket, pillow,
    table
```

E.6 Left-to-Right Ordering Task

Similarly, for left-to-right we use one image and ask the model to order objects horizontally from left to right.

Prompt.

```
You are a visual reasoner. You will be shown a
    single image
from a robot workspace. You may write notes or
    reason before
answering, but make sure to clearly state the
    final ordering.
```

E.7 Training Curves

Figures 1-6 show the reward curves for all six GRPO runs (temporal, depth, left-to-right; 3-object/image and 5-object/image) for Qwen2-VL-2B-Instruct.

Notes.

- Stamping and randomized example outputs are **temporal-only**. Depth and left-to-right use a single image, no stamping, no randomized example.
- All RL runs used a max generation budget of **700 tokens**.
- The reward on the y-axis is the kendall reward (fraction of correct pairwise orderings in model output)

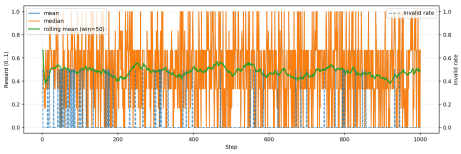


Figure 3: Temporal task, 3 img

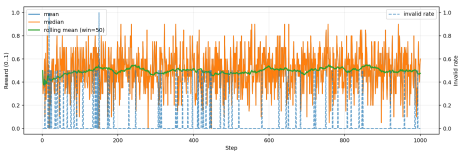


Figure 4: Temporal task, 5 image

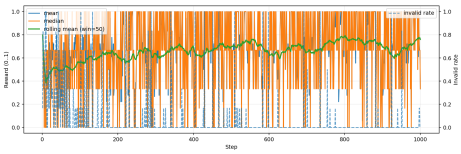


Figure 5: Depth 3 Objects

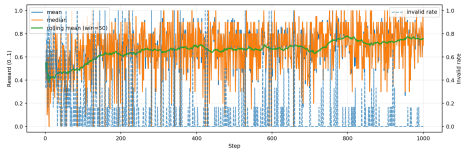


Figure 6: Depth task, 5 objects

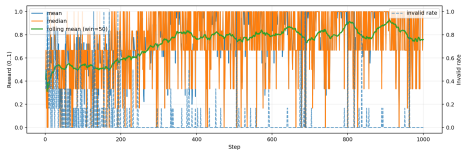


Figure 7: Left-to-Right task, 3 Objects

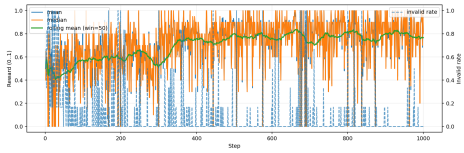


Figure 8: Left-to-Right task, 5 Objects

F Task Examples

In this section, we provide examples of temporal, depth and spatial ordering tasks in the base as well as bounding box and segmentation map settings.

F.1 Examples of Temporal Ordering Tasks

We show examples of temporal ordering tasks across three settings: base (Figures 9-11), bounding box (Figure 12), and segmentation map (Figure 13).

