SPATIALANYTHING: BIDIRECTIONAL ATTENTION AND MULTI-TEACHER DISTILLATION FOR EMBODIED SPATIAL REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large vision-language models (LVLMs) recently demonstrate promising capabilities in embodied spatial reasoning, spanning spatial understanding, object localization, navigation, and long-horizon planning. These tasks require models to process both language and visual grounding primitives such as bounding boxes, keypoints, and trajectories. However, existing methods typically generate coordinates one at a time through autoregressive decoding, so coordinates within the same visual grounding primitive depend on different contexts and generation time grows with primitive length. Meanwhile, different spatial tasks build on shared localization and spatial relation understanding, yet differ in output formats and optimization objectives. We therefore present SpatialAnything, a natively multimodal model for embodied spatial reasoning built around Primitive Bidirectional Attention (PBA). PBA enables bidirectional attention among coordinate representations within a primitive through bidirectional masks in full-attention layers and query rewriting with write compensation in gated linear-attention layers. SpatialAnything further adopts adaptive parallel decoding, which treats each primitive as a whole and predicts all of its coordinates at once, removing the wait inherent in coordinate-by-coordinate generation. The predictions are then written into the context to support subsequent generation. Building on this, SpatialAnything uses multi-task joint training through a two-stage strategy based on Spatial Multi-Teacher On-Policy Distillation (Spatial MOPD), which first establishes a foundation for localization prediction with a unified primitive teacher and then lets task-specific teachers guide the student on its own generations, giving a single model complementary spatial capabilities. Across a range of embodied spatial reasoning benchmarks, SpatialAnything achieves the highest scores on multiple metrics, exceeding GPT 6 Astra by 5.2 points on average over ten spatial understanding and reasoning benchmarks, and delivers a 2.9$\times$ inference speedup.

1 INTRODUCTION

Large vision-language models (LVLMs) are becoming an important foundation for embodied spatial reasoning. Through large-scale multimodal pretraining, these models align visual observations with language instructions and support tasks ranging from spatial understanding and object localization to navigation and long-horizon planning. These tasks require models to perceive the objects and layout of a scene, locate the targets mentioned in an instruction, judge spatial relationships, and plan the locations or paths needed for subsequent decisions. Models therefore need to process both language and visual grounding primitives such as bounding boxes, keypoints, and trajectories. Language describes spatial relationships and task decisions, while visual grounding primitives specify object regions, interaction locations, and candidate paths, turning language-based judgments into concrete spatial predictions. Accurately representing and rapidly generating the coordinates of visual grounding primitives is thus central to advancing embodied spatial reasoning.

Recent methods develop these capabilities in several directions. Some introduce visual grounding to predict the locations of objects referred to by language (Chen et al., 2023; You et al., 2024; Wang et al., 2026). Others strengthen spatial understanding, focusing on object relationships, depth, and viewpoint changes (Chen et al., 2024; Cheng et al., 2024; Tan et al., 2026). Still others use

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

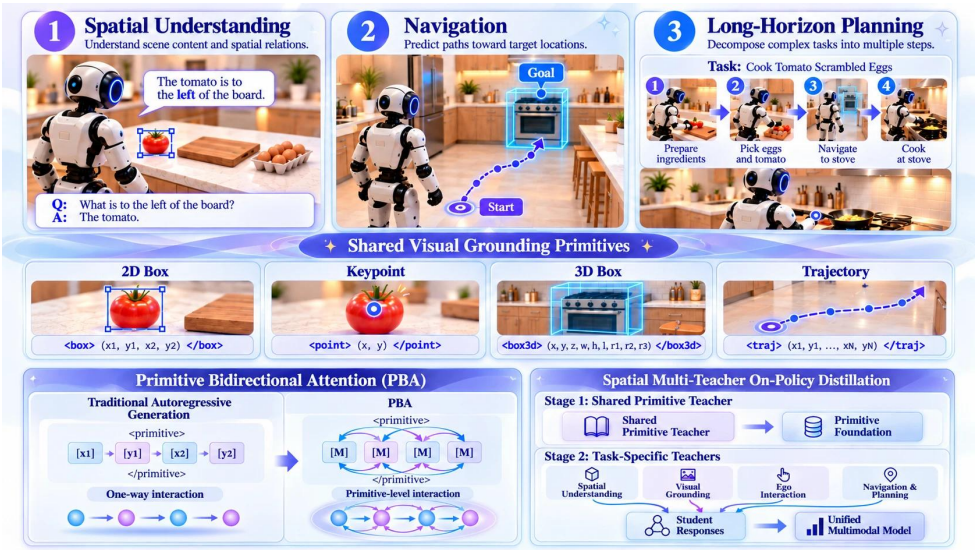


Figure 1: (a) SpatialAnything handles spatial reasoning, localization, egocentric interaction, navigation, and long-horizon planning in a single model, interleaving language with visual grounding primitives. (b) Existing methods generate the coordinates of a primitive one at a time, breaking its structure and scaling latency with its length. (c) SpatialAnything predicts a complete primitive in one forward pass with bidirectional interaction inside it and adapts the decoding granularity to each primitive.

spatial information to support navigation and sequential decision-making (Qwen Team, 2026b; Li et al., 2026; DeepCybo Team, 2026). However, their spatial predictions still follow the token-by-token generation used for language. Such generation gives coordinates within the same primitive different prefixes, passes information only forward, and increases sequential prediction cost with primitive length. Bidirectional interaction inside a primitive allows coordinate predictions to use information from both preceding and subsequent positions, supporting parallel prediction of the complete primitive. As the demand for long-context modeling grows, models increasingly combine full and linear attention to improve sequence-processing efficiency. Bidirectional masks can widen the visible range of full attention, yet they do not directly change how linear attention reads recurrent states at individual positions. Enabling bidirectional interaction within a primitive therefore requires coordinating information access across both attention mechanisms.

To address these challenges, we present SpatialAnything, a natively multimodal model for embodied spatial reasoning built around Primitive Bidirectional Attention (PBA), with 4B, 9B, and 27B variants. PBA realizes bidirectional attention within a primitive across full-attention and gated linear-attention layers. Full-attention layers use bidirectional masks so that coordinates inside a primitive see each other directly. Gated linear-attention layers apply a query transformation so that each coordinate can consult the other coordinates of the same primitive when it is predicted. This design preserves the original state recurrence and the generation order between primitives and language. Building on PBA, adaptive parallel decoding places masked placeholders for all coordinates to be predicted according to the primitive type and specified length. PBA lets these placeholders share bidirectional attention within the primitive, and the model maps their outputs to coordinate values in a single prediction forward pass, reducing sequential prediction steps. The predicted coordinates are then incorporated into the context to support subsequent language and primitive generation.

For training, spatial tasks all build on localization and spatial relation understanding, yet their output formats and optimization objectives differ. To learn the spatial knowledge these tasks share while giving the model the capabilities specific to each, we design a two-stage training strategy based on Spatial Multi-Teacher On-Policy Distillation (Spatial MOPD). In the first stage, a unified primitive teacher transfers primitive knowledge common to all tasks, establishing a shared foundation for localization prediction. In the second stage, task-specific teachers provide distillation guidance on the student's own generations, so the student learns the capabilities each task requires from what it generates. After these two stages, SpatialAnything handles language-based judgments and visual grounding primitive prediction within a single model. Our main contributions are as follows:

- **Primitive-level bidirectional interaction and parallel prediction.** We introduce PBA so that coordinates within the same primitive see each other in both full-attention and gated

linear-attention layers. Combined with adaptive parallel decoding, this design predicts all coordinates of a complete primitive simultaneously and reduces sequential prediction steps.

- **Spatial posttraining that integrates shared knowledge and task-specific expertise.** We design a two-stage Spatial MOPD training strategy that first establishes a foundation for localization prediction through a unified primitive teacher, then has task-specific teachers guide the student on its own generations, giving a single model complementary spatial capabilities across tasks.

- **Multi-scale embodied spatial reasoning models and systematic evaluation.** We build SpatialAnything at 4B, 9B, and 27B scales and evaluate it on spatial understanding, visual grounding, egocentric interaction, and navigation. Among the reported comparisons, SpatialAnything-27B achieves the highest scores on CV-Bench, MindCube, VSI-Bench, and ViewSpatial-Bench, and exceeds the evaluated GPT 6 Astra (OpenAI, 2026) configuration by 5.2 points on average across ten spatial understanding and reasoning benchmarks.

2 Related Work

Multimodal Spatial Understanding. Multimodal large language models such as BLIP-2 (Li et al., 2023), LLaVA (Liu et al., 2023), and the Qwen-VL series (Bai et al., 2023) connect a visual encoder to a language decoder and achieve strong image level understanding, yet they describe what an image contains without committing to where things are. One line of work attaches explicit geometry to language. Shikra (Chen et al., 2023) and Pink (Xuan et al., 2024) encode boxes as text tokens, Ferret (You et al., 2024) accepts points, boxes, and free form shapes, and LISA (Lai et al., 2024) and GLaMM (Rasheed et al., 2024) push the output granularity to pixel level masks. A second line targets metric and relational understanding, where SpatialVLM (Chen et al., 2024) and SpatialRGPT (Cheng et al., 2024) synthesize large scale spatial question answer data, and 3D-LLM (Hong et al., 2023) and LEO (Huang et al., 2024) inject depth, point clouds, or multi view features into the visual interface. Both lines treat spatial ability as a skill layered onto a language backbone. The coordinates they emit remain ordinary tokens in a text stream, so the model receives no structural signal that a group of numbers forms one geometric entity.

Linear Attention and Structured Decoding. Efficient attention replaces pairwise comparison with a fixed size recurrent state. Gated DeltaNet (Yang et al., 2025b) updates this state token by token under a delta rule with forgetting, writing, and normalization gates, and hybrid backbones such as Qwen3-Next and Qwen3.5 (Yang et al., 2025a) stack three linear layers for every full attention layer, which raises long sequence throughput by roughly an order of magnitude. In parallel, structured decoding methods emit a group of coordinates as one block instead of a queue. Parallel block decoding (Wang et al., 2026) edits the attention mask so that tokens inside a block attend to each other, which keeps a box from bending and improves decoding speed. This remedy presumes an attention matrix and therefore applies only to the quarter of layers that expose one. Sana (Xie et al., 2024) demonstrates in video diffusion that controlling the range of state updates yields intra frame bidirectionality without any mask, although it assumes frames of uniform length and carries no autoregressive constraint.

Embodied Models and Spatial Post Training. Vision language action models such as RT-2 (Brohan et al., 2023) and OpenVLA (Kim et al., 2024) map observations and instructions directly to control, navigation models such as Qwen-RobotNav (Qwen Team, 2026b) operate over egocentric streams and long horizons, and a recent generation of embodied foundation models unifies perception, spatial reasoning, and planning inside one backbone (Tan et al., 2026; Li et al., 2026; DeepCybo Team, 2026; Tencent Hunyuan Team, 2026; Hao et al., 2025; NVIDIA, 2026). These systems increasingly depend on post training, using reinforcement learning or distillation from a stronger teacher to sharpen behavior after supervised pretraining. Current recipes optimize each capability as a separate objective, which suits skills that occupy disjoint input distributions. Spatial competences do not, since grounding appears inside navigation and relational reasoning appears inside long horizon planning. Our SpatialAnything addresses both levels at once, combining a geometry aware backbone with an on policy distillation scheme built for overlapping signals.

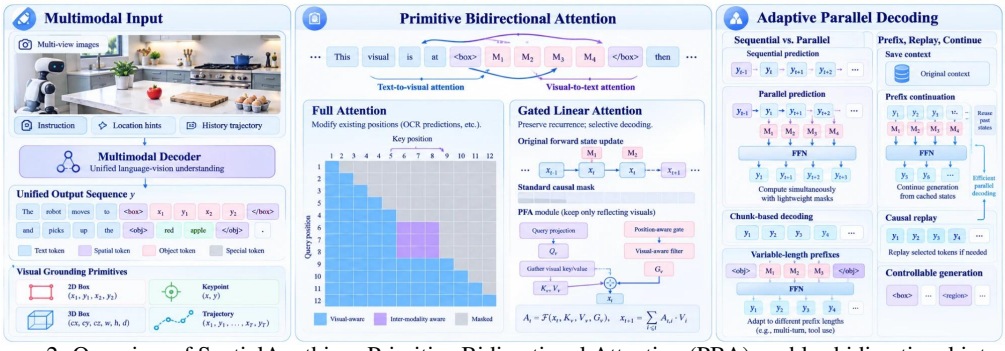


Figure 2: Overview of SpatialAnything. Primitive Bidirectional Attention (PBA) enables bidirectional interaction within each primitive through masks in full-attention layers and query rewriting in gated linear-attention layers, preserving causal order across primitives and during language generation. Adaptive parallel decoding predicts all coordinates of a primitive simultaneously.

3 METHODOLOGY

SpatialAnything is a natively multimodal model for embodied spatial reasoning built around Primitive Bidirectional Attention (PBA), jointly processing language and visual grounding primitives. We present the framework in four parts. Sec. 3.1 introduces the task formulation, visual grounding primitives, and the basic computations of full and gated linear attention. Sec. 3.2 presents PBA and explains how bidirectional interaction across both attention mechanisms supports spatial prediction. Sec. 3.3 introduces adaptive parallel decoding to predict all coordinates of a primitive simultaneously. Sec. 3.4 presents Spatial MOPD, which integrates shared spatial knowledge and task-specific expertise through primitive knowledge warm-up and multi-task on-policy distillation. Figure 2 presents an overview of SpatialAnything.

3.1 PRELIMINARIES

Task formulation. Embodied spatial tasks require models to jointly process language and visual grounding primitives. Given visual input I and a language instruction x, SpatialAnything generates a response y that may interleave language with visual grounding primitives. Language describes spatial relationships and task decisions, while primitives specify the corresponding object regions, interaction locations, and candidate paths.

