RoboBrain 2.0 Technical Report

BAAI RoboBrain Team

Please see Contributions and Author List for more author details.

Abstract

We introduce **RoboBrain 2.0**, our latest generation of embodied vision-language foundation models, designed to unify perception, reasoning, and planning for complex embodied tasks in physical environments. It comes in two variants: a lightweight 7B model and a full-scale 32B model, featuring a heterogeneous architecture with a vision encoder and a language model. Despite its compact size, RoboBrain 2.0 achieves strong performance across a wide spectrum of embodied reasoning tasks. On both spatial and temporal benchmarks, the 32B variant achieves leading results, surpassing prior open-source and proprietary models. In particular, it supports key real-world embodied AI capabilities, including spatial understanding (e.g., affordance prediction, spatial referring, trajectory forecasting) and temporal decision-making (e.g., closed-loop interaction, multi-agent long-horizon planning, and scene graph updating). This report details the model architecture, data construction, multi-stage training strategies, infrastructure and practical applications. We hope RoboBrain 2.0 advances embodied AI research and serves as a practical step toward building generalist embodied agents. The code, checkpoint and benchmark are available at https://superrobobrain.github.io.

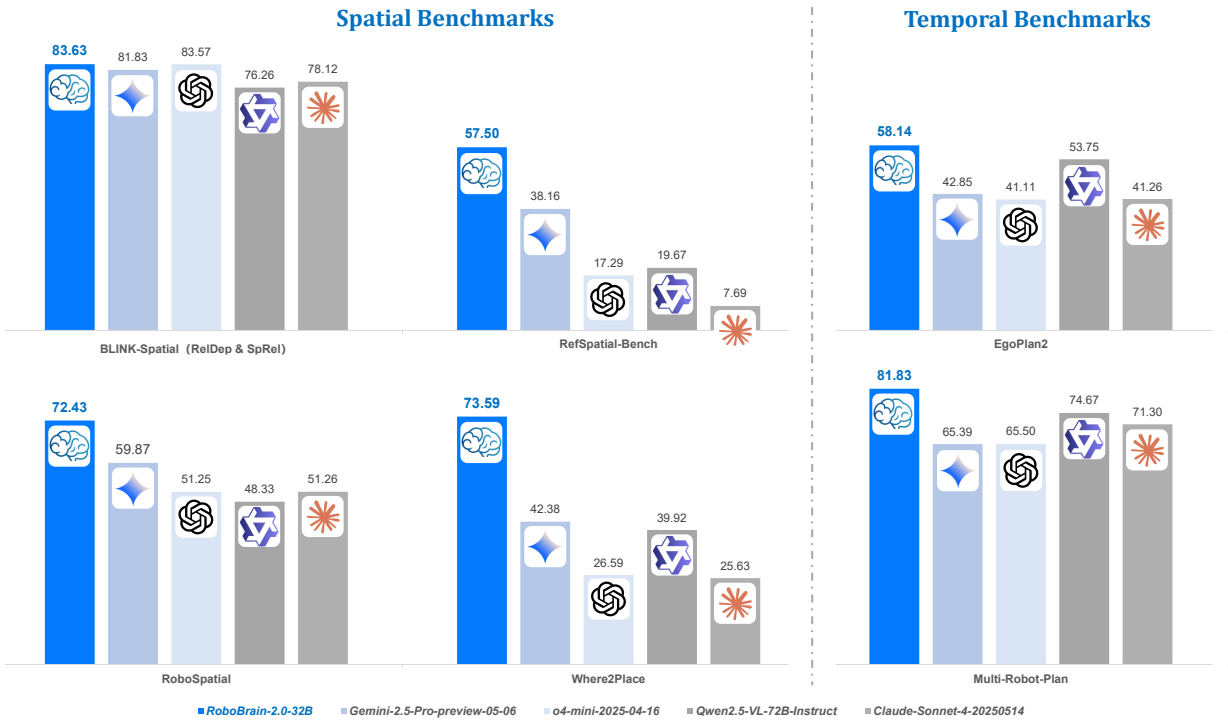


Figure 1 Benchmark comparison across spatial and temporal reasoning. RoboBrain2.0-32B achieves best performance on both spatial and temporal reasoning benchmarks across BLINK-Spatial, RoboSpatial, RefSpatial-Bench, Where2Place, EgoPlan2 and Multi-Robot-Plan, outperforming prior open-source models and proprietary models.

Contents

1 Introduction

In recent years, large language models (LLMs) and vision-language models (VLMs) have emerged as key driving forces in the advancement of general artificial intelligence (AGI). Within digital environments, these models have demonstrated remarkable capabilities in perception [5, 16, 83], understanding [22, 73], and reasoning [2, 17, 18, 45, 65], and have been widely applied in tasks such as multimodal question answering [35, 60], image generation and editing [24, 57], GUI control [37, 71], and video understanding [7, 63, 72]. They have also seen early adoption in practical domains such as education, healthcare, search, and intelligent assistants [11, 21, 82].

However, bridging the gap between "digital intelligence" and "physical intelligence"—enabling models to perceive their surroundings, understand embodied tasks, and interact with the real world—remains a critical challenge on the path toward AGI. Embodied foundation models [4, 64, 74] represent a promising research direction toward physical intelligence. Several recent efforts have extended the capabilities of LLMs and VLMs to embodied scenarios, advancing multimodal fusion, perception, and action execution. While these models have achieved encouraging progress, they still face three fundamental capability bottlenecks when deployed in complex and open-ended real-world environments: **(1) Limited spatial understanding:** Current models struggle to accurately model relative and absolute spatial relationships and identify affordances in physical environments, which hinders real-world applicability; **(2) Weak temporal modeling:** The lack of understanding of multi-stage, cross-agent temporal dependencies and feedback mechanisms limits long-horizon planning and closed-loop control; **(3) Insufficient reasoning chains:** Existing models are often incapable of extracting causal logic from complex human instructions and aligning it with dynamic environmental states, restricting their generalization to open-ended embodied tasks.

To address these challenges, we present **RoboBrain 2.0**, our latest generation of embodied vision-language foundation models, tailored to bridge perception, reasoning, and planning in physically environments. RoboBrain 2.0 processes visual observations and language instructions in a unified architecture, enabling holistic understanding of the environment, goal-directed reasoning, and long-horizon planning. We release two variants of the model: the lightweight *RoboBrain 2.0–7B* and the full-scale *RoboBrain 2.0–32B*, designed to meet different deployment needs under varying resource constraints. On both spatial reasoning and temporal reasoning benchmarks, the 32B variant mostly achieves state-of-the-art performance, outperforming prior open-source and proprietary models, as shown in Figure 1. Model capabilities are summarized in Figure 2.

This report provides a systematic overview of the design principles, core components and key innovations. In particular, we highlight the extensive data contributions that support spatial understanding, temporal reasoning, and causal inference, which form the foundation of RoboBrain 2.0's capabilities. To address the scarcity of spatial data, we develop a spatial data synthesis pipeline that constructs large-scale, high-quality datasets spanning tasks such as pointing, affordance prediction, and trajectory generation. To improve temporal reasoning and feedback modeling, we design multi-robot coordination templates across common scenarios via RoboOS [61], generate cross-agent long-horizon planning trajectories using external models [31], and simulate randomized failure events to collect closed-loop feedback data that enhances model robustness. To further enrich reasoning data, we extract step-by-step thought traces from powerful reasoning VLMs [22], conditioned on spatiotemporal task contexts. These traces serve as supervision signals for learning causal chains across vision, language, and action.

RoboBrain 2.0 adopts a high-efficiency heterogeneous architecture and a progressive multi-stage training strategy to support spatial understanding, temporal modeling, and long-chain causal reasoning in embodied settings. The model comprises a lightweight vision encoder with approximately `689M` parameters and a decoder-only language model with `7B/32B` parameters. It is trained using a three-stage curriculum—covering foundational spatiotemporal learning, embodied spatiotemporal enhancement, and chain-of-thought reasoning—on large-scale multimodal and embodied datasets. Training is conducted using our open-source framework *FlagScale*, which integrates hybrid parallelism, pre-allocated memory optimization, high-throughput I/O pipelines, and robust fault tolerance. These infrastructure innovations significantly reduce training and deployment costs while ensuring scalability for large-scale multimodal models. We evaluate RoboBrain 2.0 on over `12` public benchmarks covering spatial understanding, temporal modeling and multimodal reasoning, achieving state-of-the-art results on `6` of them despite its compact size. We release code, checkpoints, and benchmarks as open-source resources to benefit the research community. These materials facilitate reproducible

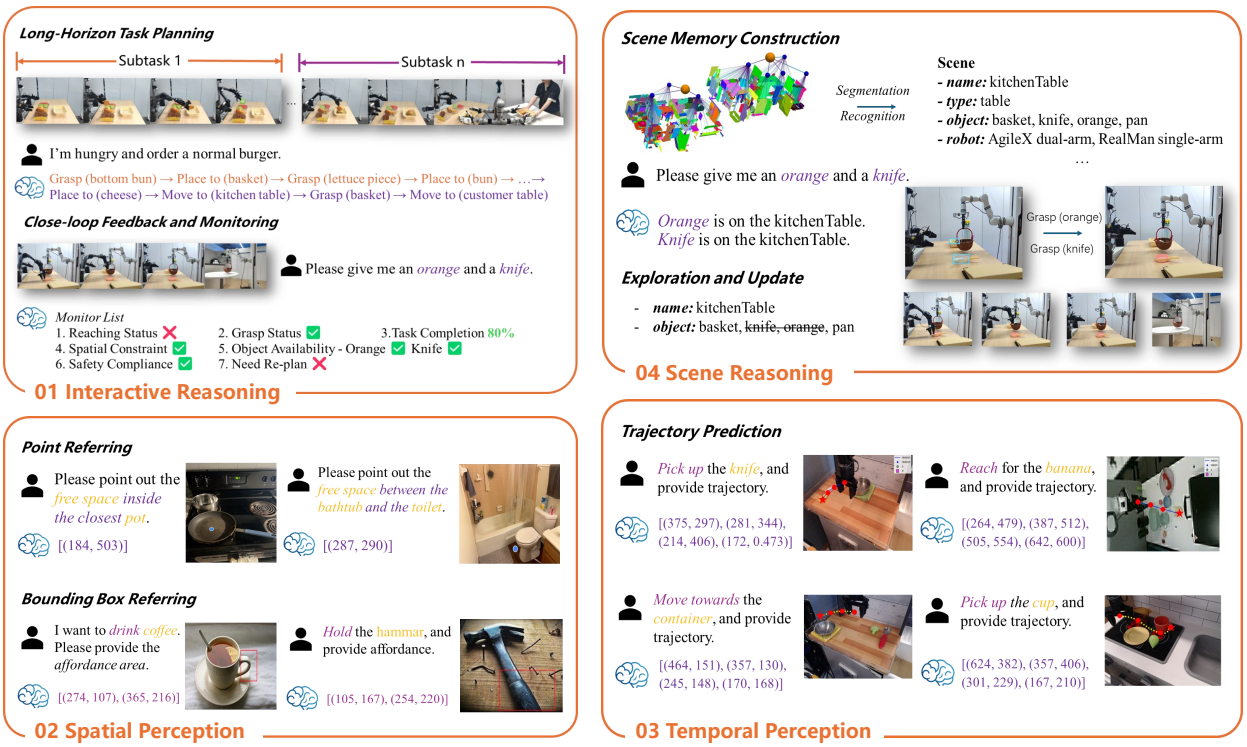


Figure 2 The overview of RoboBrain 2.0's Capabilities. RoboBrain 2.0 supports interactive reasoning with long-horizon planning and closed-loop feedback, spatial perception for precise point and bounding box prediction from complex instructions, temporal perception for future trajectory estimation, and scene reasoning through real-time scene graph construction and updating.

research, accelerate embodied AI development, and enable practical deployment in robotic systems.