Base Setting Tasks

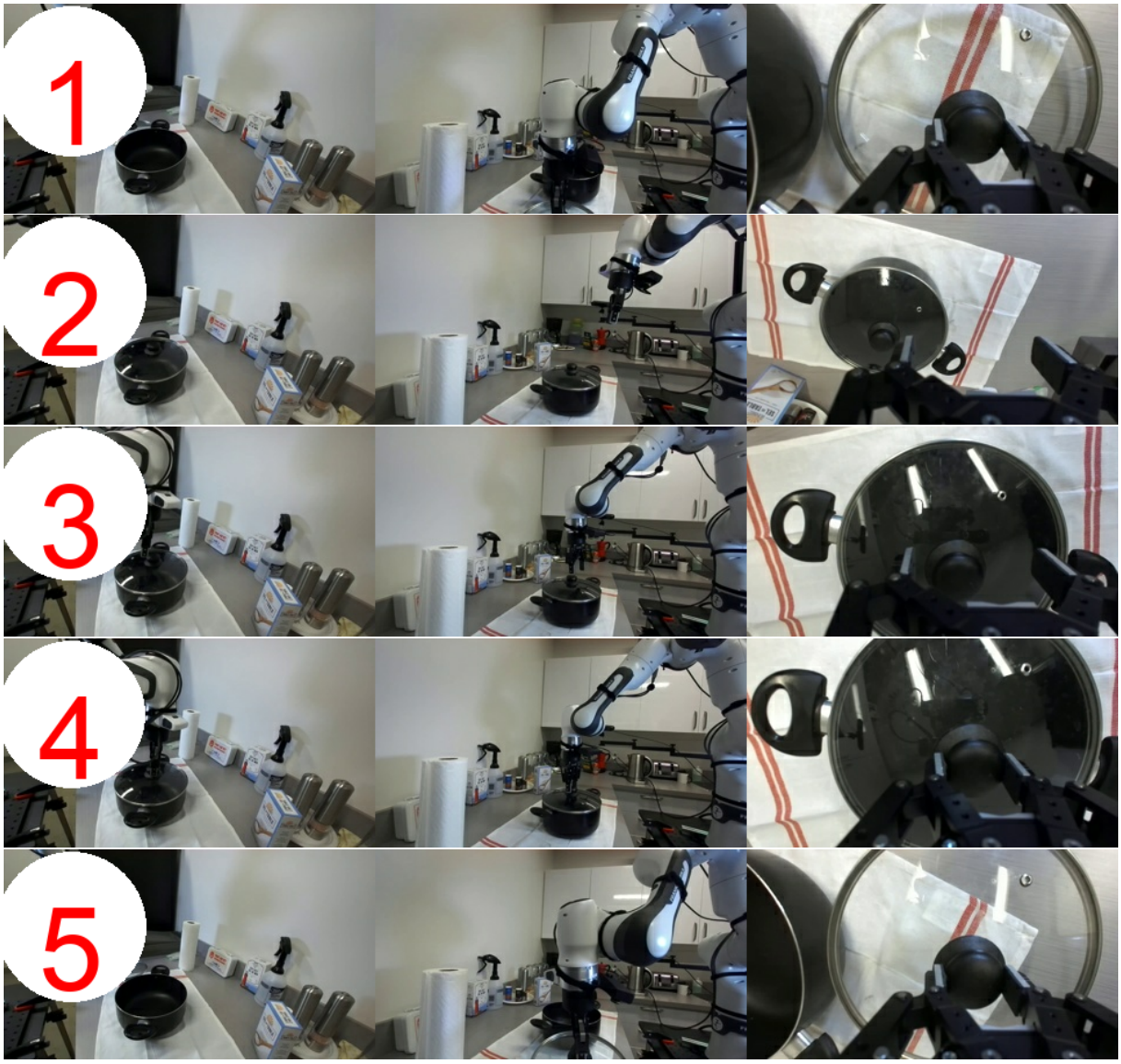


Figure 9: Task Description: “Remove the lid from the pot | Remove the lid from the black pot. | Remove the glass lid from the pot." Correct Order: 2,3,4,5,1

Figure 10: Task Description: “Pick up the object and put it in the light blue cup | Place the metal object inside the light blue cup | Put the silver object in the light blue cup"
Correct Order: 2,1,4,5,3

Figure 11: Task Description: “Pick up the third spice bottle from the right on the bottom shelf of the bottom spice rack and put it on the table | Remove the third container from the right on the bottom rack and put it on the table | Put the spice bottle on the counter"
Correct Order: 2,5,3,4,1

Bounding Box Setting Task

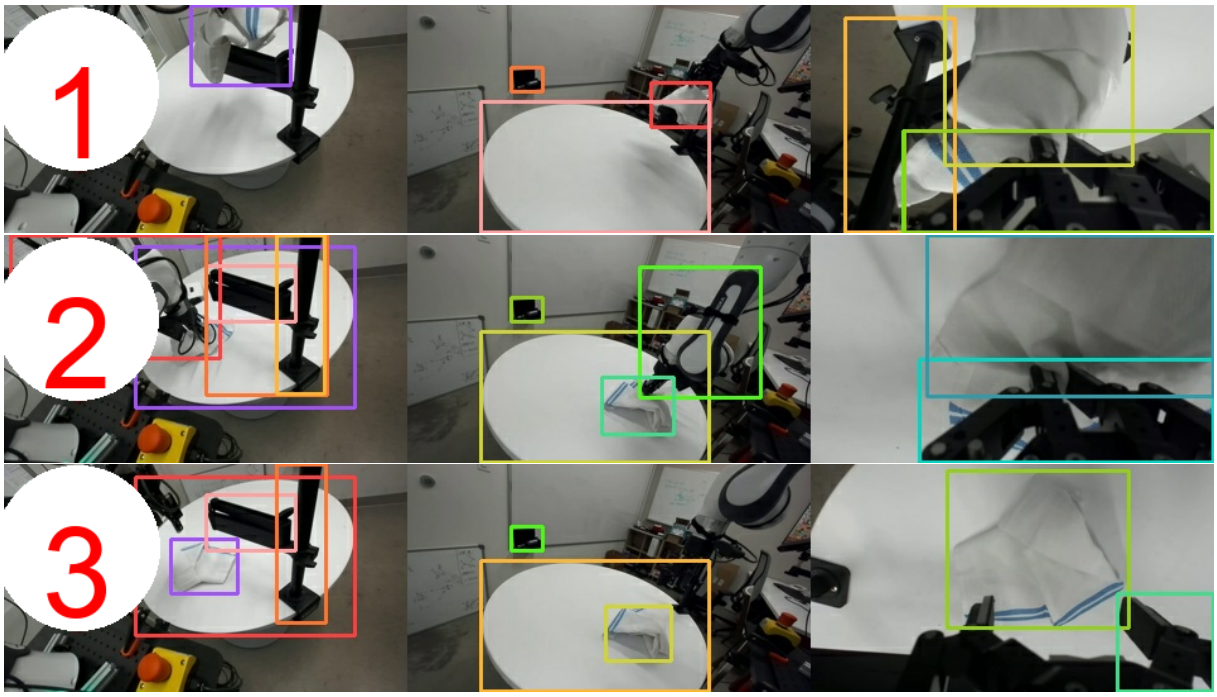


Figure 12: Task Description: “Hang the towel on the black stand | Put the towel on the black rod | Pick up the cloth from the table and place it on the black object"