Visual grounding primitives. We use paired control tags to identify primitive types and their coordinate sequences. The opening tag specifies the primitive type, and the closing tag marks its completion. Each scalar parameter is encoded as one coordinate token. A 2D box uses two opposite corners (x_1, y_1, x_2, y_2) to specify a rectangular object region in an image. A 3D box describes an object through its center, dimensions, and orientation $(x, y, z, w, h, l, r_1, r_2, r_3)$. A keypoint uses a 2D coordinate (x_1, y_1) to specify a target point or interaction location. A trajectory describes a candidate route through N ordered 2D waypoints $(x_1, y_1, \ldots, x_N, y_N)$. Although primitives differ in type and length, the coordinates within each primitive jointly describe one spatial output. We therefore use complete primitives to organize subsequent information interaction and coordinate prediction.

Full and gated linear attention. SpatialAnything combines full-attention and gated linear-attention layers to process multimodal sequences. Full attention explicitly computes similarities between queries and keys and aggregates the corresponding values:

$$Y = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d_k}} + M\right)V. \tag{1}$$

Here, Q, K, and V are queries, keys, and values projected from the inputs, d_k is the key dimension, and M is the attention mask. The mask controls information access. A causal mask allows each position to access itself and preceding tokens, while a bidirectional mask allows mutual access within a specified region.

Gated linear attention accumulates sequence information in a fixed-size state without explicitly constructing the complete token-to-token attention matrix. SpatialAnything uses Gated DeltaNet, which updates the state and computes the output as follows:

$$S_t = \alpha_t \left(\mathbf{I} - \beta_t k_t k_t^\top\right) S_{t-1} + \beta_t k_t v_t^\top, \qquad o_t = q_t^\top S_t. \tag{2}$$

Here, $S_t \in \mathbb{R}^{d_k \times d_v}$ is the recurrent state, $\mathbf{I}$ is the identity matrix, and q_t, k_t, and v_t are the query, key, and value at position t. The gate α_t controls the retention of the previous state, while β_t controls information correction along the current key direction. The query q_t computes the output but does not participate in the state update. For fixed state dimensions, the cost of this recurrence grows linearly with sequence length.

The two attention mechanisms provide different forms of information access. Full attention uses a mask to determine which tokens are directly accessible, whereas gated linear attention propagates information through recurrent states. Under causal computation, S_t contains information only up to position t. Modifying the full-attention mask therefore does not automatically give gated linear attention access to subsequent positions. This distinction motivates PBA.

3.2 How Can Bidirectional Interaction Improve Spatial Prediction?

Coordinates within a visual grounding primitive jointly describe one spatial output, but causal computation restricts their interaction. We introduce Primitive Bidirectional Attention (PBA) to allow each coordinate prediction to use information from both preceding and subsequent positions within the primitive. The current coordinates are replaced with mask tokens carrying distinct positional encodings. Each position can access the preceding context and all masked positions in the current primitive, without accessing its ground-truth coordinates or subsequent response tokens.

Primitive interaction in full attention. PBA preserves the original QKV projections and modifies the attention mask to enable bidirectional visibility within the current primitive while retaining access to the preceding context. Each coordinate prediction can thus directly interact with the other positions in the same primitive. Positions outside the primitive retain causal masking, preserving causal order across primitives and during language generation.

Primitive-aware bidirectional linear attention. Gated linear attention propagates information through recurrent states, so changing a mask alone does not alter its information access. We preserve the original state recurrence and transform queries while incorporating contributions from subsequent positions, allowing each coordinate to use the state obtained after processing the complete primitive.

Specifically, let the coordinate positions identified by the control tags of primitive σ be $s_\sigma, \ldots, e_\sigma$. Standard computation produces the output at position t as $q_t^\top S_t$, whereas PBA targets $q_t^\top S_{e_\sigma}$. All coordinates therefore use the same primitive-end state while retaining their own queries. To reuse the recurrence introduced in Sec. 3.1, we expand this target as:

$$\widetilde{o}_t = \widetilde{q}_{\sigma,t}^\top \left[\alpha_t \left(\mathbf{I} - \beta_t k_t k_t^\top\right) S_{t-1} + \beta_t k_t v_t^\top\right] + \sum_{j=t+1}^{e_\sigma} \omega_{\sigma,t,j} v_j^\top. \tag{3}$$

The first term computes an output from the current state using a transformed query, and the second incorporates contributions from subsequent positions within the primitive. The transformed query and corresponding coefficients are defined as:

$$\widetilde{q}_{\sigma,t} = T_{\sigma,t}^\top q_t, \qquad \omega_{\sigma,t,j} = \beta_j q_t^\top T_{\sigma,j} k_j, \qquad T_{\sigma,j} = A_{e_\sigma} \cdots A_{j+1}, \quad A_j = \alpha_j \left(\mathbf{I} - \beta_j k_j k_j^\top\right), \tag{4}$$

where $T_{\sigma,j}$ represents the state transformation from position j to the end of the primitive, with $T_{\sigma,e_\sigma} = \mathbf{I}$.

The query transformation accounts for how subsequent recurrence steps affect existing information, while the summation incorporates new information introduced at subsequent positions. Together, they exactly recover $\widetilde{o}_t = q_t^\top S_{e_\sigma}$. For a 2D box, the first coordinate prediction can therefore use information from the other three masked positions, while the four coordinates retain distinct queries and outputs.

Because queries do not participate in state updates, the transformed queries can be passed directly to the original computation kernel, after which the contributions from subsequent positions are added to the corresponding outputs. For fixed layer inputs, this procedure preserves the original keys, values, gates, and state recurrence. It introduces no additional learnable parameters and requires no additional reverse state scan.

3.3 How Can Parallel Primitive Prediction Accelerate Generation?

Building on PBA, we adopt adaptive parallel decoding to predict all coordinates of a complete primitive simultaneously. Language and control tags remain autoregressive, while coordinates within a primitive form a parallel prediction group determined by its type and specified length. The prediction group varies with the primitive, allowing boxes, keypoints, and trajectories of different lengths to follow a common generation procedure.

Primitive-level parallel prediction. After generating the opening tag, the decoder inserts mask tokens with distinct positional encodings for all coordinates to be predicted. A 2D box requires four prediction positions, while a trajectory containing N 2D waypoints requires $2N$ positions. The waypoint count is supplied by the task specification, without requiring a fixed block width or additional padding. These positions access the preceding context and interact bidirectionally through PBA. Let their outputs be $H_\sigma = [h_{\sigma,1}, \ldots, h_{\sigma,m_\sigma}]^\top$, where m_σ is the number of coordinates in the current primitive. A shared output head computes the logits for all coordinates in a single forward pass as $Z_\sigma = H_\sigma W_{\text{out}} + b_{\text{out}}$, where each row of Z_σ corresponds to one coordinate prediction. Unlike token-by-token prediction, later coordinates do not wait for earlier coordinate values to be selected. All coordinates of the primitive are predicted in the same forward pass.

Coordinate vocabulary constraints. To prevent coordinate positions from producing text or unrelated control tags, the decoder restricts each position to its valid coordinate token set $\mathcal{V}_{\sigma,i}$. The prediction distribution at position i is:

$$p_{\sigma,i} = \text{softmax}\left(z_{\sigma,i} + a_{\sigma,i}\right), \qquad a_{\sigma,i}(v) = \begin{cases} 0, & v \in \mathcal{V}_{\sigma,i}, \\ -\infty, & \text{otherwise}. \end{cases} \tag{5}$$

The decoder simultaneously selects the highest-probability valid token at each position to obtain the complete coordinate sequence. This constraint ensures valid token types, while spatial accuracy depends on the model predictions. Control tags are handled separately from the coordinate sequence.

Continuation after primitive prediction. The temporary cache produced by parallel prediction corresponds to masked inputs, whereas subsequent generation must use the selected coordinate values. We therefore adopt a prediction-and-causal-replay procedure. Before prediction, the decoder saves the existing context cache. After selecting the coordinates, it discards the temporary cache from the masked branch, restores the original cache, and processes the selected coordinates causally to update the full-attention cache, gated linear-attention states, and convolution history. Replay uses already selected coordinates without predicting them again. The decoder then processes the closing tag and continues generating language or the next primitive.

3.4 How Can MOPD Be Adapted to Strengthen Spatial Capabilities?

Training pipeline. Spatial tasks share foundational capabilities in localization and spatial relation understanding, but differ in output formats and optimization objectives. We first combine egocentric interaction, navigation, visual grounding primitive, and spatial reasoning question-answering data for supervised fine-tuning (SFT), using a unified language-and-primitive representation. The resulting model provides a common initialization for the student and task-specific teachers. Starting from the supervised model, we use Group Relative Policy Optimization (GRPO) to train specialized teachers for the four task families. We then apply a two-stage Spatial Multi-Teacher On-Policy Distillation (Spatial MOPD) strategy. A visual grounding primitive teacher first establishes a localization foundation, after which task-specific teachers guide student responses on their respective tasks to integrate complementary spatial capabilities into a single model. Figure 3 illustrates the two-stage training pipeline.

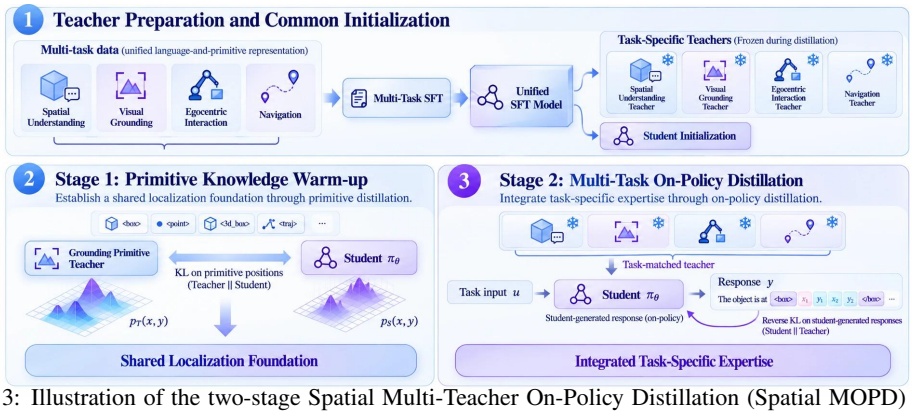


Figure 3: Illustration of the two-stage Spatial Multi-Teacher On-Policy Distillation (Spatial MOPD) training strategy. Supervised fine-tuning provides a common initialization for the student and task-specific teachers. In Stage 1, a frozen visual grounding primitive teacher transfers shared coordinate prediction knowledge. In Stage 2, task-specific teachers trained with GRPO provide on-policy distillation guidance on student-generated responses.

Stage 1: Visual grounding primitive knowledge warm-up. We use a frozen visual grounding primitive teacher π_{prim} to strengthen coordinate prediction for boxes, keypoints, and trajectories. The teacher and student receive identical masked primitive inputs, and shared grounding knowledge is transferred by matching their coordinate prediction distributions:

$$\mathcal{L}_{\text{prim}} = \mathbb{E}_{u\sim\mathcal{D}_{\text{prim}}}\left[\sum_{t\in\mathcal{G}(u)} D_{\text{KL}}\left(\pi_{\text{prim}}(\cdot \mid c_t) \,\|\, \pi_\theta(\cdot \mid c_t)\right)\right]. \tag{6}$$

Here, $\mathcal{D}_{\text{prim}}$ is the primitive supervision dataset, and $\mathcal{G}(u)$ contains the coordinate prediction positions in each sample. The context c_t includes visual inputs, language instructions, preceding context, and the masked inputs of the current primitive, excluding its ground-truth coordinates. Through primitive supervision across tasks, this stage establishes a shared localization foundation for subsequent joint training.

Stage 2: Multi-task on-policy distillation. Building on the student from the first stage, we introduce teachers for all four task families. The student generates responses from a mixture of task data, and the corresponding frozen teacher π_{ϕ_f} provides distributional supervision under the same context. All tasks update the same student through reverse KL minimization to integrate specialized expertise:

$$\mathcal{L}_{\text{Spatial-MOPD}} = \sum_f \rho_f \mathbb{E}_{\substack{u\sim\mathcal{D}_f \\ y\sim\pi_\theta(\cdot|u)}}\left[\sum_{t\in\mathcal{A}(y)} D_{\text{KL}}\left(\pi_\theta(\cdot \mid c_t) \,\|\, \pi_{\phi_f}(\cdot \mid c_t)\right)\right]. \tag{7}$$

Here, f indexes tasks, ρ_f is the task sampling proportion, and $\mathcal{A}(y)$ contains the actual prediction positions for language, control tags, and coordinates, excluding causal replay operations. Coordinate distillation uses identical masked prediction contexts for the teacher and student, while subsequent language distillation uses causal contexts containing the selected coordinates. The two stages first establish shared localization knowledge and then jointly develop task-specific capabilities, without requiring teachers during inference.

4 Experiments

4.1 Implementation Details

We evaluate SpatialAnything (SA) at three model scales: 4B, 9B, and 27B. For the 4B model, we initialize from a grounding-adapted Qwen3.5-4B checkpoint and jointly fine-tune the language model, visual encoder, and multimodal aligner using Megatron. Supervised training runs for one epoch with a global batch size of 32, a learning rate of 1×10^{-5} decayed to 1×10^{-6}, and a warm-up fraction of 5%. We enable sequence packing with a maximum sequence length of 16,384 and allow up to 1,024 visual tokens per image. Remaining distributed-training settings are given in the appendix. All benchmarks follow their specified evaluation protocols, and generation efficiency is

Table 1: Visual grounding primitive prediction results. SA denotes SpatialAnything. ↑ indicates higher is better. **Bold** marks the best result in each column and dashed underline marks the runner-up. External point-grounding results follow the micro-F1 evaluation reported in PhysBrain 1.5 (DeepCybo Team, 2026), and RynnBrain 1.1 3D detection results are taken from its original paper (Li et al., 2026).