To provide a comprehensive view of RoboBrain 2.0's architecture, training methodology, and capabilities, this report is organized as follows: Section 2 introduces the overall model design, including the coordination between the vision encoder and language model, as well as image and video input strategies. Section 3 describes the data curation and construction process, covering three major categories: general multimodal understanding, spatial reasoning, and temporal modeling. Section 4 presents our multi-stage training strategies, including foundational spatiotemporal learning, embodied enhancement, and chain-of-thought reasoning. Section 5 outlines the infrastructure stack supporting scalable training and inference, including hybrid parallelization, memory optimization, data loading, and failure recovery. Section 6 reports extensive evaluation results on public benchmarks, highlights RoboBrain 2.0's capabilities in spatial reasoning, temporal feedback, and embodied planning. Finally, Section 7 discusses current limitations, and outlines future research directions.

2 Architecture

RoboBrain 2.0 employs a modular encoder-decoder architecture that unifies perception, reasoning, and planning for complex embodied tasks. As shown in Figure 3, it processes multi-view visual observations and natural language instructions through four core components: (1) a tokenizer for textual/structured inputs, (2) a vision encoder, (3) an MLP projector mapping visual features to the language model's token space, and (4) a language model backbone initialized from Qwen2.5-VL [5]. Unlike conventional VLMs [2, 22] focused on general static VQA, RoboBrain 2.0 maintains strong general VQA capabilities while specializing in embodied reasoning tasks like spatial perception, temporal modeling, and long-chain causal reasoning. The architecture encodes high-resolution images, multi-view inputs, video frames, language instructions, and scene graphs into a unified multimodal token sequence for comprehensive processing.

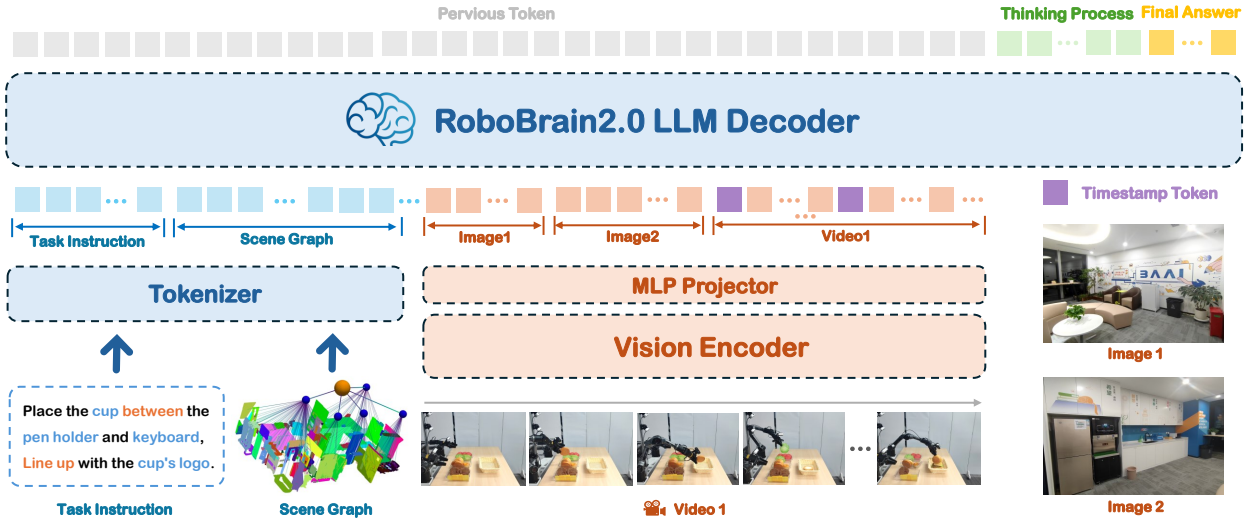


Figure 3 The Architecture of RoboBrain 2.0. The model supports multi-image, long video, and high-resolution visual inputs, along with complex task instructions and structured scene graphs on the language side. Visual inputs are processed via a vision encoder and an MLP projector, while textual inputs are tokenized into a unified token stream. All inputs are fed into an LLM decoder that performs long-chain-of-thought reasoning and generates a variety of outputs depending on the task, including structured plans, spatial relations, or relative and absolute coordinates.

2.1 Input Modalities and Tokenization

RoboBrain 2.0 supports a diverse set of input modalities tailored for embodied AI tasks:

- **Language instructions:** Natural language commands describing high-level goals or low-level actions. RoboBrain 2.0 processes natural language commands spanning different abstraction levels: from high-level, spatially grounded instructions (e.g., *"Carry the apple to the nearest table, aligned with the leftmost cup"*) to low-level motor commands (e.g., *"Navigate to the nearest table", "Grasp the apple", "Detect position aligned with the leftmost cup", "Place the apple into the box"*).
- **Scene graph:** A structured JSON representation of the explored environment, containing information about discovered objects, their categories, spatial locations, and embodiment configuration (e.g., `name: KitchenTable1, type: table, object: [basket, knife], robot: RealMan-single-arm`).
- **Multi-view static images:** Images captured from multiple viewpoints, such as head-mounted cameras, wrist-mounted cameras, or multi-view projections from a 3D environment. These are processed independently by the vision encoder and concatenated into a unified token sequence.
- **Video frames:** Video sequences (e.g., egocentric views from the agent), optionally annotated with timestamp tokens [5] to facilitate temporal grounding and reasoning.

Language instructions and scene graphs are tokenized using the language tokenizer. Visual inputs—including multi-view images and video frames—are processed by the vision encoder into dense visual embeddings, which are then projected into the LLM's token space through an MLP projector, enabling unified multi-modal reasoning within the decoder.

2.2 Vision Encoder and Projection

RoboBrain 2.0 vision encoder supports dynamic-resolution image and video inputs through adaptive positional encoding and windowed attention mechanisms [5]. This design choice enables efficient processing of high-resolution and multi-view visual observations common in embodied tasks.

To accommodate the long-horizon and temporally grounded nature of such tasks, we adopt frame-wise visual tokenization with multi-dimensional RoPE [5] for spatiotemporal encoding. Each visual embedding is projected via a lightweight MLP into the token space of the language model. For multi-view scenarios, visual tokens from different camera perspectives are serialized and augmented with view-specific positional identifiers before being fused with other input modalities.

2.3 LLM Decoder and Output Representations

RoboBrain 2.0 employs a decoder-only language model designed to unify high-level reasoning and spatially grounded output generation. Unlike conventional VLMs that primarily return short-form answers to static prompts, RoboBrain 2.0 flexibly supports both concise responses and multi-step chain-of-thought reasoning. This capability enables deeper understanding of complex instructions and physical scenes.

To enable the decoder to handle embodied tasks, the decoder is trained to produce a diverse range of outputs, including semantically grounded expressions (e.g., referring to objects or actions), spatial coordinates (e.g., absolute positions or bounding boxes), and intermediate reasoning traces. Rotary positional encodings and temporally conditioned tokens allow the model to maintain coherence across multi-round perception-action loops, which are essential for long-horizon planning in dynamic environments. Output formats supported by RoboBrain 2.0 include: **(1) Free-form text:** Used for task decomposition, scene graph updates, agent invocation, and human-agent dialogue. **(2) Spatial coordinates:** Used to represent point locations, bounding boxes, or trajectories in the image space for downstream controllers. **(3) Reasoning traces (Optional):** Long-chain-of-thought explanations to support deep problem solving and decision transparency.

This unified decoding formulation allows RoboBrain 2.0 to effectively handle a wide range of embodied tasks, from spatial grounding and visual understanding to long-horizon multi-agent planning and causal reasoning.

3 Training Data

As shown in Figure 4, RoboBrain 2.0 is trained on a diverse and extensive dataset designed to enhance its capabilities in spatial understanding, temporal modeling and long-chain causal reasoning in embodied settings. The training data encompasses a wide range of modalities, including high-resolution images, multi-view inputs, video sequences, scene graph and natural language instructions. This comprehensive dataset is meticulously categorized into three primary types: general multimodal understanding, spatial perception, and temporal modeling, ensuring the model can effectively perceive, reason, and plan in complex physical environments.

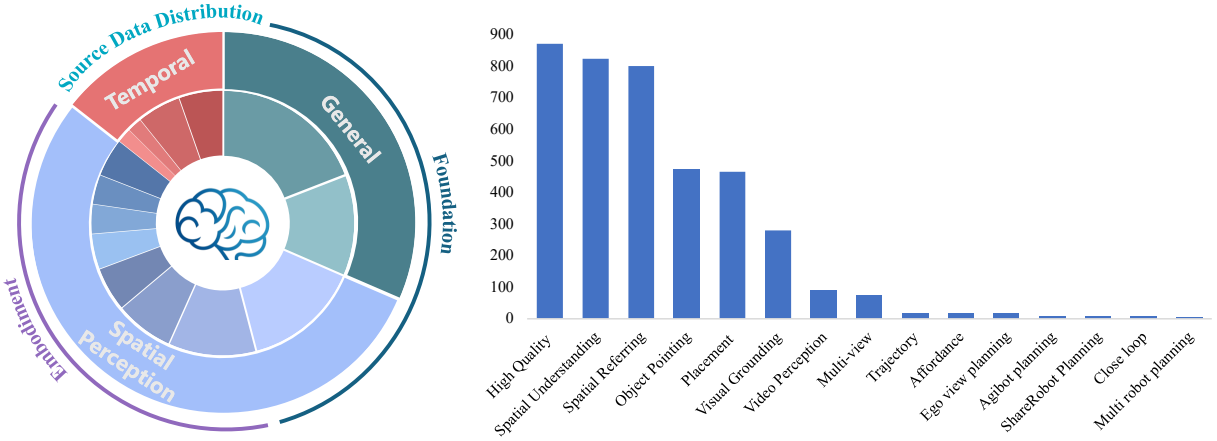


Figure 4 Training Data Distribution for RoboBrain 2.0. This figure illustrates the distribution of training data supporting RoboBrain 2.0's capabilities, including interactive reasoning with long-horizon planning and closed-loop feedback, spatial perception for precise point and bounding box prediction from complex instructions, and multi-agent collaboration tasks, which is meticulously categorized into three primary types: general multimodal understanding, spatial perception, and temporal modeling.