Correct Order: 3, 2, 1

Segmentation Map Setting Task

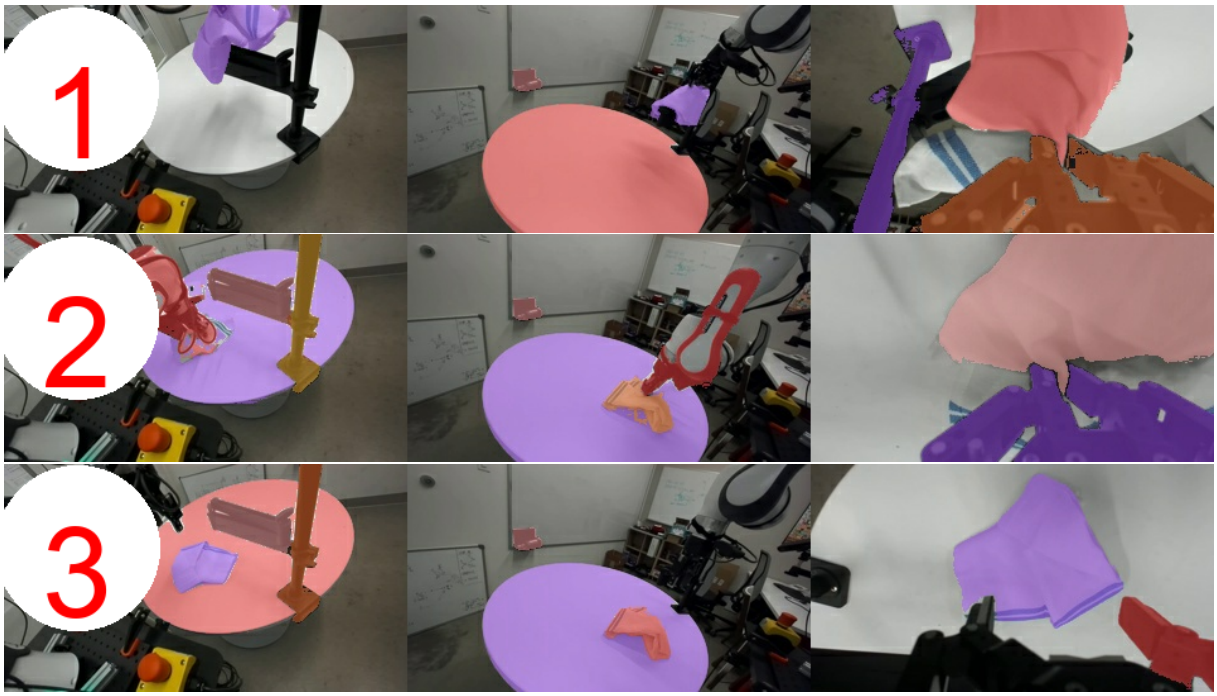


Figure 13: Task Description: “Hang the towel on the black stand | Put the towel on the black rod | Pick up the cloth from the table and place it on the black object"
Correct Order: 3, 2, 1

F.2 Examples of Depth Ordering Tasks

We show examples of depth ordering tasks across three settings: base (Figures 14-16), bounding box (Figure 17), and segmentation map (Figure 18) .

F.2.1 Base Setting Task

Figure 14: “order the following objects from closest to farthest from camera in terms of their centroid in the image: bin, robot arm, bowl, bottle, bowl cup"
Correct Order: "robot arm", "bottle", "bowl cup", "bowl", "bin"

Figure 15: “order the following objects from closest to farthest from camera in terms of their centroid in the image: container, cup, bottle, tray, robot arm"
Correct Order: "container", "cup", "tray", "robot arm", "bottle"

Figure 16: “order the following objects from closest to farthest from camera in terms of their centroid in the image: sink, faucet, trash bin, cardboard box packet, coffee maker"
Correct Order: "cardboard box packet", "trash bin", "coffee maker", "sink", "faucet"

Bounding Box Setting Task

Figure 17: “order the following objects from closest to farthest from camera in terms of their centroid in the image: napkin dispenser, paper towel roll, coffee machine"
Correct Order: "paper towel roll", "napkin dispenser", "coffee machine"

Segmentation Map Setting Task

Figure 18: “order the following objects from closest to farthest from camera in terms of their centroid in the image: napkin dispenser, paper towel roll, coffee machine"
Correct Order: "paper towel roll", "napkin dispenser", "coffee machine"

F.3 Examples of Left-to-Right Tasks

We show examples of left-to-right ordering tasks across three settings: base (Figures 19-21), bounding box (Figure 22), and segmentation map (Figure 23).