	Point					2D Box	3D Box			
Model Metric	PixMo-Points micro-F1 ↑	RefSpatial-Bench micro-F1 ↑	RoboAfford micro-F1 ↑	VABench-Point micro-F1 ↑	Where2Place micro-F1 ↑	RefCOCO Acc.@0.5 ↑	ScanRefer Acc.@0.25 ↑	Multi3DRef F1@0.25 ↑	SUN RGB-D AP@15 ↑	WildDet3D-Bench AP_{3D} ↑
RynnBrain 1.1 9B	29.9	**63.2**	74.0	15.8	72.7	88.6	44.6	36.6	41.1	23.4
Hy-Emb.-VLM-1.0 30A3B (thinking)	55.8	47.4	75.8	60.5	65.4	79.3	41.7	28.5	36.7	21.4
MiMo Embodied 7B (thinking)	51.5	40.0	71.8	47.7	63.9	71.7	39.4	22.1	33.5	20.6
PhysBrain 1.5 8B	62.2	50.9	**80.4**	**65.2**	72.1	72.6	43.9	33.6	37.3	25.4
SA-4B	50.2	49.8	55.8	14.6	68.5	88.3	42.8	36.4	34.7	22.9
SA-9B	63.3	56.0	59.5	17.3	72.1	90.1	46.5	40.7	38.6	26.1
SA-27B	**65.2**	59.0	64.8	20.5	**76.4**	**91.3**	**51.3**	**45.2**	**43.5**	**29.8**

Table 2: Spatial understanding and reasoning results on 10 benchmarks. SA denotes SpatialAnything. ↑ indicates higher is better. **Bold** marks the best result in each column and dashed underline marks the runner-up. The base model is listed for reference and excluded from the ranking.

Model	BLINK ↑	CV-Bench ↑	3DSRBench ↑	EmbSpatial-Bench ↑	MindCube ↑	MMSI-Bench ↑	RoboSpatial-Home ↑	SAT ↑	VSI-Bench ↑	ViewSpatial-Bench ↑	**Avg.** ↑
Closed-Source Models											
Gemini 3.6 Flash (minimal thinking)	**89.2**	90.0	**67.7**	**84.7**	77.1	52.8	71.0	88.7	51.8	56.6	73.0
GPT 6 Astra (low thinking)	80.6	84.9	62.3	79.8	78.8	**57.9**	73.7	**96.7**	59.8	54.2	72.9
Claude Opus 5 (adaptive, low thinking)	85.5	87.8	63.3	81.1	66.7	44.1	72.0	88.7	21.3	49.8	66.0
Open-Source Models											
Hy-Emb.-VLM-1.0 30A3B (thinking)	86.1	89.3	62.1	80.5	65.4	39.0	71.2	81.3	55.7	52.4	68.3
RynnBrain 1.1 9B	84.8	88.1	59.0	81.9	82.1	46.0	70.4	78.0	65.9	56.4	71.3
RoboBrain 2.5 8B	80.4	88.0	54.7	77.9	30.4	29.4	67.3	65.3	43.9	39.3	57.7
MiMo Embodied 7B (thinking)	69.5	89.4	53.2	77.0	31.3	31.8	60.1	76.0	45.0	39.7	57.3
Cosmos3 Nano 8B+8B	80.6	88.3	53.5	82.1	39.2	35.7	66.4	69.3	50.4	55.0	62.0
PhysBrain 1.5 8B	87.9	90.0	63.0	81.8	86.2	41.0	73.9	79.3	61.9	62.5	72.8
Base Model Reference											
Qwen3.5-4B	29.5	88.3	43.3	77.4	47.3	32.5	45.1	60.0	50.1	40.3	51.4
Ours											
SA-4B	83.3	96.2	59.3	82.9	90.6	47.6	73.7	82.7	69.3	58.6	74.4
SA-9B	85.5	96.1	62.1	82.2	92.4	45.7	76.4	84.0	70.5	59.3	75.4
SA-27B	88.2	**96.8**	66.0	83.4	**94.8**	50.2	**78.6**	85.3	**73.9**	**63.9**	**78.1**

measured using end-to-end latency and peak memory, including coordinate prediction and causal replay.

4.2 Visual Grounding Primitives

We evaluate SpatialAnything on point, 2D box, and 3D box prediction tasks, covering target and interaction-point localization, object-region localization in images, and object localization in three-dimensional space. As summarized in Table 1, performance improves with model size on all ten reported metrics: SA-27B reaches 91.3 on RefCOCO, 51.3 on ScanRefer, and 45.2 on Multi3DRef, against 88.3, 42.8, and 36.4 for SA-4B. Annotated examples of every primitive type and of the robot-manipulation sources are shown in Figures 6–12 and Figure 13 of Appendix E.

4.3 Spatial Understanding and Reasoning

We evaluate SpatialAnything on 10 spatial understanding and reasoning benchmarks covering visual-spatial perception, spatial relationships, viewpoint changes, and reasoning across observations, as summarized in Table 2, against proprietary models (OpenAI, 2026; Google DeepMind, 2026; Anthropic, 2026), open embodied models (Tencent Hunyuan Team, 2026; Li et al., 2026; Tan et al., 2026; Hao et al., 2025; NVIDIA, 2026; DeepCybo Team, 2026), and the Qwen3.5 base model (Qwen Team, 2026a). Multi-frame question examples of this family are shown in Figure 15 of Appendix E. SA-27B averages 78.1 across the ten benchmarks, exceeding GPT 6 Astra by 5.2 points, and achieves the highest reported scores on CV-Bench, MindCube, VSI-Bench, and ViewSpatial-Bench. Its 88.2 on BLINK is second only to Gemini 3.6 Flash. All three scales surpass the evaluated GPT 6 Astra configuration on average.

4.4 Egocentric Interaction and Navigation

Egocentric interaction and embodied reasoning. Table 3(a) reports EgoPlan-Bench2, ERQA, ERQA-PLUS, RoboVQA, VLABench, and VABench-V.-Trace, covering task planning, interaction reasoning, and visual trajectory prediction. SA-27B improves over SA-9B on all six benchmarks, with the largest gain of 7.7 percentage points on ERQA, and leads the compared models on EgoPlan-Bench2, RoboVQA, and VABench-V.-Trace. Gaps remain on ERQA and ERQA-PLUS relative to

Table 3: Results on egocentric interaction and navigation. SA denotes SpatialAnything. (a) Task planning, embodied reasoning, and trajectory prediction; external results follow PhysBrain 1.5 (DeepCybo Team, 2026), and VABench-V.-Trace reports $100 - \mathrm{NRMSE}$. (b) Navigation SR (%) on VLN-CE R2R, RxR, and HM3D-OVON. **Bold** marks the best and dashed underline the runner-up.

(a) Egocentric Interaction and Reasoning							(b) Navigation			
Model	EgoPlan-Bench2 ↑	ERQA ↑	ERQA-PLUS ↑	RoboVQA ↑	VLABench ↑	VABench-V.-Trace ↑	Model	R2R SR ↑	RxR SR ↑	OVON SR ↑
GPT 6 Astra	69.3	**75.8**	**86.1**	37.0	66.3	91.6	Qwen-RobotNav	58.4	45.7	29.6
Hy-Emb.-VLM-1.0 30A3B	47.5	56.3	82.2	41.2	50.9	87.5	Hy-Emb.-VLM-1.0 30A3B	48.1	51.0	33.9
RynnBrain 1.1 9B	37.0	46.5	83.6	60.9	42.4	86.6	RynnBrain 1.1	61.2	49.3	32.4
PhysBrain 1.5 8B	62.1	52.8	85.2	61.5	**76.4**	89.8	PhysBrain 1.5 8B	24.3	18.6	39.6
SA-4B	61.8	46.0	78.6	60.3	63.7	86.7	**SA-4B**	56.3	55.9	34.4
SA-9B	66.3	44.5	79.9	63.1	65.8	90.9	**SA-9B**	61.3	58.0	37.7
SA-27B	**69.4**	52.2	82.0	**66.7**	68.4	**92.6**	**SA-27B**	**62.5**	**61.2**	**42.1**

Table 4: Ablation studies on SA-4B. (a) Attention and decoding strategies. (b) Posttraining strategies. (c) Supervised data scale before posttraining. ScanRefer reports accuracy at IoU 0.25, RefCOCOg at IoU 0.5, RefSpatial denotes RefSpatial-Bench micro-F1, and Time measures end-to-end latency in milliseconds. **Bold** marks the best and dashed underline the runner-up.

(a) Attention and Decoding				(b) Posttraining				(c) Data Scaling			
Configuration	ScanRefer ↑	RefSpatial ↑	Time ↓	Strategy	RefCOCOg ↑	CV-Bench ↑	MindCube ↑	Data	RefCOCOg ↑	CV-Bench ↑	MindCube ↑
Causal parallel	40.8	37.7	320ms	SFT only	81.3	93.8	89.5	12.5%	72.6	88.3	60.7
Full-only PBA	47.9	44.0	290ms	Mix RL	78.5	91.7	77.2	25%	75.3	91.8	61.8
Linear-only PBA	48.6	45.8	210ms	MOPD	76.6	95.1	90.6	50%	72.8	92.6	78.1
Full PBA	**52.1**	**49.8**	**110ms**	**Spatial MOPD**	**88.3**	**96.2**	**91.6**	**100%**	**88.3**	**96.2**	**91.6**

GPT 6 Astra and on VLABench relative to PhysBrain 1.5. Egocentric training examples are shown in Figure 14 of Appendix E.

Navigation. We evaluate on VLN-CE R2R, VLN-CE RxR, and HM3D-OVON, using success rate (SR) to measure task completion. As shown in Table 3(b), SA-27B achieves 62.5%, 61.2%, and 42.1%, improving over SA-4B on all three benchmarks. Navigation examples with instructions and action sequences appear in Figure 16 of Appendix E. The observation inputs, model outputs, and downstream execution procedures are detailed in the appendix.

4.5 Ablation Studies

Attention and decoding strategies. Table 4(a) compares causal parallel prediction with three PBA variants applied to full-attention layers, gated linear-attention layers, or both. All variants use the same 4B initialization, supervised training data, and training schedule, without posttraining, and share masked coordinate inputs, parallel decoding, and coordinate vocabulary constraints, isolating the attention modifications. Full PBA attains the best accuracy on both 3D box localization (ScanRefer) and point localization (RefSpatial-Bench), while reducing generation time from 320 to 110 milliseconds relative to causal parallel prediction under fixed hardware, precision, and batch size.

Posttraining strategies. Table 4(b) compares the SFT checkpoint, full-data mixed-task GRPO (Mix RL), standard MOPD, and Spatial MOPD, all starting from the same supervised PBA checkpoint and, for the two distillation variants, sharing task teachers, sampling proportions, and rollout budget. Standard MOPD directly performs multi-task on-policy distillation, whereas Spatial MOPD first introduces visual grounding primitive knowledge warm-up. Spatial MOPD reaches 88.3, 96.2, and 91.6 on RefCOCOg, CV-Bench, and MindCube, the best result on each, while Mix RL falls below the SFT checkpoint on all three.

Training data scale. Table 4(c) evaluates SA-4B using 12.5%, 25%, 50%, and 100% of the supervised training corpus, without subsequent reinforcement learning or distillation. We construct nested subsets that preserve task-family proportions and keep the initialization, number of training epochs, and other training settings fixed. Performance increases with data scale on CV-Bench and MindCube, with MindCube improving from 60.7 to 91.6.

5 Conclusion

We present Primitive Bidirectional Attention (PBA), which combines structured masks in full-attention layers with query rewriting in gated linear-attention layers, enabling information exchange within each visual grounding primitive and adaptive parallel decoding per primitive. We instantiate PBA in SpatialAnything and train it with Spatial MOPD, improving localization accuracy and decoding efficiency across grounding, spatial reasoning, egocentric interaction, and navigation, which suggests that the primitive, rather than the individual coordinate token, is a useful unit for embodied spatial modeling.

Reproducibility Statement. Training and evaluation of SpatialAnything are specified in full: the supervised corpus composition and token accounting are reported in Appendix B, hyperparameters and the parallel decoding schedule in Appendix C, evaluation protocols for each benchmark family in Appendix D, and the Spatial MOPD objective derivations in Appendix G. All training sources are derived from publicly released datasets under their original licenses, and the evaluation benchmarks are publicly available. We will release the training and evaluation code, model checkpoints, and data conversion scripts upon publication.

Ethics Statement. This work trains and evaluates models on publicly released datasets and benchmarks under their original licenses, without collecting any new human or private data. The embodied spatial capabilities studied here are intended to support applications such as assistance, navigation, and human-robot interaction. As with any generative multimodal model, predictions may contain errors, and deployed systems should not rely on the model alone for safety-critical decisions. We report results under the stated evaluation settings and release model and code details to support scrutiny and reproduction.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations (ICLR)*, 2024.

Anthropic. Claude Opus 5 system card. `https://www.anthropic.com/claude-opus-5`, 2026.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-VL: A versatile vision-language model for understanding, localization, text reading, and beyond. *arXiv preprint*, 2023.

Anthony Brohan et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning (CoRL)*, 2023.

Boyuan Chen, Zhuo Xu, Sean Kirmani, Brian Ichter, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. SpatialVLM: Endowing vision-language models with spatial reasoning capabilities. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Junsong Chen, Jincheng Yu, Yitong Li, Tian Ye, Zhangjie Wu, Zian Wang, Shuchen Xue, Daquan Zhou, Haozhe Liu, Jingyu Xin, Yuyang Zhao, Ping Luo, Song Han, and Enze Xie. Sana-video 2.0: Hybrid linear attention with attention residuals for efficient video generation. *arXiv preprint arXiv:2607.21553*, 2026.

Keqin Chen, Zhao Zhang, Weili Zeng, Richong Zhang, Feng Zhu, and Rui Zhao. Shikra: Unleashing multimodal LLM's referential dialogue magic. *arXiv preprint*, 2023.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

DeepCybo Team. PhysBrain 1.5: From vision-language models to physical foundation models. *arXiv preprint arXiv:2609.14973*, 2026.