3.1 General MLLM VQA

High Quality Data. The general training dataset for RoboBrain 2.0 includes 873K high-quality samples, primarily derived from LLaVA-665K [33] and LRV-400K [32], spanning standard Visual Question Answering(VQA), region-level queries, OCR-based VQA, and visual dialogues. **(1) LLaVA-665K** serves as the primary

source and contains diverse VQA-style data, including standard VQA datasets, OCR-based questions, region-level queries, visual conversations, and language-only dialogues. To improve training efficiency, multiple question-answer(QA) pairs from the same image merge into single conversations; invalid ShareGPT [10] entries are filtered out, and overly long conversations (>2048 tokens) are truncated (resulting in 40K valid samples). Specifically, A-OKVQA [54] samples are augmented by duplicating choices to balance multiple-choice formats, OCR-VQA [41] contributes 80K sampled conversations focused on scene text understanding, Visual Genome(VG) [27] provides dense object-level annotations limited to 10 entries per image with additional captions, and RefCOCO [76] dialogues are split into short multi-turn segments (<10 exchanges). Language-only conversations, which are generally longer than visual ones, are sampled in single-modality batches to improve throughput by 25% without performance degradation. After removing bounding-box-dependent QA pairs, 531K high-quality samples are retained from this source. **(2) LRV-400K** is synthetically generated using GPT-4 [44] under a few-shot instruction-following setting. It produces 400K image-conditioned instructions across 16 vision-language tasks with textual answers. Unlike prior works that rely on sparse image captions, this dataset leverages the dense annotations in VG (e.g., bounding boxes, dimensions, and ∼21 object regions per image). GPT-4 generates both declarative and interrogative prompts for each image, with 10 tasks randomly sampled per instance. After filtering out bounding-box-related QA pairs, 342K samples are selected for training.

3.2 Spatial Data

Visual Grounding. The visual grounding dataset is constructed to enhance multimodal understanding through precise object-level localization, leveraging the extensive annotations from LVIS [19]. We carefully curate 152K high-resolution images from LVIS, ensuring broad coverage of diverse object categories and complex visual scenes. Each object annotation is converted into standardized bounding box coordinates (x_1, y_1, x_2, y_2) representing the top-left and bottom-right corners, enabling consistent spatial referencing. To facilitate rich visual dialogue, we generated 86K conversational sequences, each containing multiple rounds of QA pairs that progressively explore visual relationships, attribute reasoning, and contextual understanding. The dataset maintains a balanced distribution across object categories while preserving challenging cases of occlusion, viewpoint variation, and rare instances to support robust visual grounding.

Object Pointing. The object pointing dataset is constructed to enable RoboBrain 2.0 to identify the locations of specified objects through pointing within an image. We leverage the Pixmo-Points [13] dataset, which includes 2.3M point annotations across 223K images as our data source. However, direct utilization of Pixmo-Points data for RoboBrain 2.0 training presents challenges due to densely repeated object instances (e.g., books on a shelf). To address this, we implement a two-step filtering process: (1) we discard annotations with more than ten labeled points to simplify training, and (2) we use GPT-4o [22] as a scene analyzer to select only indoor-relevant objects, such as kitchenware, furniture, and decorations, excluding irrelevant or outdoor scenes. This process yields 190K QA pairs for 64K images with reduced clutter, making the data more suitable for embodied contexts. To construct QA pairs for pointing tasks, we construct 28 human-designed templates, such as "*Point out all instances of {label} in the image.*" or "*Help me find {label} in the image by pointing to them.*" Here, *{label}* refers to object categories from the annotations. Templates are randomly selected to ensure linguistic diversity and improve the model's generalization ability in referencing tasks. For object reference pointing, we incorporate object reference data sourced from RoboPoint [77], which includes 347K QA annotations across 288K images. To address the potential issue of excessive points hindering training convergence, we randomly sample up to ten points per question. Additionally, the normalized coordinates are converted into absolute values to better support RoboBrain 2.0 training.

Affordance. The affordance dataset focuses on understanding object functionality and spatial vacant areas for placement. For object affordance recognition, we utilize part-level annotations from PACO-LVIS [51], covering 75 object categories and 200 part categories across 46K images. Bounding boxes and segmentation masks are extracted for both whole objects and their functional parts. These annotations are transformed into bounding box coordinates (x_1, y_1, x_2, y_2), serving as ground truth labels for affordance prediction tasks. Questions are constructed using GPT-4o [22] to query object functionality and part usage, e.g., "*Which part of a handbag can be grasped to carry it?*" for the handle of a handbag. For whole-object affordances, questions avoid naming the object directly, such as "*What device can be moved to control the cursor on a screen?*" for a mouse (computer equipment). This automatic process results in 561K QA pairs. For spatial

affordance learning, we include region reference data from RoboPoint [77]. This dataset consists of 270K images with 320K QA pairs and 14 spatial relationship labels. Each annotation is converted into a set of absolute coordinates $[(x_1, y_1), (x_2, y_2), ...]$, and ground truth points are resampled to a maximum of ten points per answer for optimization. This dataset enables RoboBrain 2.0 to reason about spatial affordances for object placement in real-world settings.

Spatial Understanding. To enhance RoboBrain 2.0's 3D spatial reasoning, we present the Spatial Understanding Dataset, comprising 826K samples. This dataset emphasizes object-centric spatial attributes (e.g., position, orientation) and inter-object relations (e.g., distance, direction), covering both qualitative and quantitative aspects. It covers 31 distinct spatial concepts, substantially surpassing the ∼15 typically found in previous datasets. We partially adopt the RefSpatial [81] pipeline to construct 2D web image and 3D video datasets via automated template- and LLM-based generation: **(1) 2D web images** aim to provide core spatial concepts and depth perception across diverse indoor and outdoor scenes. To bridge scale and category gaps between these domains, we utilize the large-scale OpenImage [28] dataset. Since direct 3D reasoning from 2D images is challenging, we convert them into pseudo-3D scene graphs. Specifically, after filtering 1.7M images to 466K, we first use RAM [79] for object category prediction and GroundingDINO [34] for 2D boxes Detection. Then we enhance using Qwen2.5-VL [50] and a heuristic method to generate hierarchical captions given the 2D bounding box, ranging from coarse (e.g., "cup") to fine-grained (e.g., "the third cup from the left"). This enables unambiguous spatial referring in cluttered environments and captures both coarse and fine-grained spatial references. Next, we use UniDepth V2 [48] and WildeCamera [84] for depth and camera intrinsics to enable 3D point cloud reconstruction. Finally, combining this with object boxes from GroundingDINO [34] and masks from SAM 2.1 [52], each scene graph includes object labels, 2D boxes, instance masks, and object-level point clouds, yielding axis-aligned 3D boxes. Object captions serve as nodes, and spatial relations form the edges. QA pairs are generated via templates and LLMs (e.g., QwQ [66]), including object-location questions derived from the hierarchical captions. **(2) 3D scene-based videos** integrates multimodal 3D scene understanding data from five original datasets: MMScan [38], 3RScan [69], ScanQA [3], SQA3D [39], and SpaceR [46]. We conduct template-based question filtering through rigorous data processing to ensure task relevance, perform multi-stage quality screening (e.g., consistency checks, outlier removal), and standardize all formats into a unified representation. This curation enables fine-grained environmental perception with enhanced reliability, supporting tasks ranging from object localization to complex spatial reasoning in 3D scenes. **(3) 3D embodied videos** focus on fine-grained spatial understanding in indoor environments. We leverage the CA-1M [29] dataset, filtering 2M frames to 100K high-quality ones. Compared to 2D, the availability of accurate 3D bounding boxes allows us to construct richer scene graphs with more diverse spatial relations, thereby generating more quantitative QA pairs (e.g., size, distances).

Spatial Referring. After enhancing foundational 3D spatial understanding, we extend these capabilities to physical-world interactions by introducing the Spatial Referring Dataset [81], consisting of 802K samples. Unlike prior datasets in visual grounding or object pointing, which often deal with ambiguous or multiple referents, this dataset targets a single unambiguous target, aligning with robotic applications such as precise pick-and-place that demand accurate object identification and localization. Following the RefSpatial [81] construction pipeline, for location data, we sample caption-point pairs from scene graphs built on 2D web images (OpenImage [28]) and 3D embodied videos (CA-1M [29]), using hierarchical captions. For placement data, we leverage fully annotated 3D datasets to generate top-down occupancy maps encoding object positions, orientations, and metric spatial relations (e.g., "10cm right of the chair"), facilitating accurate spatial referring.

3.3 Temporal Data

Ego-View Planning. We construct Ego-View Planning dataset by partially processing the EgoPlan-IT [9] dataset, which contains 50K automatically generated samples. For each selected task instance, we extract multiple frames from prior actions to represent task progress, and one frame to capture the current viewpoint. To enhance linguistic variety, we use multiple prompt templates that describe the task goal, video context, and current observation. Each question includes the correct next action along with up to three distractor actions randomly sampled from negative examples. This setup supports multimodal instruction tuning with diverse visual and textual input, aimed at improving egocentric task planning performance.

ShareRobot Planning. The ShareRobot dataset [23] is a large-scale, fine-grained resource for robotic manipu-

lation, offering multi-dimensional annotations tailored for task planning. Its planning component provides detailed low-level instructions aligned with individual video frames, effectively transforming high-level task descriptions into structured and executable sub-tasks. Each data instance includes precise planning annotations to support accurate and consistent task execution. The dataset comprises 1M QA pairs from 51K instances, spanning 102 diverse scenes across 12 robot embodiments and 107 atomic tasks filtered according to the Open-X-Embodiment taxonomy [47]. All planning data were meticulously annotated by human experts following the RoboVQA [55] format, enabling models to learn robust multi-step planning strategies grounded in diverse real-world scenarios. The scale, quality, and diversity of ShareRobot help improve the model's ability to perform fine-grained reasoning and task decomposition in complex embodied environments.

Agitbot Planning. The AgiBot Planning dataset is a large-scale robotics task planning dataset built upon the AgiBot-World [6] dataset, comprising 9,148 QA pairs across 19 manipulation tasks with 109,378 first-person perspective images. Each sample contains 4-17 consecutive frames documenting task progression with multimodal conversational format. AgiBot-Planning provides step-by-step planning instructions that transform high-level goals into executable sub-tasks. Each data point includes current objectives, historical steps, and required subsequent actions. The dataset covers diverse scenarios from household refrigerator operations to supermarket shopping tasks across different environments. The meticulously crafted annotations use standardized conversational formats, enabling models to learn from varied real-world contexts. Through continuous visual sequences and fine-grained action plans, AgiBot-Planning enhances RoboBrain 2.0's ability to perform long-horizon task planning and spatial reasoning in complex embodied scenarios.