Base Setting Task

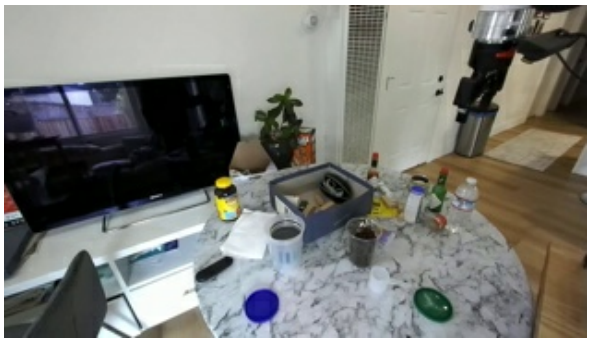


Figure 19: “Arrange the following objects in order from leftmost to rightmost in the image: rug, chair, plant, table, door"
Correct Order: "chair", "table", "plant", "door", "rug"

Figure 20: “Arrange the following objects in order from leftmost to rightmost in the image: robot, paper towel, box, microwave, pumpkin"
Correct Order: "robot", "pumpkin", "paper towel", "microwave", "box"

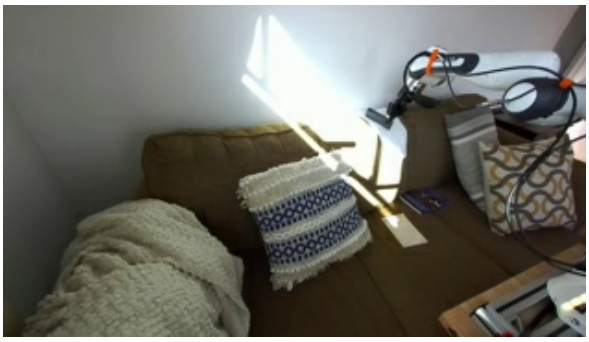


Figure 21: “Arrange the following objects in order from leftmost to rightmost in the image: blanket, lamp, pillow blanket, table, book"
Correct Order: "blanket", "pillow blanket", "lamp", "book", "table"

Bounding Box Setting Task

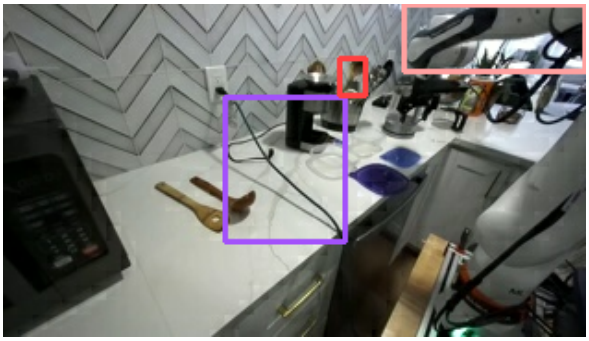


Figure 22: “Arrange the following objects in order from leftmost to rightmost in the image: cable, robot arm, spoon"
Correct Order: "cable", "spoon", "robot arm"

Segmentation Map Setting Task

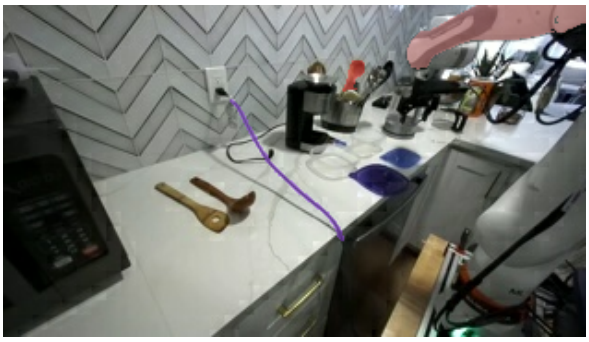


Figure 23: “Arrange the following objects in order from leftmost to rightmost in the image: cable, robot arm, spoon"
Correct Order: "cable", "spoon", "robot arm"

G LLM Usage

We used LLM-generated code (which was reviewed by the authors) for implementing some experi-

ments.

H Token Limits for Reasoning Levels

# Images/Objects	Temporal Ordering Task			Depth Ordering Task			Left-to-Right Ordering Task		
	Low	Medium	High	Low	Medium	High	Low	Medium	High
2	419	1204	2912	721	1487	5839	960	1664	5376
3	535	1619	3859	788	2067	5778	896	2432	4416
4	668	2062	5012	985	2262	6745	1024	2496	4992
5	801	2496	121,436	1052	2781	7194	1216	3136	6784
Mergesort 3	N/A	N/A	5701	N/A	N/A	2653	N/A	N/A	6400
Mergesort 5	N/A	N/A	6036	N/A	N/A	2800	N/A	N/A	11264

Table 11: Maximum output tokens permitted across reasoning levels and number of ordering elements.