Google DeepMind. Gemini 3.6 Flash model card. `https://deepmind.google/models/model-cards/gemini-3-6-flash/`, 2026.

Xiaoshuai Hao et al. MiMo-Embodied: X-Embodied foundation model technical report. *arXiv preprint arXiv:2511.16518*, 2025.

Yining Hong, Haoyu Zhen, Peihao Chen, Shuhong Zheng, Yilun Du, Zhenfang Chen, and Chuang Gan. 3D-LLM: Injecting the 3d world into large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Jiangyong Huang, Silong Yong, Xiaojian Ma, Xiongkun Linghu, Puhao Li, Yan Wang, Qing Li, Song-Chun Zhu, Baoxiong Jia, and Siyuan Huang. An embodied generalist agent in 3d world. In *International Conference on Machine Learning (ICML)*, 2024.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. OpenVLA: An open-source vision-language-action model. In *Conference on Robot Learning (CoRL)*, 2024.

Xin Lai, Zhuotao Tian, Yukang Chen, Yanwei Li, Yuhui Yuan, Shu Liu, and Jiaya Jia. LISA: Reasoning segmentation via large language model. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International Conference on Machine Learning (ICML)*, 2023.

Kehan Li, Bohan Hou, Minghao Zhu, Tianyi Zhang, Zesen Cheng, Zhikai Wang, Sicong Leng, Xin Li, Xiao Lin, Biying Yao, Minghua Zeng, Jiangpin Liu, Ronghao Dang, and Jiayan Guo. Rynnbrain 1.1: Towards more capable and generalizable embodied foundation model. *arXiv preprint arXiv:2607.17977*, 2026.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

NVIDIA. Cosmos 3: Omnimodal world models for physical AI. *arXiv preprint arXiv:2606.02800*, 2026.

OpenAI. GPT-6 Astra system card. `https://deploymentsafety.openai.com/gpt-6-astra`, 2026.

Qwen Team. Qwen3.5-Omni technical report. *arXiv preprint arXiv:2604.15804*, 2026a.

Qwen Team. Qwen-RobotNav technical report: A scalable navigation model designed for an agentic navigation system. *arXiv preprint arXiv:2606.18112*, 2026b.

Hanoona Rasheed, Muhammad Maaz, Sahal Shaji, Abdelrahman Shaker, Salman Khan, Hisham Cholakkal, Rao Muhammad Anwer, Eric Xing, Ming-Hsuan Yang, and Fahad Shahbaz Khan. GLaMM: Pixel grounding large multimodal model. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

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

Huajie Tan, Enshen Zhou, Zhiyu Li, Yijie Xu, Yuheng Ji, Xiansheng Chen, Cheng Chi, Pengwei Wang, Tiejun Huang, Shanghang Zhang, et al. Robobrain 2.5: Depth in sight, time in mind. *arXiv preprint arXiv:2601.14352*, 2026.

Tencent Hunyuan Team. Hy-Embodied-VLM-1.0: Efficient physical-world agents. *arXiv preprint arXiv:2607.12894*, 2026.

Shihao Wang, Shilong Liu, Yuanguo Kuang, Xinyu Wei, Yangzhou Liu, Zhiqi Li, Yunze Man, Guo Chen, Andrew Tao, Guilin Liu, Jan Kautz, Lei Zhang, and Zhiding Yu. Locateanything: Fast and high-quality vision-language grounding with parallel box decoding. *arXiv preprint arXiv:2605.27365*, 2026.

Enze Xie, Junsong Chen, Junyu Chen, Han Cai, Haotian Tang, Yujun Lin, Zhekai Zhang, Muyang Li, Ligeng Zhu, Yao Lu, and Song Han. SANA: Efficient high-resolution image synthesis with linear diffusion transformers. *arXiv preprint*, 2024.

Shiyu Xuan, Qingpei Guo, Ming Yang, and Shiliang Zhang. Pink: Unveiling the power of referential comprehension for multi-modal LLMs. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

An Yang et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *International Conference on Learning Representations (ICLR)*, 2025b.

Haoxuan You, Haotian Zhang, Zhe Gan, Xianzhi Du, Bowen Zhang, Zirui Wang, Liangliang Cao, Shih-Fu Chang, and Yinfei Yang. Ferret: Refer and ground anything anywhere at any granularity. In *International Conference on Learning Representations (ICLR)*, 2024.

A THE USE OF LARGE LANGUAGE MODELS (LLMS)

We acknowledge the use of large language models as writing assistants only for grammar and style. LLMs are not employed in the core research methodology, experimental design, data analysis, or generation of research findings presented in this paper. All textual content has been reviewed and verified by the authors.

B SUPERVISED TRAINING CORPUS

B.1 DESIGN PRINCIPLES

The supervised corpus covers the four task families used throughout this paper: visual grounding primitives, spatial reasoning question answering, egocentric interaction, and navigation. All families are stored in one unified language-and-primitive representation. Coordinate answers are wrapped in paired control tags, for example <bbox2d>$c_1c_2c_3c_4$</bbox2d> for a 2D box and <point>c_1c_2</point> for a keypoint, where each c_i is a single coordinate token from the extended vocabulary of Sec. C.1. The opening tag fixes the primitive type, and the length follows from the type and the task specification, so the span boundary of every primitive is known before its coordinates are generated. Prose coordinates that appear inside reasoning text or inside the user instruction, such as waypoints quoted from the question, are kept as plain text; only assistant-side structured outputs use containers. This distinction keeps the supervised objective aligned with the decoding procedure of Sec. 3.3.

B.2 CORPUS COMPOSITION

Table 5 and Figure 4 summarize the corpus: 75 source sets, 9.31M samples, and 13.95B tokens, grouped into the four training task families of Sec. 3.4; annotated examples of every family appear in Sec. E. Families denote the intended training capability rather than mutually exclusive properties of the original datasets; each converted training example is assigned to exactly one family. Token counts cover the full packed sample, including instruction, response, and visual tokens, counted with the extended coordinate vocabulary. Samples whose packed length exceeds 16,384 tokens are discarded; the 7,466 discarded samples account for 0.08% of the corpus. Very large detection sources are subsampled for budget reasons: PixMo-Points uses a fixed 35% subset and OpenImages V6 a fixed 5% subset, while all other listed sources are used in full. Counts are rounded independently, so displayed entries may not sum exactly to subtotals.

B.3 CONSTRUCTION DETAILS

Grounding primitives. Detection and referring sources follow a common conversion. Box coordinates are normalized to the integer range $[0, 1000]$ on both axes and emitted as four coordinate tokens inside a <bbox2d> container. 3D boxes use the nine-parameter form $(x, y, z, w, h, l, r_1, r_2, r_3)$ with per-dimension normalization, inside a <bbox3d> container. Grasp poses adopt the three-parameter form (x, y, θ), where (x, y) is the contact point and θ is the gripper orientation in degrees mapped linearly to $[0, 1000]$. Trajectories store an explicit waypoint count followed by $2N$ coordinate tokens. Multi-frame 3D sources are reframed to at most 12 frames per sample: frame references quoted by the question are always preserved, and remaining slots are filled by uniform sampling. This recovers long multi-frame samples that would otherwise exceed the packing length.

Grounded chain-of-thought. Reasoning traces are constructed in two complementary ways. For a subset of questions, an LLM drafts numbered step-by-step reasoning that weaves perception containers into the reasoning spans, for example locating an object with a <bbox2d> or tracing a partial path with a <trajectory> before stating the conclusion; the final answer coordinates

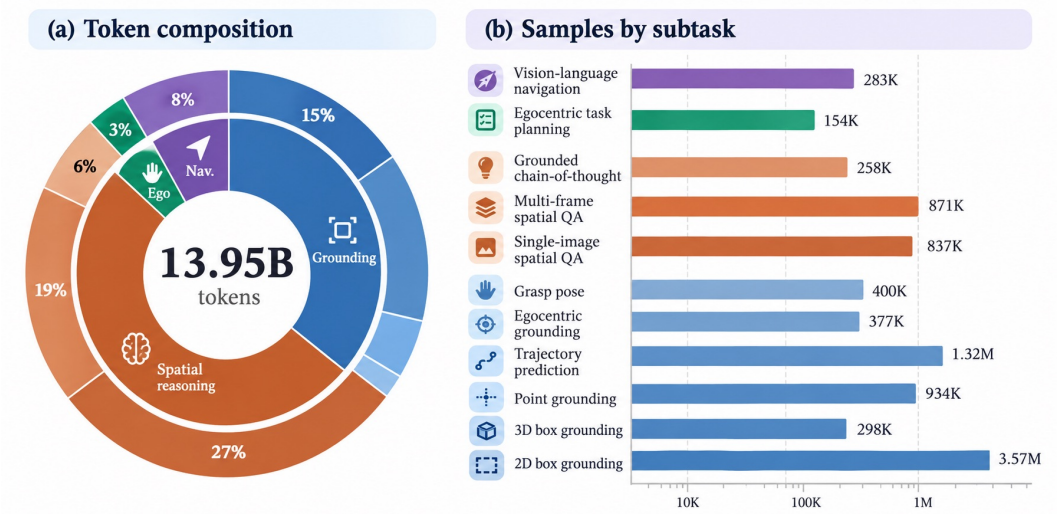
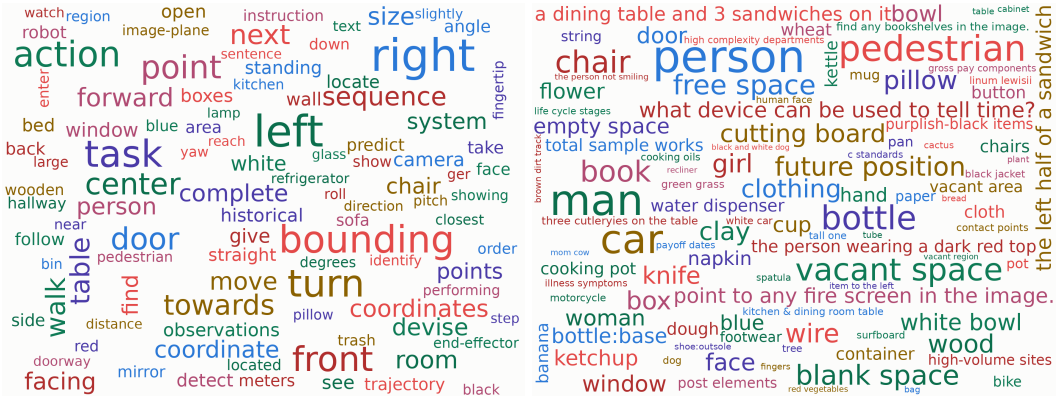


Figure 4: Composition of the supervised training corpus. Left: token shares of the four task families (inner ring) and the eleven subtasks (outer ring, shaded within each family color). Right: number of samples per subtask on a logarithmic scale; bar colors match the corresponding outer-ring segments.



(a) Question keywords (b) Answer object labels

Figure 5: Word clouds of the supervised corpus. (a) Question keywords sampled from the user side of every source set; formatting boilerplate and stop words are removed. (b) Object labels appearing in assistant answers; each source contributes its top-15 labels, normalized to that source's most frequent label, so detection-scale sources do not dominate the aggregate. In both panels, word area is proportional to frequency.

never appear in the reasoning text, which prevents trivial copying, and each draft is verified against the annotated answer before it enters the corpus. For the remaining questions, CoT is generated per-sample with visual grounding and answer checking, following the same container discipline.

Navigation. Vision-language navigation data pairs egocentric observation sequences with instruction-following dialogues. Each sample carries the instruction, a sequence of sampled frames, and the model-side response that references the observation history; no executable actions are required during supervised training.

Quality control. Every conversion is audited automatically. The audit checks container pairing, that the inner layer of every container consists solely of coordinate tokens, that no sample crosses a packing boundary mid-primitive, and that instruction-side prefill templates match the answer format. Projections used to derive 3D supervision from motion-capture or annotated 2D data are verified by enrichment tests, for example checking that projected contact points concentrate in the annotated interaction intervals rather than uniformly over time.

Table 5: Supervised training corpus by training task. Each row lists the source sets converted for that subtask; slash-separated entries count individually (RefCOCO/+/g denotes three sets). Samples and tokens reflect the retained corpus after source subsampling and length filtering; token counts include visual tokens.

Subtask	Sources	Samples	Tokens
Visual grounding primitives			*4.95B*
2D box grounding	COCO, V3Det, OpenImages V6†, PACO, EgoObjects, BDD100K, nuImages, MOT17/20Det, OWDOD, RefCOCO/+/g, gRefCOCO, RoboAfford, HumanParts, HumanRef (17 sets)	3.57M	2.15B
Point grounding	PixMo-Points†, CoSyn-Point (2 sets)	0.93M	0.76B
3D box grounding	ScanNet det/desc/temporal, WildDet3D (4 sets)	0.30M	1.47B
Trajectory prediction	MolmoAct-Trace (1 set)	1.32M	0.26B
Grasp pose	Grasp-Anything++ (1 set)	0.40M	0.12B
Egocentric grounding	FSD-Robot spatial/traj, FSD-COCO, RoboPoint, ShareRobot affordance/trajectory, RefSpatial (7 sets)	0.38M	0.19B
Spatial reasoning QA			*7.43B*
Single-image spatial QA	VSI, MindCube, ScanNet-QA, EmbSpatial, LLaVA-Hound, SenseNova-SI, Spatial-MLLM, ViCA (20 sets)	0.84M	3.83B
Multi-frame spatial QA	SenseNova-12f, SPAR-12f (11 sets)	0.87M	2.70B
Grounded chain-of-thought	Mixed-150k, Norm-200k, SenseNova-CoT, Nav-CoT, ViCA-CoT (5 sets)	0.26M	0.90B
Egocentric interaction			*0.41B*
Egocentric task planning	ShareRobot-Planning (1 set)	0.15M	0.41B
Navigation			*1.18B*
Vision-language navigation	StreamVLN-92K, DAgger-v2-RxR, DAgger-iter2/3, ScaleVLN, EnvDrop (6 sets)	0.28M	1.18B
Total	75 source sets	**9.31M**	**13.95B**

†Fixed sampled subsets: PixMo-Points 35%, OpenImages V6 5%.