Multi-Robot Planning. The Multi-Robot Planning dataset is constructed by simulating collaborative task scenarios across three environments—household, supermarket, and restaurant—based on RoboOS [61]. Each sample is generated using structured templates that specify a detailed scene graph, robot specifications, and associated tool lists. For every scenario, we design high-level, long-horizon collaborative task goals that require coordination among multiple robots present in the scene, and generate corresponding workflow graphs that decompose the tasks into subtasks with detailed reasoning explanations. Based on these decompositions, we further generate agent-specific robotic tool plans that translate high-level task goals into precise low-level Observation-Action pairs for each subtask. Specifically, we define 1,659 types of multi-robot collaboration tasks across the three environments and produce 44,142 samples using DeepSeek-V3 [31].

Close-Loop Interaction. The Close-Loop Interaction dataset is designed to facilitate advanced embodied reasoning [80], featuring a large-scale collection of synthesized Observation-Thought-Action (OTA) trajectories that combine first-person visual observations with structured thought tokens. It spans 120 diverse indoor environments—including kitchens, bathrooms, bedrooms, and living rooms—containing over 4,000 interactive objects and receptacles. The dataset is constructed within the AI2Thor [25] simulator through a rigorous multi-stage pipeline based on Embodied-Reasoner [78], which includes: (1) crafting task instructions from constrained templates to ensure scene-appropriate validity; (2) deriving key action sequences from an object-affiliation graph encoding functional relationships; and (3) strategically incorporating search actions to emulate realistic exploration. To enrich the depth of reasoning, GPT-4o generates detailed thought processes—covering situational analysis, spatial reasoning, self-reflection, task planning, and verification—which are seamlessly integrated between observations and actions, forming coherent reasoning chains that guide models through complex, long-horizon interactive tasks.

4 Training Strategy

RoboBrain 2.0 achieves embodied capabilities (spatial understanding, temporal modeling, and chain-of-thought reasoning) through a progressive three-phase training strategy, as shown in Table 1. Starting from a robust vision-language foundation, we introduce escalating complexity in embodied supervision, enabling the model to evolve from static perception to dynamic reasoning and actionable planning in real-world environments.

4.1 Stage 1: Foundational Spatiotemporal Learning

The first stage focuses on building general capabilities in spatial perception and temporal understanding. We fine-tune the model on large-scale multimodal datasets covering dense captioning, object localization, interleaved image-text documents, and basic video QA, along with referring expression comprehension. These

Table 1 Detailed configuration for each training stage of the RoboBrain 2.0.

		Stage-1	Stage-2	Stage-3	
		SFT	SFT	COT-SFT	RFT (RLVR)
Data	**Dataset**	Foundation	Embodied	Embodied (Phase 1)	Embodied (Phase 2)
	#Samples	4.8M	224K	195K	45K
Model	**Trainable Part**	Full Model	Full Model	Full Model	Full Model
	#Tunable Parameters	8.29B or 33.45B	8.29B or 33.45B	8.29B or 33.45B	8.29B or 33.45B
Training	**Per-device Batch Size**	2	2	4	1
	Gradient Accumulation	2	2	2	2
	LR: $\{\psi_v^{\text{ViT}}, \phi_v^{\text{LLM}}\}$	1×10^{-4}	1×10^{-5}	1×10^{-5}	1×10^{-6}
	Epoch	1	1	1	3
	Optimizer	AdamW	AdamW	AdamW	AdamW
	Deepspeed	–	–	Zero3	Zero3
	Weight Decay	0.1	0.1	0.1	0.0
	Warmup Ratio	0.01	0.01	0.03	0.00
	LR Schedule	Cosine	Cosine	Cosine	Cosine
	Max Seq. Length	16384	16384	32768	32768
	Max Compl. Length	–	–	–	1024
	Num. of Compl.	–	–	–	8
	GPU Nums	$16/64\times8$	$16/64\times8$	4×8	4×8

datasets span common physical scenes and interaction patterns, helping the model develop fundamental grounding for objects, spatial relations, and motion events. This stage lays the groundwork for understanding egocentric video streams and spatially anchored instructions.

4.2 Stage 2: Embodied Spatiotemporal Enhancement

To better align the model with embodied tasks, we introduce a carefully curated collection of high-resolution, multi-view, and egocentric video datasets, along with instruction-augmented navigation and interaction data. Tasks include viewpoint-aware referring expressions, 3D affordance estimation, and object-centric scene graph construction. This stage of training emphasizes the modeling of long-horizon temporal dependencies, enabling the model to reason over extended sequences of actions and observations. Additionally, it incorporates multi-agent coordination scenarios, where the model learns to interpret and predict the behaviors of other agents in shared environments. To support these capabilities, we employ extended sequence lengths and multi-camera input encoding, allowing the model to process and fuse visual information from multiple viewpoints simultaneously. Through this training stage, the model can integrate historical visual cues with current instructions, fostering more coherent long-horizon planning, robust scene understanding, and adaptive decision-making in dynamic, interactive settings.

4.3 Stage 3: Chain-of-Thought Reasoning in Embodied Contexts

In the third stage, we augment the model's high-level reasoning capabilities using Chain-of-Thought (CoT) methodology, following the two-phase framework of Reason-RFT [62]: CoT-based Supervised Fine-Tuning (CoT-SFT) and Reinforcement Fine-Tuning (RFT). We leverage multi-turn reasoning examples from both synthetic and real-world embodied scenarios, encompassing long-horizon task planning, manipulation prediction, closed-loop interaction, spatiotemporal understanding, and multi-robot collaboration, sourced from Section 3. Specifically, (1) CoT-SFT Phase: We annotate 10% of the constructed training data with CoT rationales annotated by GPT-4o [22] with custom prompts, then perform supervised fine-tuning for initial model from Stage 2. (2) RFT Phase: An additional 10% of the constructed training data is sampled to collect model's responses, with incorrect answers curated into a reformatted training set (e.g., multiple-choice questions or LaTeX/numerical answers). Optimization employs Group Relative Policy Optimization (GRPO) [17], guided by a composite reward function evaluating both answer accuracy and format correctness.

5 Infrastructures

5.1 Large-Scale Training Infrastructure

To improve the efficiency and stability of multimodal model training, we have developed and integrated a series of key optimization techniques, including hybrid parallelism strategies, memory pre-allocation, distributed data loading, kernel fusion, and fine-grained compute-communication overlapping. These optimizations significantly enhance both resource utilization and training throughput. For data preprocessing, we build upon the Megatron–Energon framework [30] and incorporate custom optimization strategies. Our system supports dynamic mixing of multiple datasets containing diverse modalities, including plain text, single image, multiple images, and video, while also allowing for strict sample order preservation within each dataset. A custom WebDataset-based format [1] enables compatibility with various data modalities and greatly reduces preprocessing time while improving flexibility and scalability in data handling.

5.1.1 Multi-Dimensional Hybrid Parallelism

Multimodal models differ significantly from conventional LLMs in both architecture and data characteristics [33]. On the architectural side, multimodal models are inherently heterogeneous: the vision module (e.g., ViT with Adaptor) is typically a small-scale encoder-only component, while the language module is a much larger decoder-only transformer. On the data side, training samples include plain text, single images, multi-image sequences, and videos. The number of image tokens, text tokens, and the length of the fused token sequence can vary dramatically between samples.

These heterogeneities pose substantial challenges to distributed training frameworks. To address this, we implemented several targeted strategies in our custom framework, FlagScale [12]:

- **Non-uniform Pipeline Parallelism** [43]: Since the ViT module appears early in the model and has relatively low computational cost, we reduce the number of LLM layers in the first pipeline stage, thereby improving training throughput without increasing memory overhead.
- **Separate Recompute Strategy**: During the annealing stage, the vision input may contain up to 20,000–30,000 tokens, frequently causing an Out-of-Memory (OOM) error in the ViT module. To mitigate this, we enable recompute [8, 26] only in the ViT module to reduce memory usage of intermediate activations, while disabling recompute in the LLM module to preserve computational efficiency.

5.1.2 Pre-Allocate Memory

In the supervised fine-tuning training process of RoboBrain 2.0, input lengths vary significantly across samples. PyTorch's default caching memory allocator [49] can lead to memory fragmentation under such dynamic input conditions, frequently resulting in OOM errors. A common but inefficient workaround is to call `torch.cuda.empty_cache()` before every forward pass, which severely degrades performance.Instead, we take a more efficient approach by analyzing PyTorch's memory allocation mechanism. Fragmentation often results from the lack of a sufficiently large and contiguous cached memory block for new tensors, prompting new allocations and worsening fragmentation. To address this, we introduce a memory pre-allocation strategy: we compute the maximum sequence length across the entire dataset before training, and pad all samples to this maximum length in the first step. This ensures that tensors can reuse pre-allocated memory blocks, reducing fragmentation and maintaining throughput.

5.1.3 Data Pre-Processing

We adopt native Megatron-Energon [30] for unified data loading, eliminating the need for external training frameworks. Additionally, we optimized the preprocessing pipeline to reduce time consumption by up to 90%. We evaluated and compared two preprocessing strategies:

- **Preprocessing Both JSON and Images.** Using the default Megatron-Energon data pipeline, both JSON metadata and images are compressed into binary files for WebDataset. However, this approach suffers from two major issues: (1) Low efficiency: Preprocessing 320,000 samples can take over 2 hours. (2) Inconsistent image readers: Megatron-Energon uses `cv2`, while models such as RoboBrain 2.0 use `PIL`, introducing subtle differences that may affect training performance.

- **Preprocessing JSON Only (Recommended).** In our optimized pipeline, only JSON files are preprocessed, and images are kept in their original form. Image preprocessing is deferred to the TaskEncoder module using the same preprocessor as Qwen2.5-VL. (1) High efficiency: Preprocessing 320,000 samples takes less than 10 minutes. (2) Alignment with model input: Ensures image handling is fully aligned between preprocessing and training, eliminating inconsistency and improving model performance.

5.1.4 Distributed Data Loading

To minimize the I/O burden on compute nodes, we reduce redundant data loading in large-scale distributed training. Unlike single-node setups, GPUs in distributed training systems play different roles depending on the chosen parallel strategy. Data loading typically occurs along the data parallel (DP) dimension, where each DP rank handles a unique data shard. However, in multi-dimensional hybrid parallelism (e.g., DP-PP-TP), only a subset of GPU processes actually need to load data: (1) In each Pipeline Parallel (PP) [42] group, only the first and last stages need to perform data loading. (2) Within Tensor Parallel (TP) [58] groups, only one GPU per group is required to load data, with others receiving data via broadcast. This design significantly reduces redundant I/O operations and improves overall data throughput.