B.4 Data Scaling Subsets

The data-scaling ablation of Table 4(c) uses nested subsets of the corpus that preserve the task-family proportions of Table 5. Token counts of the 12.5%, 25%, 50%, and 100% subsets are listed in Table 6.

Table 6: Processed token counts of the nested data-scaling subsets.

Subset	12.5%	25%	50%	100%
Tokens	2.6B	6.5B	10.3B	13.95B

C Training and System Configuration

C.1 Coordinate Vocabulary Extension

All models extend the base tokenizer with a contiguous block of 1,001 coordinate tokens <0>–<1000>, one token for each integer in the normalized coordinate range, plus a mask token used by PBA and the container tags of Sec. F.1. The vocabulary therefore grows from 248,077 to 249,094 entries, and the grasp container tags are added on top. New embeddings are initialized from the mean and covariance of the existing embedding table, and the identifiers of the coordinate tokens

are aligned across the 4B, 9B, and 27B checkpoints so that containers tokenize identically at every scale.

C.2 Supervised Fine-Tuning Hyperparameters

Table 7 lists the supervised fine-tuning configuration. All scales share the same optimizer, schedule, and packing settings and differ only in parallel topology and global batch size. Training runs for one epoch over the corpus with sequence packing at a maximum length of 16,384; the global batch size counts packed sequences. Samples whose packed length exceeds the limit are discarded (0.08% of the corpus). Each image contributes at most 1,024 visual tokens.

Table 7: Supervised fine-tuning configuration.

Item	Setting
Initialization	grounding-adapted Qwen3.5 (4B / 9B / 27B)
Learning rate	1×10^{-5}, cosine decay to 1×10^{-6}
Warmup fraction	5%
Epochs	1
Sequence packing	length 16,384, padding-free bin packing
Optimizer	Adam, weight decay 0.1, gradient clipping 1.0
Precision	bF16, distributed optimizer
Visual tokens per image	up to 1,024
Random seed	42

Table 8: Parallel configuration and training duration of the three scales. The global batch size counts packed sequences and equals micro-batch $\times$ data-parallel degree $\times$ gradient-accumulation steps.

	SA-4B	SA-9B	SA-27B
Nodes	2	4	8
Total GPUs	16	32	64
Tensor / pipeline parallel	2 / 2	1 / 1	2 / 1
Data-parallel degree	4	32	32
Gradient-accumulation steps	8	1	2
Sequence parallel	✓	–	✓
Global batch size	32	32	64
Packed sequences	747k	721k	721k
Training steps	23,347	22,522	11,261
Wall-clock time	27.5 h	27.8 h	36.0 h

C.3 Spatial MOPD Configuration

The precise objectives, the teacher-training procedure, and the analysis of the two KL directions are given in Sec. G. This subsection lists the numeric configuration. Stage 2 samples tasks from the four families with proportions ρ_f and distills the corresponding frozen teacher on student-generated responses.

D Evaluation Protocols

D.1 Benchmark Suite and Protocol Alignment

The evaluation suite covers 25 benchmarks across the four task families. For each benchmark we follow its official task formulation, visual-input configuration, and aggregation rule. Multiple-choice benchmarks report top-1 accuracy under a shared answer-only instruction; circular-evaluation benchmarks require all rotated variants of one question to be answered correctly. External baseline numbers follow the evaluation reported in the corresponding original papers: point-grounding micro-F1 scores follow PhysBrain 1.5, 3D detection results on ScanRefer, Multi3DRef, SUN RGB-D, and WildDet3D-Bench follow RynnBrain 1.1 and PhysBrain 1.5, and egocentric results follow PhysBrain 1.5.

D.2 Metric Definitions

Point grounding. Following the unified point-localization protocol, a predicted point is valid if it falls inside at least one ground-truth target region. Predicted points and target regions are matched one to one, and micro-averaged precision P and recall R are computed over the full benchmark. The reported score is the harmonic mean $F_1 = 2PR/(P + R)$. An explicit no-object prediction is treated as correct when no valid target exists.

Box grounding. RefCOCO reports accuracy at IoU 0.5 between the predicted and ground-truth boxes. ScanRefer reports accuracy at IoU 0.25 in 3D, and Multi3DRef reports F1 at IoU 0.25. For 3D detection, SUN RGB-D reports AP at the IoU 0.15 operating point and WildDet3D-Bench reports $\mathrm{AP}_{\mathrm{3D}}$.

Trajectory prediction. VABench-Visual-Trace reports $100 - \mathrm{NRMSE}$, where the normalized root-mean-square error is computed between the predicted and ground-truth waypoint sequences and normalized per benchmark.

Navigation. Vision-language navigation reports success rate (SR): the fraction of episodes in which the agent stops within the distance threshold of the goal. HM3D-OVON additionally requires the agent to reach a goal object of the specified open-vocabulary category.

Efficiency. Generation efficiency is measured as end-to-end wall-clock latency and peak GPU memory for one full response, including coordinate prediction and causal replay, under fixed hardware, precision, and batch size.

D.3 Navigation Evaluation Details

D.4 Inference Settings

E Additional Qualitative Results

This section presents annotated training examples for each grounding primitive type, with ground-truth primitives overlaid on the input images. Point coordinates are drawn as circles, 2D boxes as rectangles, 3D boxes as wireframes projected with the per-source camera intrinsics, trajectories as polylines with start (green) and end (red) markers, and grasp poses as gripper-direction segments through the contact point. The caption of each example shows the corresponding instruction.

F Primitive Bidirectional Attention: Derivations and Implementation

F.1 Structured Block Types

A structured block is a maximal run of coordinate tokens wrapped in a pair of control tokens. The opening control token fixes the block length, so the span boundary is known before the coordinates are generated. Table 9 lists the block types used in this work. A contour uses a variable length $2N$, and a trajectory is a variable length run of point coordinates.

Table 9: Structured block types recognized by the trigger scan.

Control token	Block type	Length
`<box>`	two dimensional box	4
`<box3d>`	three dimensional box	9
`<pt>`	point	2
`<seg>`	contour	$2N$

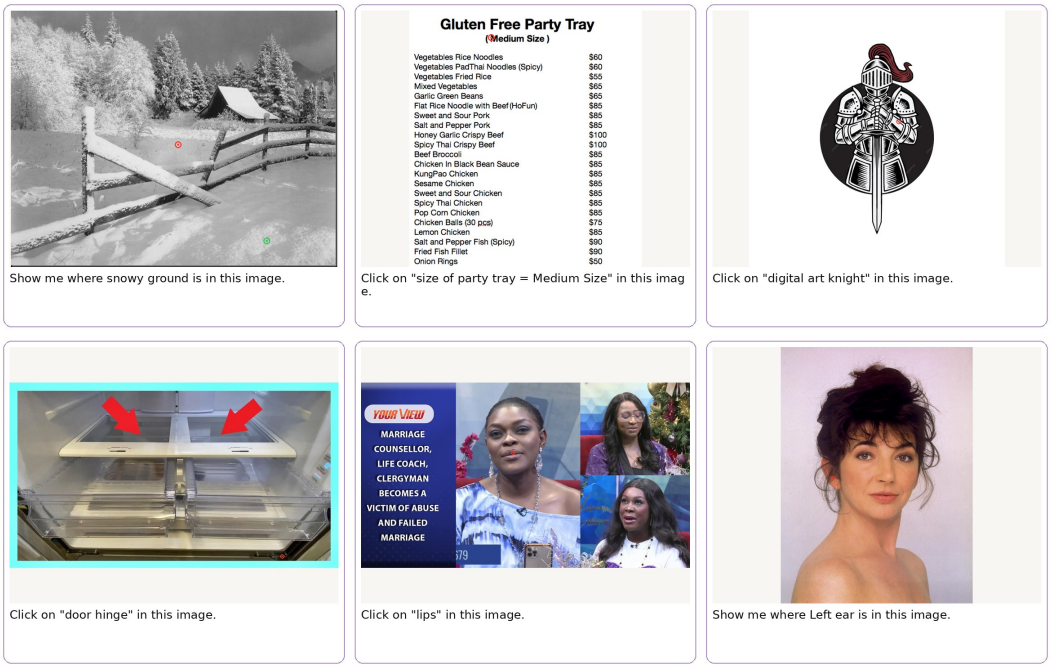


Figure 6: Point grounding training examples from PixMo-Points.

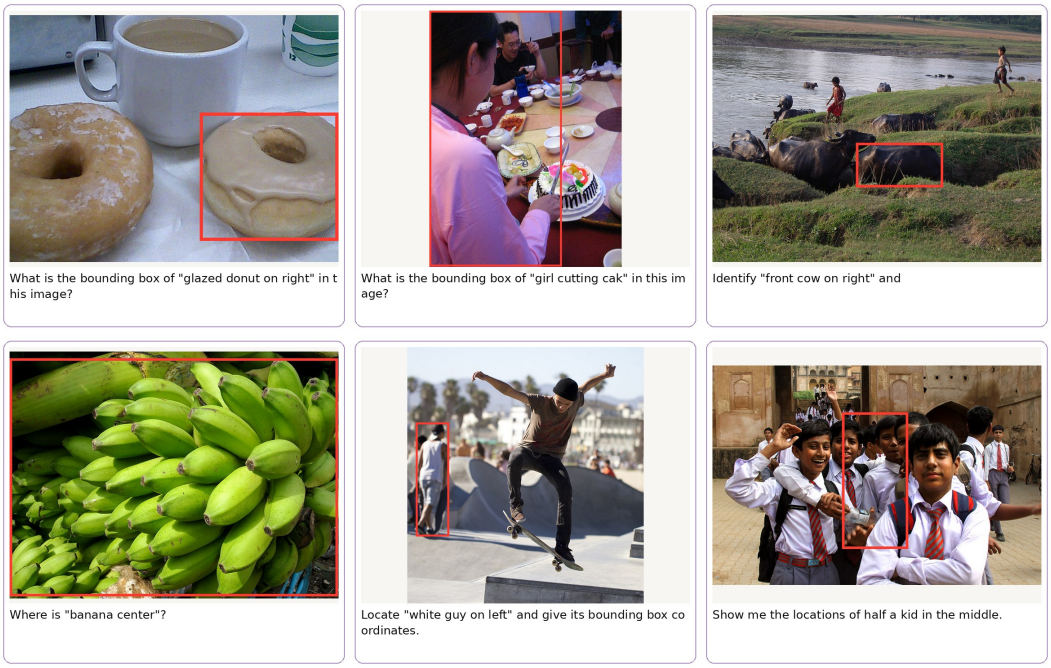


Figure 7: Referring 2D box grounding training examples from RefCOCO.

F.2 Why Mask Editing Does Not Reach a Hybrid Backbone

Throughout this appendix we refer to the positions occupied by one visual grounding primitive as a *span*, since the implementation operates on rows of the packed sequence rather than on grounding primitives.

Structured decoding obtains primitive internal visibility by editing the attention mask, so that the coordinate tokens of one block see each other. Editing a mask presumes that an attention matrix exists to be edited, which holds when every layer is softmax attention. Backbones for long sequences no longer have that shape. To raise throughput on long observation streams, recent models interleave

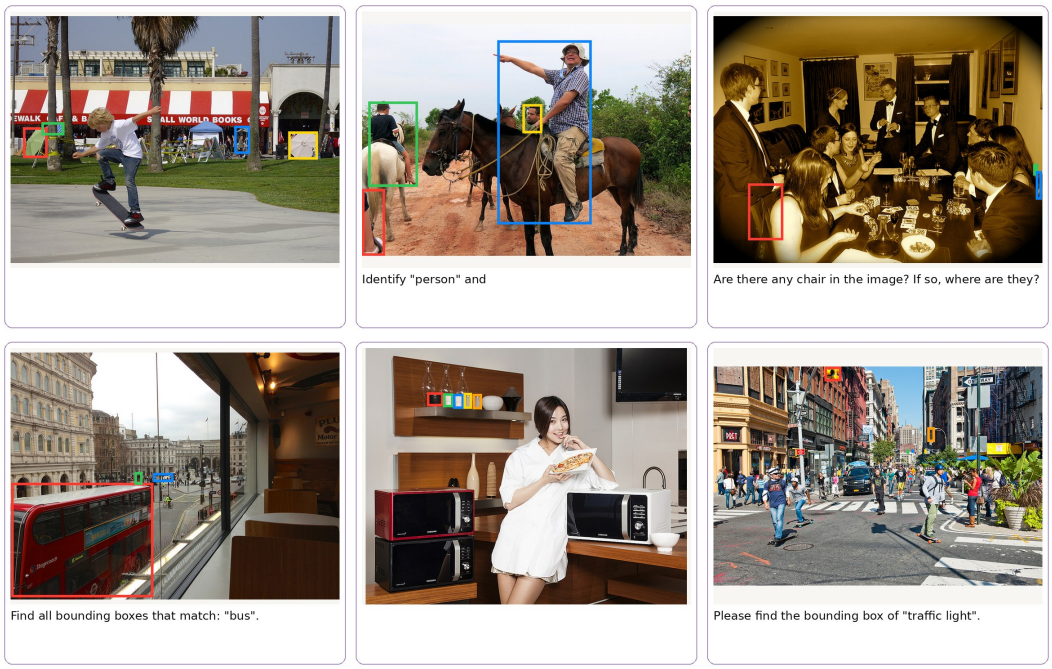


Figure 8: Multi-object 2D box grounding training examples from COCO.