5.1.5 Fault Tolerance

To handle both hardware and software failures during training, we co-designed fault-tolerant mechanisms between our FlagScale [12] training framework and the system platform. Common errors, such as `LostCard`, `KubeNodeNotReady`, are automatically detected and trigger automatic job recovery and restart, ensuring minimal disruption. Furthermore, our custom DataLoader module based on Megatron-Energon supports full data state recovery, allowing seamless resumption from the most recent checkpoint with complete consistency in data loading and sample shuffling states.

5.2 Reinforcement Fine-Tuning Infrastructure

We employ Reinforcement Learning with Verifiable Rewards (RLVR) to enhance RoboBrain 2.0 using VeRL [68], an open-source RL framework specifically designed for post-training LLMs and VLMs. Based on the HybridFlow architecture [56], VeRL features a hybrid-controller model that integrates both a global controller for inter-RL-role dataflow coordination and distributed controllers for intra-RL-role parallel processing. This architecture enables efficient execution of complex post-training workflows while ensuring scalability. VeRL's support for multiple RL algorithms (e.g., GRPO) and seamless LLM integration makes it particularly suitable for RoboBrain 2.0's reinforcement fine-tuning (RFT) requirements. The framework enables high-performance model tuning with minimal overhead through its optimized dataflow management and parallel processing capabilities. Its efficient handling of large-scale training tasks and rigorous reward verification establishes VeRL as an ideal platform for advancing RoboBrain 2.0's capabilities via RLVR.

5.3 Inference Infrastructure

To improve the efficiency of model inference, we adopt FlagScale [12], also a multi-backend inference framework, which can automatically search for the optimal inference engine and configuration parameters based on the performance characteristics of different models on heterogeneous hardware accelerators, thereby effectively reducing inference latency. Given the high sensitivity of embodied AI models to accuracy, we further introduce a mixed-bit quantization strategy [40, 70]. This strategy enhances inference efficiency and resource utilization while maintaining model performance. Specifically, the vision encoder retains full-precision floating-point computation to ensure the accuracy of key feature extraction. In contrast, during the language module, weights are quantized to 8-bit integers, while activations are preserved in 16-bit floating-point format. This mixed-precision approach significantly reduces computational overhead and memory usage with negligible impact on model accuracy. Moreover, the quantization process is minimally invasive to existing inference pipelines and can be flexibly integrated into current systems. In end-to-end embodied tasks, weight-only quantization alone achieves approximately a 30% reduction in inference latency, demonstrating the effectiveness and practicality of the proposed method in real-world deployment scenarios.

6 Evaluation Results

We conducted a comprehensive evaluation of RoboBrain-2.0, focusing on its performance across spatial and temporal reasoning capabilities on embodiment. To ensure consistency and rigor in evaluation, we adopted FlagEvalMM [20], our flexible framework for systematic multimodal model assessment. Evaluations on spatial reasoning benchmarks (e.g., CV-bench [67], Blink [15], Where2Place [77], ShareRobot-Bench [23]), presented in Section 6.1, underscore the model's strengths in embodied spatial reasoning. An in-depth analysis of multi-robot collaboration [61] and long-horizon planning (e.g., EgoPlan2 [9], RoboBench) capabilities is provided in Section 6.2, highlighting the model's advancements in temporal reasoning tasks. Qualitative examples and prompt details are provided in Appendix A and Appendix B, respectively.

6.1 Spatial Reasoning Capability

RoboBrain-32B-2.0 and RoboBrain-7B-2.0 demonstrate exceptional performance across nine spatial reasoning benchmarks: **BLINK**, **CV-Bench**, **EmbSpatial**, **RoboSpatial**, and **RefSpatial-Bench** (Table 2), as well as **SAT**, **VSI-Bench**, **Where2Place**, and **ShareRobot-Bench** (Table 3). Below is a detailed analysis highlighting their state-of-the-art (SOTA) achievements and near-SOTA competitive results.

Table 2 Performance across five spatial reasoning benchmarks. The best results among different models are highlighted in **bold**, while the second-best results are underlined.

Models / Metrics	BLINK			CV-Bench	EmbSpatial	RoboSpatial	RefSpatial-Bench		
	Dep.	Spa.	All ↑	All ↑	All ↑	All ↑	Loc.	Pla.	All ↑
General Baselines									
Gemini-2.5-Pro-preview-05-06 [16]	79.03	84.62	81.83	84.59	**78.74**	59.87	44.58	31.73	38.16
Gemini-2.5-Flash-preview-04-17 [16]	77.42	79.02	78.22	84.03	74.75	54.10	37.50	23.00	30.25
GPT-o4-mini-2025-05-16 [45]	79.03	**88.11**	83.57	85.21	78.29	51.25	15.00	19.58	17.29
GPT-4o-2024-11-20 [22]	72.58	83.22	77.90	78.63	71.92	44.42	8.00	9.55	8.78
Claude-Sonnet-4-2025-05-14 [2]	75.81	80.42	78.12	78.43	64.26	51.26	5.00	10.37	7.69
Qwen2.5-VL-32B-Instruct [50]	77.42	85.31	81.37	81.59	74.45	52.16	16.83	10.60	13.72
Qwen2.5-VL-72B-Instruct [50]	74.19	78.32	76.26	82.68	73.30	48.33	23.50	15.83	19.67
Embodied Baselines									
Cosmos-Reason1-7B [4]	63.71	73.43	68.57	74.71	65.22	38.81	9.84	1.04	5.44
VeBrain-8B [36]	78.23	81.12	79.68	78.57	70.52	42.48	0.03	0.57	0.30
Magma-8B [74]	65.32	66.43	65.88	60.98	64.59	33.71	1.00	8.00	4.50
RoboBrain-7B-1.0 [23]	75.81	78.32	77.07	76.22	68.13	51.53	14.43	5.41	9.92
RoboBrain-7B-2.0	**84.68**	83.22	**83.95**	**85.75**	76.32	54.23	36.00	29.00	32.50
RoboBrain-32B-2.0	79.84	87.41	83.63	83.92	78.57	**72.43**	**54.00**	**54.00**	**54.00**

- **BLINK.** In the BLINK [15] benchmark, models are evaluated on depth perception (Dep.) and spatial relation understanding (Spa.). RoboBrain-7B-2.0 achieves a SOTA average score of **83.95** (Dep.: **84.68**, Spa.: 83.22), outperforming all general baselines, including GPT-o4-mini-2025-05-16 (83.57), Gemini-2.5-Pro-preview-05-06 (81.83), Qwen2.5-VL-32B-Instruct (81.37), Claude-Sonnet-4-2025-05-14 (78.12), GPT-4o-2024-11-20 (77.90), and Qwen2.5-VL-72B-Instruct (76.26), as well as embodied baselines like VeBrain-8B (79.68) and Cosmos-Reason1-7B (68.57). RoboBrain-32B-2.0 follows closely with an average of 83.63 (Dep.: 79.84, Spa.: **87.41**), surpassing all general and embodied baselines except RoboBrain-7B-2.0, demonstrating strong spatial reasoning capabilities.
- **CV-Bench.** The CV-Bench [67] benchmark assesses a model's accuracy in 2D/3D spatial understanding and visual processing. RoboBrain-7B-2.0 secures a SOTA accuracy of **85.75**, slightly ahead of RoboBrain-32B-2.0 (83.92), both outperforming all general baselines, including GPT-o4-mini-2025-05-16 (85.21), Gemini-2.5-Pro-preview-05-06 (84.59), Qwen2.5-VL-72B-Instruct (82.68), Qwen2.5-VL-32B-Instruct (81.59), GPT-4o-2024-11-20 (78.63), and Claude-Sonnet-4-2025-05-14 (78.43), as well as embodied baselines like VeBrain-8B (78.57) and Cosmos-Reason1-7B (74.71).
- **EmbSpatial.** The EmbSpatial [14] benchmark evaluates models on embodied spatial tasks. RoboBrain-32B-2.0 achieves a near SOTA accuracy of **78.57**, slightly less than Gemini-2.5-Pro-preview-05-06 (78.74) and surpassing all other general baselines, including GPT-o4-mini-2025-05-16 (78.29), Qwen2.5-VL-32B-Instruct

Table 3 Performance across four spatial reasoning benchmarks. The best results among different models are highlighted in **bold**, while the second-best results are underlined.

Models / Metrics	SAT	VSI-Bench	Where2Place*			ShareRobot-Bench	
	All ↑	All ↑	Seen	Unseen	All ↑	Afford. ↑	Traj. (DFD ↓)
General Baselines							
Gemini-2.5-Pro-preview-05-06 [16]	79.33	47.81	42.92	41.13	42.38	10.26	0.7666
Gemini-2.5-Flash-preview-04-17 [16]	74.00	**48.83**	31.54	21.73	28.60	2.50	0.9087
GPT-o4-mini-2025-05-16 [45]	82.00	41.96	26.63	26.49	26.59	8.27	0.5726
GPT-4o-2024-11-20 [22]	66.67	43.60	20.28	20.71	20.41	6.00	0.6850
Claude-Sonnet-4-2025-05-14 [2]	75.33	47.02	21.56	35.11	25.63	8.00	0.7591
Qwen2.5-VL-32B-Instruct [50]	80.00	36.07	18.22	32.55	22.52	11.97	0.9222
Qwen2.5-VL-72B-Instruct [50]	58.67	35.51	35.74	49.65	39.92	23.80	0.5034
Embodied Baselines							
Cosmos-Reason1-7B [4]	60.67	25.64	5.07	6.53	5.51	9.98	0.8524
VeBrain-8B [36]	58.00	26.30	12.27	9.17	11.34	3.66	1.1659
Magma-8B [74]	71.33	12.65	9.93	13.14	10.89	–	0.7478
RoboBrain-7B-1.0 [23]	59.33	31.12	54.58	49.45	53.04	10.20	0.6248
RoboBrain-7B-2.0	75.33	36.10	64.33	61.88	63.59	28.05	0.5512
RoboBrain-32B-2.0	**86.67**	42.69	**73.95**	**72.74**	**73.59**	**35.28**	**0.2368**

(74.45), Qwen2.5-VL-72B-Instruct (73.30), GPT-4o-2024-11-20 (71.92), and Claude-Sonnet-4-2025-05-14 (64.26). RoboBrain-7B-2.0 follows with a competitive score of 76.32, outperforming most general baselines and all embodied baselines, indicating strong embodied spatial reasoning.

- **RoboSpatial.** The RoboSpatial [59] benchmark measures spatial reasoning in robot environments, such as object localization and manipulation. RoboBrain-32B-2.0 achieves a clear SOTA score of **72.43**, substantially ahead of general baselines like Gemini-2.5-Pro-preview-05-06 (59.87), Qwen2.5-VL-72B-Instruct (48.33), GPT-o4-mini-2025-05-16 (51.25), and Claude-Sonnet-4-2025-05-14 (51.26). RoboBrain-7B-2.0 scores 54.23, outperforming all general baselines except RoboBrain-32B-2.0, demonstrating significant improvements in spatial reasoning for robotic tasks.
- **RefSpatial-Bench.** The RefSpatial-Bench [81] benchmark evaluates models on spatial referring expressions, requiring precise point predictions under spatial constraints, with metrics for Location (Loc.) and Placement (Pla.) accuracy. RoboBrain-32B-2.0 achieves SOTA scores of **54.00** (Loc.) and **54.00** (Pla.), significantly outperforming all general baselines, including Gemini-2.5-Pro-preview-05-06 (44.58, 31.73), Qwen2.5-VL-72B-Instruct (23.50, 15.83), Qwen2.5-VL-32B-Instruct (16.83, 10.60), GPT-o4-mini-2025-05-16 (15.00, 19.58), GPT-4o-2024-11-20 (8.00, 9.55), and Claude-Sonnet-4-2025-05-14 (5.00, 10.37). RoboBrain-7B-2.0 scores 36.00 (Loc.) and 29.00 (Pla.), outperforming all general baselines except RoboBrain-32B-2.0, showing competitive precision in complex spatial referring tasks.
- **SAT.** The SAT [53] benchmark measures general spatial reasoning abilities across various scenes and tasks. RoboBrain-32B-2.0 achieves a clear SOTA score of **86.67**, significantly outperforming all general baselines, including GPT-o4-mini-2025-05-16 (82.00), Gemini-2.5-Pro-preview-05-06 (79.33), Qwen2.5-VL-72B-Instruct (58.67), and Claude-Sonnet-4-2025-05-14 (75.33). RoboBrain-7B-2.0 achieves 75.33, surpassing most general and embodied baselines, showcasing its strong spatial reasoning capability.
- **VSI-Bench.** The VSI-Bench [75] evaluates visual-spatial integration capabilities. Gemini-2.5-Flash-preview-04-17 achieves the best performance with **48.83**. RoboBrain-32B-2.0 achieves 42.69, outperforming most general and embodied baselines, including GPT-o4-mini-2025-05-16 (41.96) and Qwen2.5-VL-72B-Instruct (35.51). RoboBrain-7B-2.0 reaches 36.10, indicating solid visual-spatial integration skills.
- **Where2Place.** The Where2Place [77] benchmark measures a model's ability to predict object placements in both seen and unseen scenarios under spatial constraints. RoboBrain-32B-2.0 achieves a SOTA average of **73.59** (Seen: **73.95**, Unseen: **72.74**), substantially surpassing all general and embodied baselines, including Qwen2.5-VL-72B-Instruct (39.92), Gemini-2.5-Pro-preview-05-06 (42.38), Claude-Sonnet-4-2025-05-14 (25.63), and VeBrain-8B (11.34). RoboBrain-7B-2.0 also performs strongly with an average of 63.59 (Seen:

*Upon inspection, we found that the test set included several incorrect cases, which were manually screened and excluded.

64.33, Unseen: 61.88), outperforming all baselines except RoboBrain-32B-2.0.

- **ShareRobot-Bench-Affordance.** The ShareRobot Affordance task [23] evaluates models on object functionality and interaction understanding. RoboBrain-32B-2.0 secures a SOTA performance with an accuracy of **35.28**, ahead of all general baselines, including Qwen2.5-VL-72B-Instruct (23.80), Qwen2.5-VL-32B-Instruct (11.97), GPT-4o-2024-11-20 (6.00), and Claude-Sonnet-4-2025-05-14 (8.00). RoboBrain-7B-2.0 achieves 28.05, outperforming all general and embodied baselines except RoboBrain-32B-2.0.
- **ShareRobot-Bench-Trajectory.** The ShareRobot Trajectory task [23] assesses navigation and motion prediction, using Dynamic Fréchet Distance (DFD), where lower values denote better performance. RoboBrain-32B-2.0 achieves a SOTA DFD of **0.2368**, outperforming all general and embodied baselines, including Qwen2.5-VL-72B-Instruct (0.5034), GPT-o4-mini-2025-05-16 (0.5726), and Gemini-2.5-Pro-preview-05-06 (0.7666). RoboBrain-7B-2.0 follows with a competitive DFD of 0.5512, demonstrating strong path-planning capabilities.

6.2 Temporal Reasoning Capability

RoboBrain-32B-2.0 and RoboBrain-7B-2.0 exhibit outstanding performance across three critical measures of temporal reasoning benchmarks: **Multi-Robot Planning**, **Ego-Plan2**, and **RoboBench**, as shown in Table 4. Below is a detailed analysis highlighting their state-of-the-art (SOTA) achievements and near-SOTA results.

Table 4 Performance across three temporal reasoning benchmarks. The best results among different models are highlighted in **bold**, while the second-best results are underlined.

Models / Metrics	Multi-Robot Planning				Ego-Plan2					RoboBench
	Super.	Rest.	House.	All ↑	Daily.	Hobbies.	Rec.	Work.	All ↑	Plan. ↑
General Baselines										
Gemini-2.5-Pro-preview-05-06 [16]	63.51	54.77	78.39	65.39	44.19	43.05	46.45	39.60	42.85	63.49
Gemini-2.5-Flash-preview-04-17 [16]	59.44	55.78	76.88	63.86	38.72	35.59	43.72	33.42	37.09	69.33
GPT-o4-mini-2025-05-16 [45]	63.32	55.28	78.89	65.50	47.61	35.93	42.62	37.13	41.11	70.01
GPT-4o-2024-11-20 [22]	77.89	67.34	79.40	74.50	47.38	40.00	44.81	35.64	41.79	68.60
Claude-Sonnet-4-2025-05-14 [2]	73.08	61.81	80.40	71.30	43.51	41.02	42.62	38.87	41.26	70.21
Qwen2.5-VL-32B-Instruct [50]	67.84	61.81	75.38	68.00	**64.46**	51.53	57.92	50.00	56.25	45.92
Qwen2.5-VL-72B-Instruct [50]	77.39	68.34	79.40	74.67	60.36	48.14	**63.39**	46.29	53.75	66.94
Embodied Baselines										
Cosmos-Reason1-7B [4]	35.17	25.62	40.70	33.66	30.75	27.12	31.69	20.30	26.87	53.17
VeBrain-8B [36]	41.70	35.67	39.69	38.83	31.79	35.31	31.19	34.43	27.30	46.77
Magma-8B [74]	–	–	–	–	4.56	3.39	6.56	2.97	4.09	–
RoboBrain-7B-1.0 [23]	4.52	7.04	5.03	5.50	–	–	–	–	–	38.93
RoboBrain-7B-2.0	83.92	**77.39**	84.42	**81.50**	39.41	32.20	33.88	26.98	33.23	**72.16**
RoboBrain-32B-2.0	**84.42**	72.36	**85.43**	80.33	64.01	**53.22**	57.92	**52.48**	**57.23**	68.33

- **Multi-Robot Planning.** In the Multi-Robot Planning task [61], models are evaluated on their ability to coordinate multiple robots across different scenarios: Super (Supermarket), Rest (Restaurant), and House (Household). RoboBrain-32B-2.0 achieves a SOTA average score of **80.33** (Super: **84.42**, Rest: 72.36, House: **85.43**), significantly outperforming all general baselines, including GPT-4o-2024-11-20 (74.50), Qwen2.5-VL-72B-Instruct (74.67), Claude-Sonnet-4-2025-05-14 (71.30), Gemini-2.5-Pro-preview-05-06 (65.39), and Qwen2.5-VL-32B-Instruct (68.00). It also surpasses the embodied baseline RoboBrain-7B-2.0 (81.50). RoboBrain-7B-2.0 follows closely with an average of 81.50 (Super: 83.92, Rest: 77.39, House: 84.42), outperforming all general baselines and matching the performance of RoboBrain-7B-1.5-OS in Rest and House scenarios.
- **Ego-Plan2.** The Ego-Plan2 [9] benchmark assesses a model's capability to plan daily activities across four categories: Daily (Daily Routines), Hobbies, Rec (Recreation), and Work. RoboBrain-32B-2.0 secures a SOTA average score of **57.23** (Daily: **64.01**, Hobbies: **53.22**, Rec: **57.92**, Work: **52.48**), significantly outperforming all general and embodied baselines, including Qwen2.5-VL-32B-Instruct (56.25), Qwen2.5-VL-72B-Instruct (53.75), Gemini-2.5-Pro-preview-05-06 (42.85), GPT-4o-2024-11-20 (41.79), Claude-Sonnet-4-2025-05-14 (41.26), GPT-o4-mini-2025-05-16 (41.11), VeBrain-8B (27.30), and Cosmos-Reason1-7B (26.87). In contrast, RoboBrain-7B-2.0 achieves an average of 33.23 (Daily: 39.41, Hobbies: 32.20,

Rec: 33.88, Work: 26.98), which is lower than general baselines like Qwen2.5-VL-32B-Instruct and Qwen2.5-VL-72B-Instruct but surpasses embodied baselines such as VeBrain-8B and Cosmos-Reason1-7B.

- **RoboBench.** The RoboBench Benchmark (Planning part) evaluates a model's ability to plan robotic mobile manipulation tasks according to their pre-defined skills across three categories: cross-embodiment, cross-object, and cross-view. On this benchmark, RoboBrain-7B-2.0 achieves a state-of-the-art (SOTA) score of **72.16**, surpassing all general and embodied baselines, including Claude-Sonnet-4-2025-05-14 (70.21), GPT-o4-mini-2025-05-16 (70.01). The performance of RoboBrain-32B-2.0, with a score of 68.33, outperforming several general baselines like GPT-4o-2024-11-20 (68.60) and Qwen2.5-VL-72B-Instruct (66.94), as well as other embodied baselines such as Cosmos-Reason1-7B (53.17) and VeBrain-8B (46.77).

7 Conclusion and Future Works

In this report, we introduced RoboBrain 2.0, our latest generation of embodied vision-language foundation models, developed to support unified perception, reasoning, and planning in complex physical environments. Built on a modular architecture with a dedicated vision encoder and a decoder-only language model, RoboBrain 2.0 enables high-resolution image and video comprehension, as well as spatial and temporal reasoning. Through a progressive three-stage training strategy—encompassing foundational spatiotemporal learning, embodied enhancement, and chain-of-thought reasoning—the model demonstrates strong generalization across a wide variety of challenging embodied tasks. Despite its compact size, RoboBrain 2.0 achieves state-of-the-art results on most of public embodied spatial and temporal reasoning benchmarks, outperforming both open-source and proprietary models in spatial understanding, closed-loop interaction, and long-horizon planning. Its capabilities span a broad spectrum of embodied scenarios, including affordance prediction, spatial referring, trajectory forecasting, multi-agent coordination, and scene graph construction and updating.