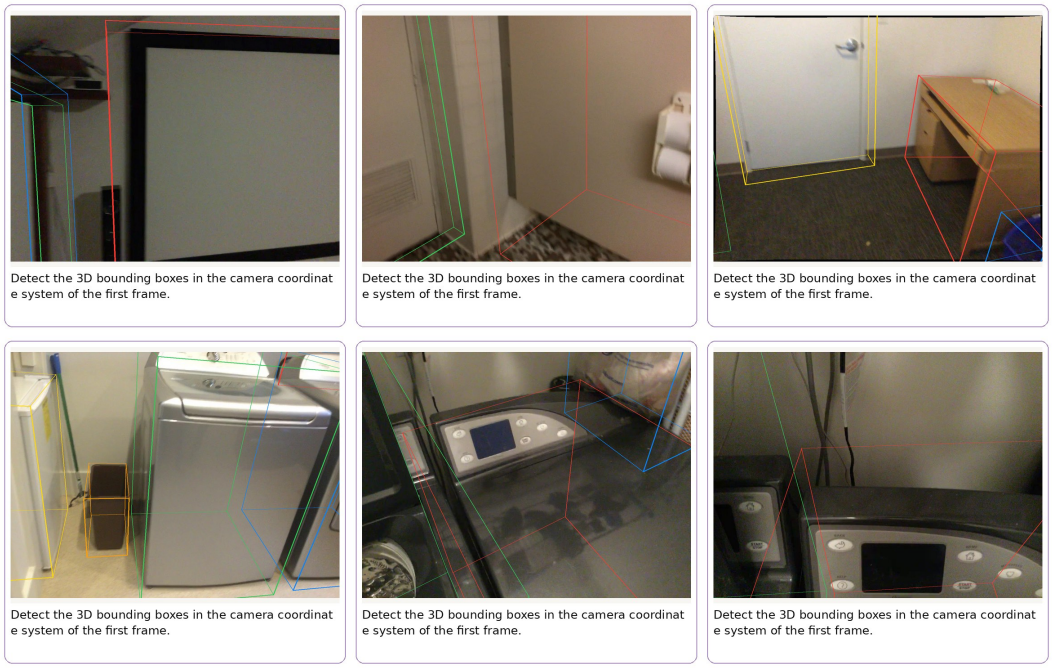


Figure 9: 3D box grounding training examples from ScanNet. Wireframes are projected onto the first frame of each sample using the scene intrinsics.

three linear attention layers with one full attention layer (Yang et al., 2025b;a), and video generators adopt the same three to one ratio (Chen et al., 2026). A linear attention layer carries a recurrent state instead of an attention matrix, so there is no mask to edit and the operation is silently a no-op. The grounding constraint then holds in one layer out of four and lapses in the rest, and the gap widens as a backbone leans further on linear attention.

Diffusion language models do not face this problem, because their attention is bidirectional to begin with and primitive internal visibility is free. An autoregressive model cannot take the same route, since making attention bidirectional everywhere would destroy the causal factorization it is trained

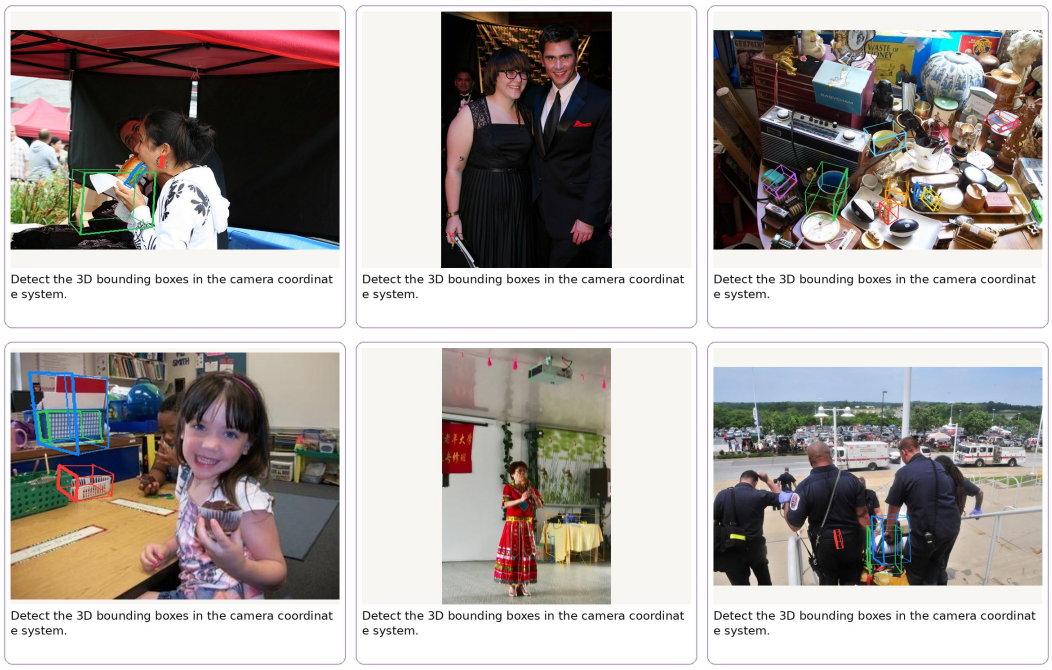


Figure 10: Open-vocabulary 3D box grounding training examples from WildDet3D, spanning indoor, outdoor, and driving scenes.

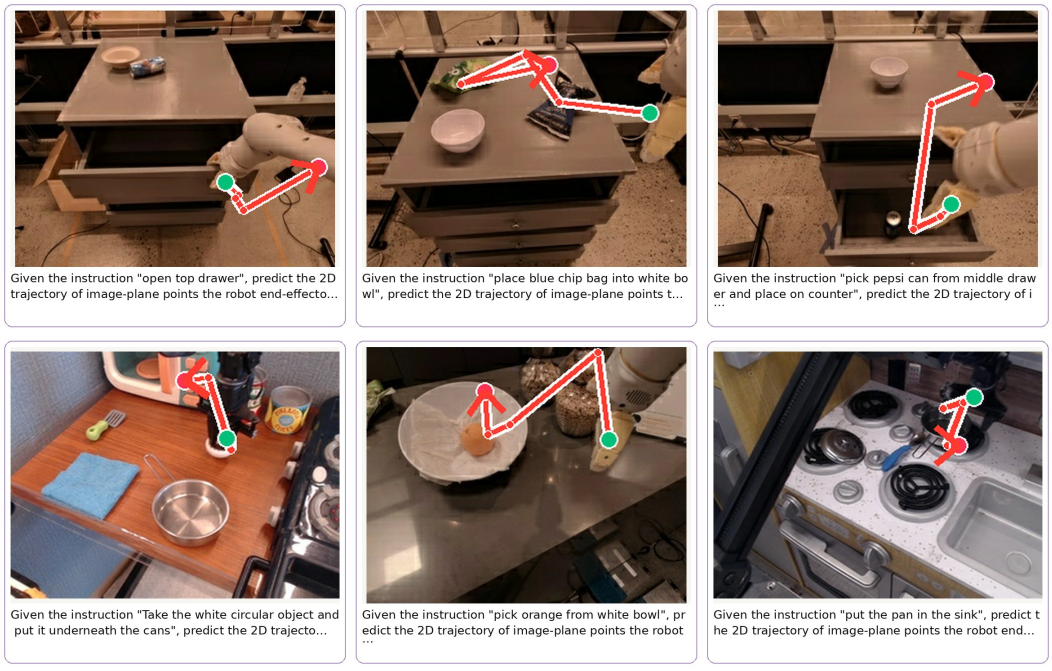


Figure 11: Trajectory prediction training examples from MolmoAct-Trace. Green and red markers denote the trajectory start and end.

under. The constraint has to be recovered inside the linear recurrence itself, which is what the query rewrite of Sec. 3.2 does.

Two further consequences follow for the layers that do carry the constraint. The layers that enforce it and the layers that do not must still agree on where one primitive ends and the next begins, otherwise the same constraint lands on different positions in different layers. The shared boundary list removes this depth direction misalignment, since every layer cuts the sequence from the same source.

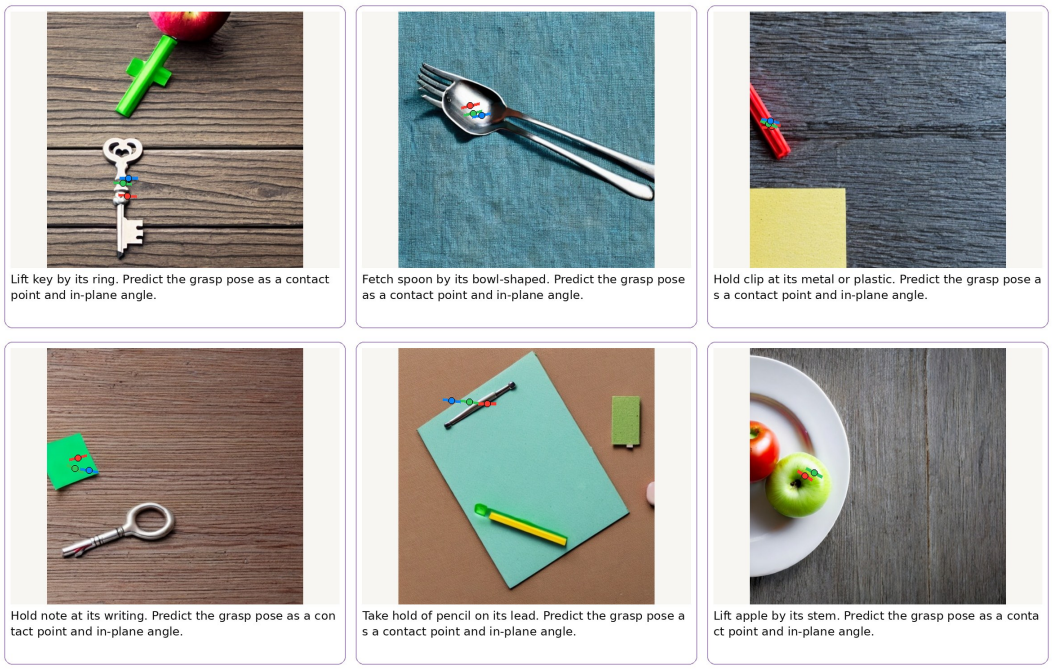


Figure 12: Grasp-pose training examples from Grasp-Anything++. Each segment indicates the contact point and the gripper orientation.

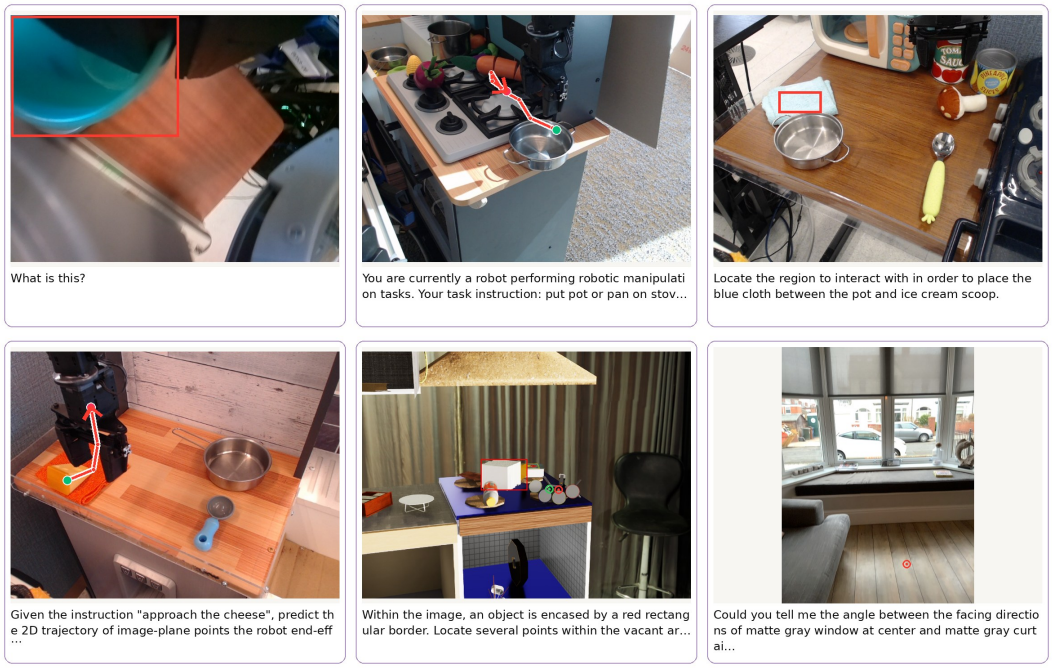


Figure 13: Robot-manipulation training examples: FSD-Robot spatial grounding and trajectory prediction, ShareRobot affordance grounding and trajectory prediction, RoboPoint and RefSpatial point grounding. Green and red markers on trajectories denote the start and end points.

F.3 Unified Span Mask

A single relation states the visibility scope that Sec. 3.2 enforces in both layer families. Writing $b(t)$ for a visibility bound that equals e_σ when position t lies in primitive σ and equals t otherwise, position q attends to position k if and only if

$$M(q,k) = \big[\, k \le b(q) \,\big], \tag{8}$$

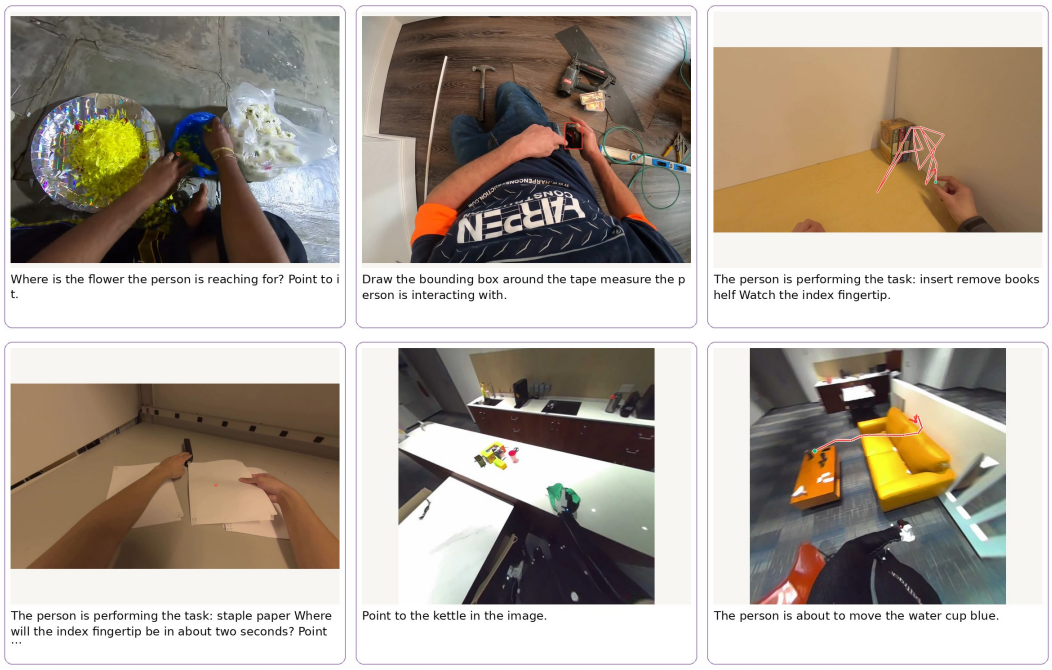


Figure 14: Egocentric training examples: Ego4D next-interaction localization (points and boxes), EgoDex fingertip trajectories and future points, and ACE mocap-grounded interaction.