We regard RoboBrain 2.0 as a solid foundation toward developing more general embodied AI, emphasizing the importance of tightly integrated perception, reasoning, and planning. Moving forward, we plan to expand RoboBrain 2.0 along two key directions:

- **Embodied VLM-powered VLA:** We aim to integrate cutting-edge embodied VLMs into the Vision-Language-Action (VLA) framework. By harnessing the powerful spatiotemporal perception and high-level reasoning capabilities of VLMs, this direction seeks to substantially enhance the generality and robustness of action generation. The resulting system will support more nuanced understanding and precise execution of complex, open-ended instructions in real-world scenarios.
- **System-Level Integration:** To improve RoboBrain 2.0's practical utility, we will pursue tight integration with advanced robotics platforms and operating systems. This will enable serverless deployment, adaptation-free skill registration, and low-latency real-time control. In parallel, we envision building a collaborative embodied AI ecosystem—an "intelligence app store"—that supports plug-and-play components for perception, reasoning, and control in real-world robotic systems.

We release RoboBrain 2.0 at https://superrobobrain.github.io, including model checkpoints, training recipes, and evaluation tools, to support broader research and downstream applications in embodied AI. We hope this work bridges the gap between vision-language intelligence and real-world physical interaction.

References

[1] Thomas Breuel Alex Aizman, Gavin Maltby. Webdataset: High-performance data loading for deep learning, 2020. URL https://webdataset.github.io/webdataset/.

[2] Anthropic. Claude sonnet 4. 2025.

[3] Daichi Azuma, Taiki Miyanishi, Shuhei Kurita, and Motoaki Kawanabe. Scanqa: 3d question answering for spatial scene understanding. In CVPR, pages 19129–19139, 2022.

[4] Alisson Azzolini, Hannah Brandon, Prithvijit Chattopadhyay, Huayu Chen, Jinju Chu, Yin Cui, Jenna Diamond, Yifan Ding, Francesco Ferroni, Rama Govindaraju, et al. Cosmos-reason1: From physical common sense to embodied reasoning. arXiv preprint arXiv:2503.15558, 2025.

[5] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. arXiv preprint arXiv:2502.13923, 2025.

[6] Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xu Huang, Shu Jiang, et al. Agibot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. arXiv preprint arXiv:2503.06669, 2025.

[7] Lin Chen, Xilin Wei, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Zhenyu Tang, Li Yuan, et al. Sharegpt4video: Improving video understanding and generation with better captions. Advances in Neural Information Processing Systems, 37:19472–19495, 2024.

[8] Tianqi et al. Chen. Gradient checkpointing in pytorch, 2018. URL https://pytorch.org/docs/stable/checkpoint.html.

[9] Yi Chen, Yuying Ge, Yixiao Ge, Mingyu Ding, Bohao Li, Rui Wang, Ruifeng Xu, Ying Shan, and Xihui Liu. Egoplan-bench: Benchmarking multimodal large language models for human-level planning, 2024. URL https://arxiv.org/abs/2312.06722.

[10] Wei-Lin Chiang, Zhuohan Xu, Hao Zhao, Shuyang Zhuang, Zi Lin Li, Yonghao Lin, Isaac Safo, Eric Singh, Rishi Taori, Noah Shinn, et al. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, 2023. URL https://arxiv.org/abs/2306.05685.

[11] Zhendong Chu, Shen Wang, Jian Xie, Tinghui Zhu, Yibo Yan, Jinheng Ye, Aoxiao Zhong, Xuming Hu, Jing Liang, Philip S Yu, et al. Llm agents for education: Advances and applications. arXiv preprint arXiv:2503.11733, 2025.

[12] FlagScale Contributors. Flagscale: A unified meta-framework enabling adaptive heterogeneous computing for the llm ecosystem. https://github.com/FlagOpen/FlagScale, 2024. Accessed: 2025-06-26.

[13] Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, et al. Molmo and pixmo: Open weights and open data for state-of-the-art multimodal models. arXiv preprint arXiv:2409.17146, 2024.

[14] Mengfei Du, Binhao Wu, Zejun Li, Xuan-Jing Huang, and Zhongyu Wei. Embspatial-bench: Benchmarking spatial understanding for embodied tasks with large vision-language models. In ACL, 2024.

[15] Xingyu Fu, Yushi Hu, Bangzheng Li, Yu Feng, Haoyu Wang, Xudong Lin, Dan Roth, Noah A Smith, Wei-Chiu Ma, and Ranjay Krishna. Blink: Multimodal large language models can see but not perceive. In ECCV, 2024.

[16] Google. Gemini 2.5 pro preview: even better coding performance. https://developers.googleblog.com/en/gemini-2-5-pro-io-improved-coding-performance/, 2025. Accessed: 2025-05-06.

[17] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. arXiv preprint arXiv:2501.12948, 2025.

[18] Dong Guo, Faming Wu, Feida Zhu, Fuxing Leng, Guang Shi, Haobin Chen, Haoqi Fan, Jian Wang, Jianyu Jiang, Jiawei Wang, et al. Seed1. 5-vl technical report. arXiv preprint arXiv:2505.07062, 2025.

[19] Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 5356–5364, 2019.

[20] Zheqi He, Yesheng Liu, Jing shu Zheng, Xuejing Li, Jin-Ge Yao, Bowen Qin, Richeng Xuan, and Xi Yang. Flagevalmm: A flexible framework for comprehensive multimodal model evaluation. 2025. URL https://arxiv.org/abs/2506.09081.

[21] Yining Huang, Keke Tang, Meilian Chen, and Boyuan Wang. A comprehensive survey on evaluating large language model applications in the medical industry. arXiv preprint arXiv:2404.15777, 2024.

[22] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. arXiv preprint arXiv:2410.21276, 2024.

[23] Yuheng Ji, Huajie Tan, Jiayu Shi, Xiaoshuai Hao, Yuan Zhang, Hengyuan Zhang, Pengwei Wang, Mengdi Zhao, Yao Mu, Pengju An, et al. Robobrain: A unified brain model for robotic manipulation from abstract to concrete. In Proceedings of the Computer Vision and Pattern Recognition Conference, pages 1724–1734, 2025.

[24] Bahjat Kawar, Shiran Zada, Oran Lang, Omer Tov, Huiwen Chang, Tali Dekel, Inbar Mosseri, and Michal Irani. Imagic: Text-based real image editing with diffusion models. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 6007–6017, 2023.

[25] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, et al. Ai2-thor: An interactive 3d environment for visual ai. arXiv preprint arXiv:1712.05474, 2017.

[26] Vijay Korthikanti, Jared Casper, Sangkug Lym, Lawrence McAfee, Michael Andersch, Mohammad Shoeybi, and Bryan Catanzaro. Reducing activation recomputation in large transformer models, 2022. URL https://arxiv.org/abs/2205.05198.

[27] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. International journal of computer vision, 123:32–73, 2017.

[28] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. IJCV, 2020.

[29] Justin Lazarow, David Griffiths, Gefen Kohavi, Francisco Crespo, and Afshin Dehghan. Cubify anything: Scaling indoor 3d object detection. arXiv preprint arXiv:2412.04458, 2024.

[30] Xuechen Li, Yifan Mai, Percy Liang, and Matei Zaharia. Energon: Scaling megatron-lm training with data and expert parallelism, 2023. URL https://github.com/HazyResearch/megatron-energon.

[31] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. arXiv preprint arXiv:2412.19437, 2024.

[32] Fuxiao Liu, Kevin Lin, Linjie Li, Jianfeng Wang, Yaser Yacoob, and Lijuan Wang. Mitigating hallucination in large multi-modal models via robust instruction tuning. arXiv preprint arXiv:2306.14565, 2023.

[33] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 26296–26306, 2024.

[34] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In ECCV, 2024.

[35] Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. Advances in Neural Information Processing Systems, 35:2507–2521, 2022.

[36] Gen Luo, Ganlin Yang, Ziyang Gong, Guanzhou Chen, Haonan Duan, Erfei Cui, Ronglei Tong, Zhi Hou, Tianyi Zhang, Zhe Chen, et al. Visual embodied brain: Let multimodal large language models see, think, and control in spaces. arXiv preprint arXiv:2506.00123, 2025.

[37] Run Luo, Lu Wang, Wanwei He, and Xiaobo Xia. Gui-r1: A generalist r1-style vision-language action model for gui agents. arXiv preprint arXiv:2504.10458, 2025.

[38] Ruiyuan Lyu, Tai Wang, Jingli Lin, Shuai Yang, Xiaohan Mao, Yilun Chen, Runsen Xu, Haifeng Huang, Chenming Zhu, Dahua Lin, and Jiangmiao Pang. Mmscan: A multi-modal 3d scene dataset with hierarchical grounded language annotations. arXiv preprint arXiv:2406.09401, 2024.

[39] Xiaojian Ma, Silong Yong, Zilong Zheng, Qing Li, Yitao Liang, Song-Chun Zhu, and Siyuan Huang. Sqa3d: Situated question answering in 3d scenes. In ICLR, 2023. URL https://openreview.net/forum?id=IDJx97BC38.

[40] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training, 2018. URL https://arxiv.org/abs/1710.03740.

[41] Anand Mishra, Shashank Shekhar, Ajeet Kumar Singh, and Anirban Chakraborty. Ocr-vqa: Visual question answering by reading text in images. In 2019 international conference on document analysis and recognition (ICDAR), pages 947–952. IEEE, 2019.

[42] Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, et al. Efficient large-scale language model training on gpu clusters using megatron-lm. In Proceedings of the international conference for high performance computing, networking, storage and analysis, pages 1–15, 2021.

[43] NVIDIA. Megatron-lm: Training multi-billion parameter language models using model parallelism, 2021. URL https://github.com/NVIDIA/Megatron-LM.

[44] OpenAI. Gpt-4 technical report, 2023. URL https://doi.org/10.48550/arXiv.2303.08774.

[45] OpenAI. Gpt-4v(ision) system card. https://openai.com/index/introducing-o3-and-o4-mini/, 2025. Accessed: 2025-04-16.

[46] Kun Ouyang. Spatial-r1: Enhancing mllms in video spatial reasoning. arXiv preprint arXiv:2504.01805, 2025.

[47] Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In 2024 IEEE International Conference on Robotics and Automation (ICRA), pages 6892–6903. IEEE, 2024.

[48] Luigi Piccinelli, Christos Sakaridis, Yung-Hsu Yang, Mattia Segu, Siyuan Li, Wim Abbeloos, and Luc Van Gool. Unidepthv2: Universal monocular metric depth estimation made simpler. arXiv, 2025.