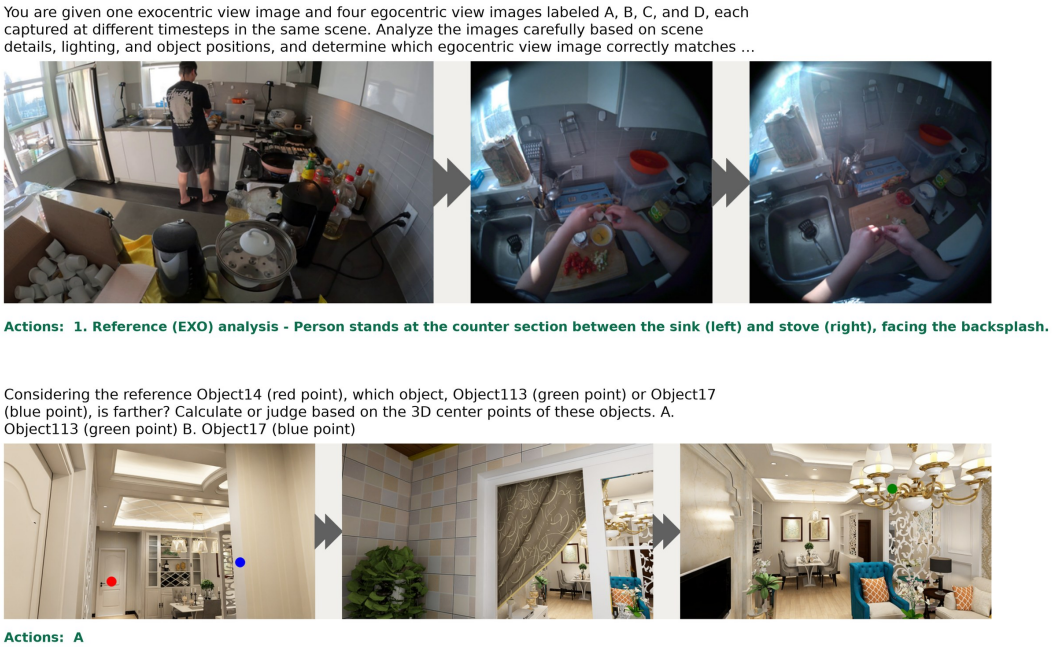


Figure 15: Multi-frame spatial QA training examples from SenseNova and SPAR. Each strip shows three frames of one sample; the caption above the strip is the question and the text below is the annotated answer.

Both layer families implement the same attention relation in Eq. 8. A plain token lies in no span and has $b(q) = q$, so it is causal with $k \leq q$. A coordinate token in span σ has $b(q) = e_\sigma$, so it sees its whole primitive through $k \leq e_\sigma$ together with everything causally before it, but nothing past the end of the primitive and nothing in another sample. A later causal token $q' > e_\sigma$ still sees the primitive tokens because they lie in its past, so the primitive content propagates forward normally. The span layout is computed from the input ids by labeling each maximal run of the coordinate token id and splitting the runs at sample boundaries, so no span crosses a boundary.

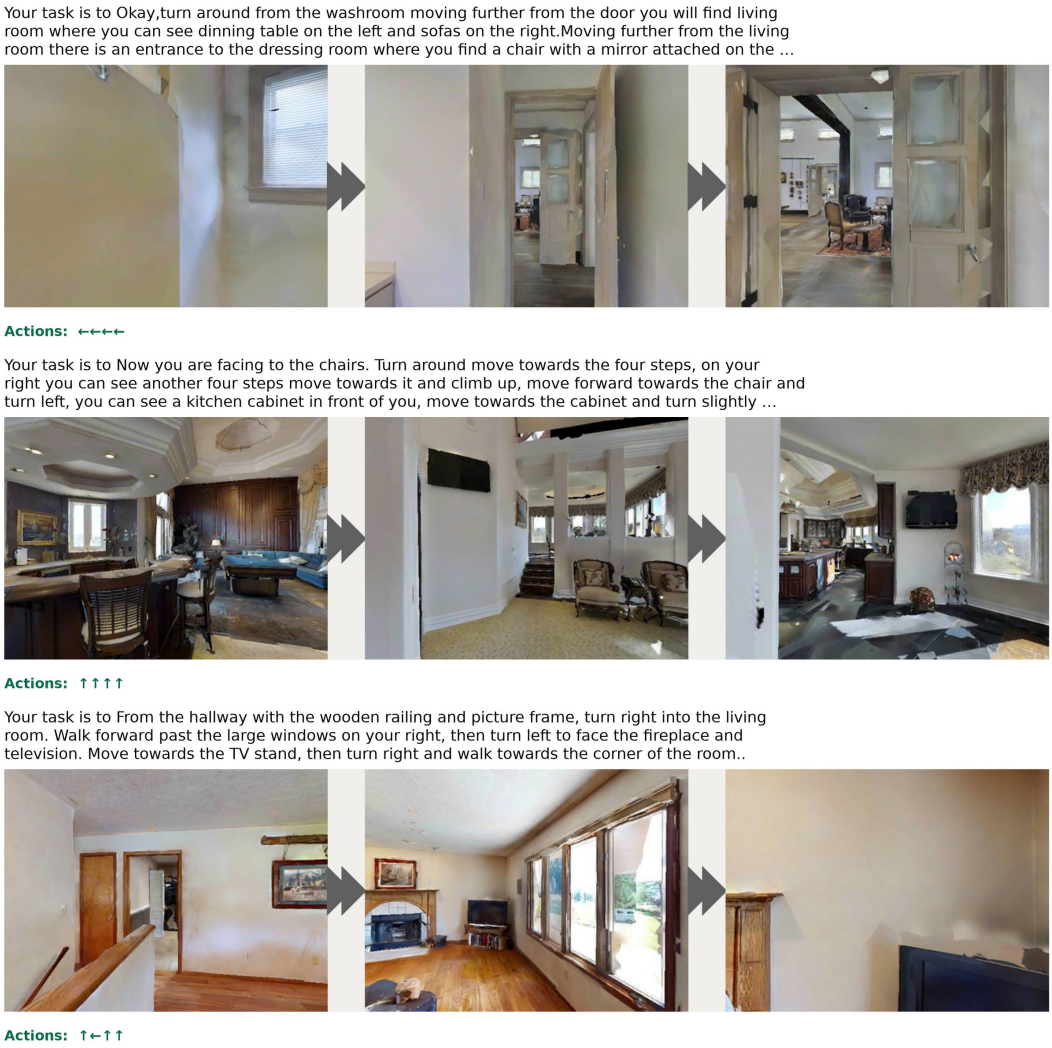


Figure 16: Vision-language navigation training examples from StreamVLN, DAgger, and ScaleVLN. Each strip shows three observations along one episode; the text above is the navigation instruction and the text below is the annotated action sequence.

F.4 Full Attention Layers are Exact by Construction

For a full attention layer the mask is the semantics, so Eq. 8 is applied directly and the readout is exact,

$$O = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d_k}} + \log M\right) V, \tag{9}$$

where $\log M(q, k)$ is 0 when Eq. 8 holds and $-\infty$ otherwise. Only the coordinate rows differ from a plain causal pass. Every other output row is identical to what the stock kernel produces, so the stock kernel runs once over the full sequence and only the coordinate rows, about two to three percent of the tokens, are recomputed with a small bidirectional attention and written back. Kept rows and replaced rows are each exact.

F.5 Linear Attention Layers are Exact by a Query Rewrite

This section gives the full derivation summarized in Sec. 3.2. The gated delta rule carries a state $S_t \in \mathbb{R}^{d_k \times d_v}$,

$$S_t = A_t S_{t-1} + \beta_t k_t v_t^\top, \qquad A_t = \alpha_t \left(I - \beta_t k_t k_t^\top\right), \qquad \alpha_t = e^{g_t}, \tag{10}$$

with causal output $o_t = q_t^\top S_t/\sqrt{d_k}$. For a coordinate token t in span $[s_\sigma, e_\sigma]$ we want the readout taken at the end of the span, $\tilde{o}_t = q_t^\top S_{e_\sigma}/\sqrt{d_k}$, since S_{e_σ} is the state after the whole primitive has been absorbed. Unrolling Eq. 10 from S_t to S_{e_σ} and contracting with $q_t^\top$ gives

$$q_t^\top S_{e_\sigma} = \underbrace{q_t^\top \big(A_{e_\sigma} \cdots A_{t+1}\big) S_t}_{\text{(I) state carry}} + \underbrace{\sum_{j=t+1}^{e_\sigma} q_t^\top \big(A_{e_\sigma} \cdots A_{j+1}\big) \beta_j k_j v_j^\top}_{\text{(II) correction}}. \tag{11}$$

Each A_j is symmetric because α_j and β_j are scalars and $k_j k_j^\top$ is symmetric, so $\big(A_{e_\sigma} \cdots A_{t+1}\big)^\top = A_{t+1} \cdots A_{e_\sigma}$. Term (I) then transposes into an ordinary causal readout of a rewritten query,

$$\text{(I)} = \tilde{q}_t^\top S_t, \qquad \tilde{q}_t = A_{t+1} A_{t+2} \cdots A_{e_\sigma}\, q_t. \tag{12}$$

The substitution is legal because the query never enters the state recurrence in Eq. 10. Only k, v, g, and β build S, so replacing the query at span rows changes those outputs and no state. The same symmetry writes term (II) as a running partial product,

$$\text{(II)} = \sum_{j=t+1}^{e_\sigma} \beta_j \big(k_j^\top \tilde{q}_t^{(j)}\big) v_j^\top, \qquad \tilde{q}_t^{(j)} = A_{j+1} \cdots A_{e_\sigma}\, q_t. \tag{13}$$

Algorithm. For each span we set the running query $r_t \leftarrow q_t$ and the correction $c_t \leftarrow 0$, then walk j from e_σ down to $t+1$,

$$c_t \mathrel{+}= \beta_j \big(k_j^\top r_t\big) v_j, \qquad r_t \leftarrow \alpha_j \big(r_t - \beta_j k_j (k_j^\top r_t)\big) = A_j r_t. \tag{14}$$

After the loop $r_t = \tilde{q}_t$ and c_t equals term (II). We replace the query at span rows by $\tilde{q}_t$, leave k, v, g, β untouched, run the unmodified chunked delta rule kernel once over the whole packed sequence, and add the correction on span rows,

$$\tilde{o}_t = \frac{\tilde{q}_t^\top S_t + c_t}{\sqrt{d_k}}. \tag{15}$$

A span of length L costs $L-1$ rank one contractions, three steps for a four coordinate box, plus one kernel launch. There is no reverse scan, no per span kernel, and no serialized state chain. A length one span is already causal with $\tilde{q}_t = q_t$ and $c_t = 0$, so it is skipped.

F.6 Why a Reverse Scan Fails for the Delta Rule

A tempting alternative writes the span readout as a causal pass plus a within span reverse scan minus a self term. This identity holds for plain gated linear attention, but it fails for the delta rule because the state removal term $-\beta_j k_j k_j^\top$ couples the forward and reverse passes. In our measurements at span length four the residual of this alternative is 0.9 to 1.1 times the size of the effect being computed, with direction cosine between 0.51 and 0.82, and only a length one span is exact. The query rewrite in Eq. 12 to Eq. 15 is algebraically exact up to floating point rounding.

F.7 Numerical Stability

The forgetting gate satisfies $g_t \le 0$, so $\alpha_t \le 1$. With unit norm k_t and $\beta_t \in [0, 1]$, the eigenvalues of $I - \beta_t k_t k_t^\top$ are 1 and $1 - \beta_t$, so $\|A_t\|_2 \le 1$ and $\|\tilde{q}_t\| \le \|q_t\|$. Every rewrite is a contraction, so Eq. 12 cannot grow without bound.

F.8 Implementation Caveats

Two caveats matter when porting the method.

Normalization of the rewritten query. If the kernel normalizes the query and key internally, that step must be moved outside and the flag turned off. The derivation needs unit norm k_j inside A_j and unit norm q_t as the seed of Eq. 12, but the rewritten query $\tilde{q}_t$ is not unit norm and must not be renormalized. The same small constant used by the kernel keeps the non span rows consistent.

Grouped value attention. When the number of value heads exceeds the number of query key heads, each value head carries its own state driven by its own gates, so each head calls for a different rewritten query. A single query per query key head cannot express this, so the query and key are broadcast to the number of value heads before the rewrite. The reference stack broadcasts rather than averaging.

F.9 Optional Span Bidirectional Short Convolution

The gated delta rule applies a depthwise causal convolution of width W to the fused query key value features before the scan, so a coordinate token still assembles its features from a causal window. To make the features span aware rather than only the readout, we add mirror taps clipped at the span end,

$$y_t \mathrel{+}= \sum_{d=1}^{W-1} \left[\, t + d \le e \,\right] \, w_{W-1-d} \odot x_{t+d}, \tag{16}$$

with the activation applied after the sum. This is off by default. A convolution is not normalized, so on span rows the pre activation scale shifts and those weights were pretrained causal. It is a modeling choice rather than a correctness fix, since the span semantics already hold in all layers without it.

F.10 System Requirements

The derivation needs three things from the surrounding framework. First, a span layout that labels each coordinate run and splits it at sample boundaries. Second, a per microbatch carrier that delivers the layout to the kernels, since a plain thread local value is read as empty when activation recompute reruns the forward on a different thread and would silently train a causal model. Third, a stable cache key for the span plan, since building the plan triggers a device to host synchronization that must be cached on the identity of the stable layout tensor rather than a per call view. Tensor, sequence, pipeline, and expert parallelism are supported. Context parallelism is not supported, because it shards the sequence per rank while the kernel sees the full sequence after its communication step, so the span indices no longer align. A self check requires each patched layer to fire an even and positive number of times per microbatch under activation recompute.

F.11 Reinforcement Learning Consistency

Reinforcement learning adds a generation loop, and the sampling pass and the log probability recompute pass must agree. The span mask lets a coordinate token attend to future tokens of its own primitive, which is the supervised objective but is not a valid causal conditional. The invariant is that the mask which defines the sampling distribution at rollout must be the same mask used when log probabilities are recomputed for the update. Two settings satisfy it. In the first, each primitive is generated as a block whose coordinate slots are decoded jointly under the span mask, so the bidirectional attention is the sampling mechanism and both passes use the same mask. The log probability of a primitive is the joint over its coordinate slots, or a sum over the reveal steps of a block decoding schedule. In the second, the span mask is turned off for both rollout and recompute, so reinforcement learning runs on a standard causal policy and the bidirectional training acts as a supervised prior, with a short causal warmup to close the gap between the bidirectional forward and the causal forward. Sampling causally while scoring with the span mask is the one combination that is silently wrong.

F.12 Validation of the Kernel

We pin each path against a high precision reference. The gated delta rule rewrite matches a step by step recurrence in double precision with relative error between 0.00 and 0.01 and direction cosine 1.000, below the causal rounding noise floor, across span lengths one to thirty two and four seeds. The full attention path matches an explicit mask softmax in double precision with error at or below the noise floor. The optional convolution matches its definition exactly, leaves non span rows unchanged, and shows no cross sample leakage. An end to end run over thirty two layers with a packed

length of sixteen thousand tokens converges without numerical warnings and preserves the even call tally at every step.

G Spatial MOPD: Derivations and Analysis

This section gives the precise definitions behind the objectives of Sec. 3.4, the teacher-training procedure, and the analysis of the two KL directions.

G.1 Supervision Positions and Contexts

Primitive supervision positions $\mathcal{G}(u)$. For a primitive-supervision sample u, $\mathcal{G}(u)$ collects the coordinate prediction positions of all primitives in u. Each position $t \in \mathcal{G}(u)$ is conditioned on the masked context c_t, which contains the visual inputs, the instruction, the preceding context, and the masked positions of the current primitive, and excludes its ground-truth coordinates. Under PBA these positions form one parallel prediction group per primitive, so the per-position distributions are read out from the same forward pass that decoding would use.

Generated positions $\mathcal{A}(y)$. For a student rollout $y \sim \pi_\theta(\cdot \mid u)$, $\mathcal{A}(y)$ collects every position the student actually predicted: language tokens, control tags, and the coordinate slots of each primitive, decoded jointly under the span mask. The causal replay of Sec. 3.3 is excluded from $\mathcal{A}(y)$, because replay recomputes already-selected coordinates and involves no policy decision. Coordinate positions in $\mathcal{A}(y)$ use the same masked contexts as $\mathcal{G}(u)$, whereas language positions use causal contexts that contain the selected coordinates.

G.2 Teacher Training with GRPO

Each task teacher π_{ϕ_f} is initialized from the supervised checkpoint π_{SFT} and trained with GRPO (Shao et al., 2024) on its task family f. For an instruction $u \sim \mathcal{D}_f$, a group of G rollouts $\{y_i\}_{i=1}^{G} \sim \pi_{\phi_f}(\cdot \mid u)$ is collected, and the objective is

$$J(\phi_f) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|\mathcal{A}(y_i)|}\sum_{t\in\mathcal{A}(y_i)}\min\big(z_{i,t}\hat{A}_i,\ \mathrm{clip}(z_{i,t},\,1-\epsilon,\,1+\epsilon)\,\hat{A}_i\big) - \beta\, D_{\mathrm{KL}}\big(\pi_{\phi_f}\,\|\,\pi_{\mathrm{SFT}}\big)\right], \tag{17}$$

where $z_{i,t} = \pi_{\phi_f}(y_{i,t} \mid c_{i,t})/\pi_{\mathrm{old}}(y_{i,t} \mid c_{i,t})$ is the importance ratio to the rollout policy, and the group-relative advantage is

$$\hat{A}_i = \frac{r_i - \mathrm{mean}_j\, r_j}{\mathrm{std}_j\, r_j + \delta}, \tag{18}$$

with r_i the task-family reward of rollout y_i. The reference KL with weight β anchors each teacher to the shared supervised initialization, which preserves the common interface across teachers. Rollouts are generated with the primitive-aware decoding of Sec. 3.3 under the span mask, and the log probabilities in Eq. 17 are recomputed under the same mask, satisfying the consistency requirement of Appendix F.11.

G.3 Stage 1: Forward-KL Primitive Warm-Up

Stage 1 minimizes

$$\mathcal{L}_{\mathrm{prim}}(\theta) = \mathbb{E}_{u\sim\mathcal{D}_{\mathrm{prim}}}\left[\sum_{t\in\mathcal{G}(u)} D_{\mathrm{KL}}\big(\pi_{\mathrm{prim}}(\cdot \mid c_t)\,\|\,\pi_\theta(\cdot \mid c_t)\big)\right], \tag{19}$$

a forward KL with the frozen primitive teacher as the target. The forward direction is mass-covering: wherever the teacher assigns positive probability and the student assigns none, the objective diverges, so the student must place mass on every coordinate configuration the teacher considers plausible. This is the desired behavior while a shared localization foundation is being established, because dropping coordinate modes early would permanently bias every downstream task family. The objective is a plain supervised distillation loss on masked primitive contexts: it involves no sampling from the student, and its gradient is the exact gradient of each per-position KL.

G.4 Stage 2: On-Policy Reverse-KL Distillation

Stage 2 minimizes the objective of Eq. equation 7, restated here per family:

$$\mathcal{L}_f(\theta) = \mathbb{E}_{\substack{u\sim\mathcal{D}_f \\ y\sim\pi_\theta(\cdot|u)}} \left[\sum_{t\in\mathcal{A}(y)} D_{\mathrm{KL}}\big(\pi_\theta(\cdot \mid c_t) \,\|\, \pi_{\phi_f}(\cdot \mid c_t)\big) \right]. \tag{20}$$

Gradient decomposition. Both the rollout distribution and the KL terms depend on θ. Writing $g(u,y) = \sum_{t\in\mathcal{A}(y)} D_{\mathrm{KL}}(\pi_\theta(\cdot \mid c_t) \,\|\, \pi_{\phi_f}(\cdot \mid c_t))$, the exact gradient is

$$\nabla_\theta \mathcal{L}_f = \mathbb{E}_{u,y}\Big[\underbrace{\nabla_\theta\, g(u,y)}_{\text{(I) distillation gradient}} + \underbrace{g(u,y) \sum_{t\in\mathcal{A}(y)} \nabla_\theta \log \pi_\theta(y_t \mid c_t)}_{\text{(II) self-weighting term}} \Big], \tag{21}$$

where the sum in (II) is the score function of the rollout under the student, and coordinate groups contribute the joint log probability of the group. Term (I) directly reduces each per-position divergence on the student's own rollouts. Term (II) reweights whole rollouts by their total divergence, moving sampling mass toward regions where the student already agrees with the teacher. If the sampling distribution is treated as constant, term (II) vanishes and the update reduces to distillation on self-generated rollouts, the standard on-policy distillation surrogate (Agarwal et al., 2024).

Mode-seeking and format sharpness. The reverse direction is zero-forcing: wherever the student assigns probability but the task teacher assigns none, the objective diverges. Errors that on-policy rollouts actually make, such as an out-of-range coordinate token or a malformed container, receive direct gradient signal at exactly the positions where they occur. The forward direction could not provide this, since it only constrains the student on the teacher's support.

Per-sample teacher assignment. A multi-teacher objective that averages the teachers, for example $D_{\mathrm{KL}}(\sum_f w_f \pi_{\phi_f} \,\|\, \pi_\theta)$, would supervise the student toward a mixture that may itself be an invalid distribution for every task, for example blending two coordinate conventions. Spatial MOPD instead assigns each rollout to exactly one teacher through its task family: the supervision target is always a single specialized distribution, and the mixture weights ρ_f control only the exposure of each family, never the supervision target itself. The total objective $\sum_f \rho_f \mathcal{L}_f(\theta)$ is a convex combination, so no cross-teacher term appears in the loss; interaction between families happens only through the shared parameters, which is precisely the capability integration the student is meant to learn.

Role of the Stage 1 warm-up. Stage 2 supervises only positions the student actually generates. Without a localization foundation, early rollouts would contain malformed primitives at a high rate, so most of the rollout budget would be spent on format repair rather than on the task-specific behaviors the teachers encode. Stage 1 ensures that π_θ already produces well-formed primitives, so that Stage 2 rollouts land in regions the task teachers can score meaningfully.

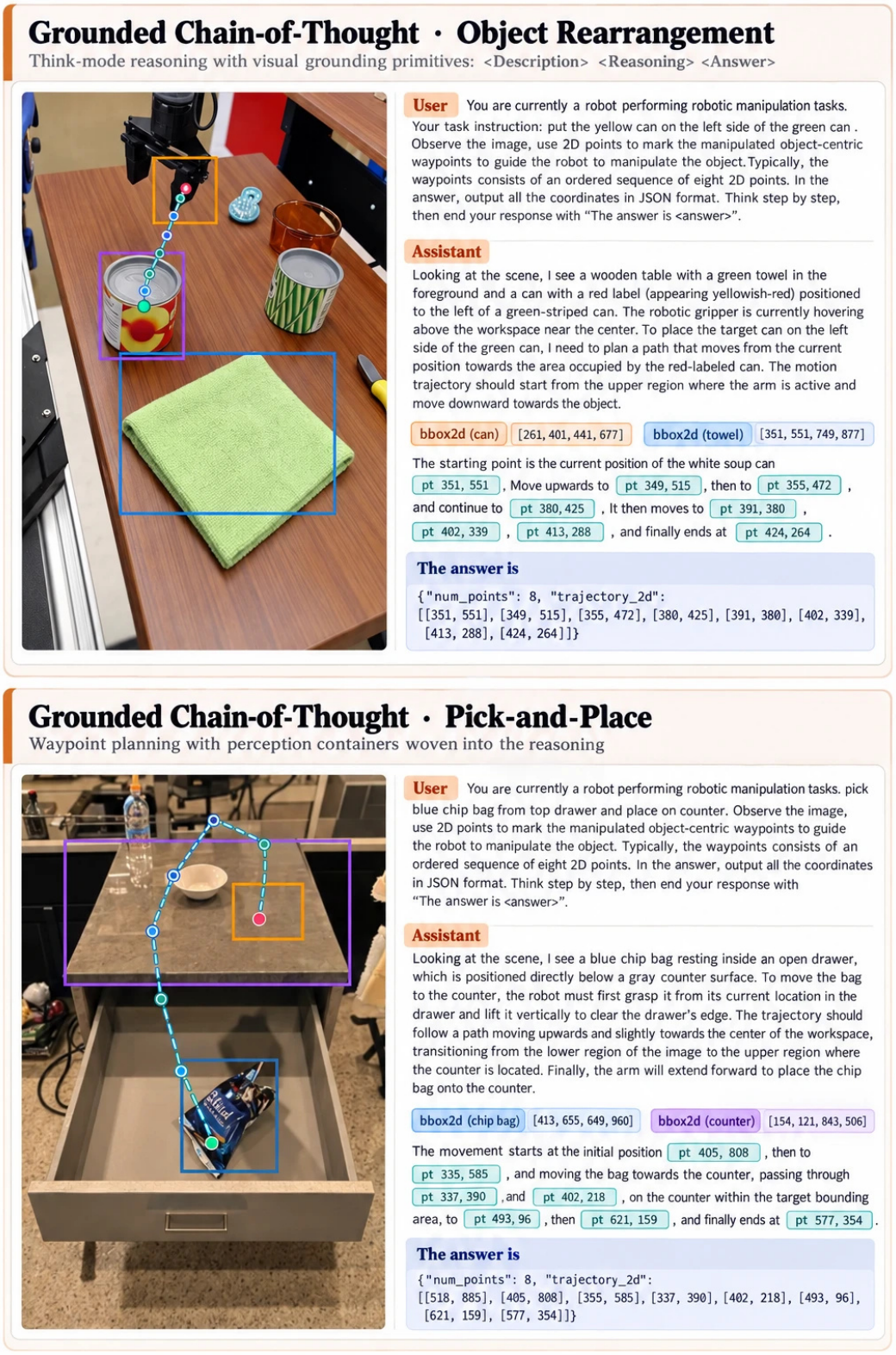


Figure 17: Think-mode grounded chain-of-thought dialogues. Each panel shows one training example: the image with ground-truth primitives overlaid on the left, and the user instruction with the assistant's staged response on the right. Perception containers are rendered as colored chips whose colors match the corresponding boxes, points, and trajectories on the image; the responses interleave *Description*, *Reasoning*, and *Answer* stages with `bbox2d`, `point`, and `trajectory` primitives woven into the reasoning text.