[49] PyTorch Developers. Cuda memory management, 2023. URL https://pytorch.org/docs/stable/notes/cuda.html#cuda-memory-management.

[50] Qwen Team. Qwen2.5-vl: Multimodal llms from alibaba, 2025. URL https://github.com/QwenLM/Qwen2.5-VL.

[51] Vignesh Ramanathan, Anmol Kalia, Vladan Petrovic, Yi Wen, Baixue Zheng, Baishan Guo, Rui Wang, Aaron Marquez, Rama Kovvuri, Abhishek Kadian, et al. Paco: Parts and attributes of common objects. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 7141–7151, 2023.

[52] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. ICLR, 2025.

[53] Arijit Ray, Jiafei Duan, Reuben Tan, Dina Bashkirova, Rose Hendrix, Kiana Ehsani, Aniruddha Kembhavi, Bryan A Plummer, Ranjay Krishna, Kuo-Hao Zeng, et al. Sat: Spatial aptitude training for multimodal language models. arXiv preprint arXiv:2412.07755, 2024.

[54] Dustin Schwenk, Apoorv Khandelwal, Christopher Clark, Kenneth Marino, and Roozbeh Mottaghi. A-okvqa: A benchmark for visual question answering using world knowledge. In European conference on computer vision, pages 146–162. Springer, 2022.

[55] Pierre Sermanet, Tianli Ding, Jeffrey Zhao, Fei Xia, Debidatta Dwibedi, Keerthana Gopalakrishnan, Christine Chan, Gabriel Dulac-Arnold, Sharath Maddineni, Nikhil J Joshi, et al. Robovqa: Multimodal long-horizon reasoning for robotics. In 2024 IEEE International Conference on Robotics and Automation (ICRA), pages 645–652. IEEE, 2024.

[56] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In Proceedings of the Twentieth European

Conference on Computer Systems, EuroSys '25, page 1279–1297. ACM, March 2025. doi: 10.1145/3689031.3696075. URL http://dx.doi.org/10.1145/3689031.3696075.

[57] Shelly Sheynin, Adam Polyak, Uriel Singer, Yuval Kirstain, Amit Zohar, Oron Ashual, Devi Parikh, and Yaniv Taigman. Emu edit: Precise image editing via recognition and generation tasks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 8871–8879, 2024.

[58] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. arXiv preprint arXiv:1909.08053, 2019.

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

[60] Alon Talmor, Ori Yoran, Amnon Catav, Dan Lahav, Yizhong Wang, Akari Asai, Gabriel Ilharco, Hannaneh Hajishirzi, and Jonathan Berant. Multimodalqa: Complex question answering over text, tables and images. arXiv preprint arXiv:2104.06039, 2021.

[61] Huajie Tan, Xiaoshuai Hao, Minglan Lin, Pengwei Wang, Yaoxu Lyu, Mingyu Cao, Zhongyuan Wang, and Shanghang Zhang. Roboos: A hierarchical embodied framework for cross-embodiment and multi-agent collaboration. arXiv preprint arXiv:2505.03673, 2025.

[62] Huajie Tan, Yuheng Ji, Xiaoshuai Hao, Minglan Lin, Pengwei Wang, Zhongyuan Wang, and Shanghang Zhang. Reason-rft: Reinforcement fine-tuning for visual reasoning. arXiv preprint arXiv:2503.20752, 2025.

[63] Yunlong Tang, Jing Bi, Siting Xu, Luchuan Song, Susan Liang, Teng Wang, Daoan Zhang, Jie An, Jingyang Lin, Rongyi Zhu, et al. Video understanding with large language models: A survey. IEEE Transactions on Circuits and Systems for Video Technology, 2025.

[64] Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza, Michiel Blokzijl, et al. Gemini robotics: Bringing ai into the physical world. arXiv preprint arXiv:2503.20020, 2025.

[65] Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. arXiv preprint arXiv:2501.12599, 2025.

[66] Qwen Team. Qwq-32b: Embracing the power of reinforcement learning, March 2025. URL https://qwenlm.github.io/blog/qwq-32b/.

[67] Peter Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Adithya Jairam Vedagiri IYER, Sai Charitha Akula, Shusheng Yang, Jihan Yang, Manoj Middepogu, Ziteng Wang, et al. Cambrian-1: A fully open, vision-centric exploration of multimodal llms. NeurIPS, 2024.

[68] volcengine. verl: Volcano engine reinforcement learning for llms, 2024. URL https://github.com/volcengine/verl.

[69] Johanna Wald, Armen Avetisyan, Nassir Navab, Federico Tombari, and Matthias Nießner. Rio: 3d object instance re-localization in changing indoor environments. In ICCV, pages 7658–7667, 2019.

[70] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision, 2019. URL https://arxiv.org/abs/1811.08886.

[71] Shuai Wang, Weiwen Liu, Jingxuan Chen, Yuqi Zhou, Weinan Gan, Xingshan Zeng, Yuhan Che, Shuai Yu, Xinlong Hao, Kun Shao, et al. Gui agents with foundation models: A comprehensive survey. arXiv preprint arXiv:2411.04890, 2024.

[72] Yi Wang, Kunchang Li, Xinhao Li, Jiashuo Yu, Yinan He, Guo Chen, Baoqi Pei, Rongkun Zheng, Zun Wang, Yansong Shi, et al. Internvideo2: Scaling foundation models for multimodal video understanding. In European Conference on Computer Vision, pages 396–416. Springer, 2024.

[73] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. arXiv preprint arXiv:2505.09388, 2025.

[74] Jianwei Yang, Reuben Tan, Qianhui Wu, Ruijie Zheng, Baolin Peng, Yongyuan Liang, Yu Gu, Mu Cai, Seonghyeon Ye, Joel Jang, et al. Magma: A foundation model for multimodal ai agents. In Proceedings of the Computer Vision and Pattern Recognition Conference, pages 14203–14214, 2025.

[75] Jihan Yang, Shusheng Yang, Anjali W Gupta, Rilyn Han, Li Fei-Fei, and Saining Xie. Thinking in space: How multimodal large language models see, remember, and recall spaces. In Proceedings of the Computer Vision and Pattern Recognition Conference, pages 10632–10643, 2025.

[76] Licheng Yu, Patrick Poirson, Shan Yang, Alexander C Berg, and Tamara L Berg. Modeling context in referring expressions. In Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14, pages 69–85. Springer, 2016.

[77] Wentao Yuan, Jiafei Duan, Valts Blukis, Wilbert Pumacay, Ranjay Krishna, Adithyavairavan Murali, Arsalan Mousavian, and Dieter Fox. Robopoint: A vision-language model for spatial affordance prediction for robotics, 2024. URL https://arxiv.org/abs/2406.10721.

[78] Wenqi Zhang, Mengna Wang, Gangao Liu, Xu Huixin, Yiwei Jiang, Yongliang Shen, Guiyang Hou, Zhe Zheng, Hang Zhang, Xin Li, et al. Embodied-reasoner: Synergizing visual search, reasoning, and action for embodied interactive tasks. arXiv preprint arXiv:2503.21696, 2025.

[79] Youcai Zhang, Xinyu Huang, Jinyu Ma, Zhaoyang Li, Zhaochuan Luo, Yanchun Xie, Yuzhuo Qin, Tong Luo, Yaqian Li, Shilong Liu, et al. Recognize anything: A strong image tagging model. In CVPR, 2024.

[80] Enshen Zhou, Qi Su, Cheng Chi, Zhizheng Zhang, Zhongyuan Wang, Tiejun Huang, Lu Sheng, and He Wang. Code-as-monitor: Constraint-aware visual programming for reactive and proactive robotic failure detection. arXiv preprint arXiv:2412.04455, 2024.

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

[82] Hao Zhou, Chengming Hu, Ye Yuan, Yufei Cui, Yili Jin, Can Chen, Haolun Wu, Dun Yuan, Li Jiang, Di Wu, et al. Large language model (llm) for telecommunications: A comprehensive survey on principles, key techniques, and opportunities. IEEE Communications Surveys & Tutorials, 2024.

[83] Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, et al. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. arXiv preprint arXiv:2504.10479, 2025.

[84] Shengjie Zhu, Abhinav Kumar, Masa Hu, and Xiaoming Liu. Tame a wild camera: in-the-wild monocular camera calibration. NIPS, 2023.

8 Contributions and Author List

Core Contributors

Model Training

- Mingyu Cao*
- Huajie Tan*
- Yuheng Ji*
- Minglan Lin*†
- Zhiyu Li
- Zhou Cao
- Pengwei Wang†

Data & Evaluation

- Enshen Zhou
- Yi Han
- Yingbo Tang
- Xiangqi Xu
- Wei Guo
- Yaoxu Lyu
- Yijie Xu
- Jiayu Shi
- Cheng Chi†
- Mengdi Zhao
- Xiaoshuai Hao

Research Leads

- Yonghua Lin
- Zhongyuan Wang
- Tiejun Huang
- Shanghang Zhang✉

Contributors

Real-Robot Experiments

- Shanyu Rong
- Zhengliang Cai
- Bolun Zhang
- Shuyi Zhang
- Huaihai Lyu
- Mengfei Du
- Lingfeng Zhang

Product & Operations

- Xi Feng
- Xiaodan Liu
- Yance Jiao

Infrastructure

- Chenrui He
- Mengsi Lyu
- Zhuo Chen
- Yulong Ao

Evaluation

- Xue Sun
- Zheqi He
- Jingshu Zheng
- Xi Yang

System Management

- Donghai Shi
- Kunchang Xie
- Bochao Zhang
- Shaokai Nie
- Chunlei Men

* Equal Contribution (Co-first Authors).
† Project Leaders.
✉ Corresponding Author. Team Email: robobrain@baai.ac.cn

Appendix

A Qualitative examples

This section provides a comprehensive set of qualitative examples that illustrate the capabilities of RoboBrain 2.0 in various embodied AI tasks. These examples demonstrate the model's proficiency in spatial reasoning, temporal planning, and interactive reasoning, showcasing its potential for real-world applications.

A.1 Examples for Pointing

In the pointing task, RoboBrain 2.0 is required to identify and point to specific objects within an image based on complex spatial instructions. For instance, given the instruction "Please point out the orange box," the model accurately identifies the orange box in the image. Similarly, for more complex instructions such as "Please point out the brown box on the shelf," RoboBrain 2.0 demonstrates its ability to understand spatial relationships and accurately points to the correct object. The model's proficiency in this task is further exemplified by its performance on a variety of pointing examples, as shown in Figure 5-Figure 20. These examples highlight the model's robust spatial reasoning capabilities, enabling it to handle a wide range of pointing tasks with high precision. Whether the instructions involve simple object identification or more intricate spatial relationships, RoboBrain 2.0 consistently demonstrates its ability to accurately locate and point to the specified objects. This capability is crucial for applications in robotics and automation, where precise object localization is essential for effective interaction with the physical environment.

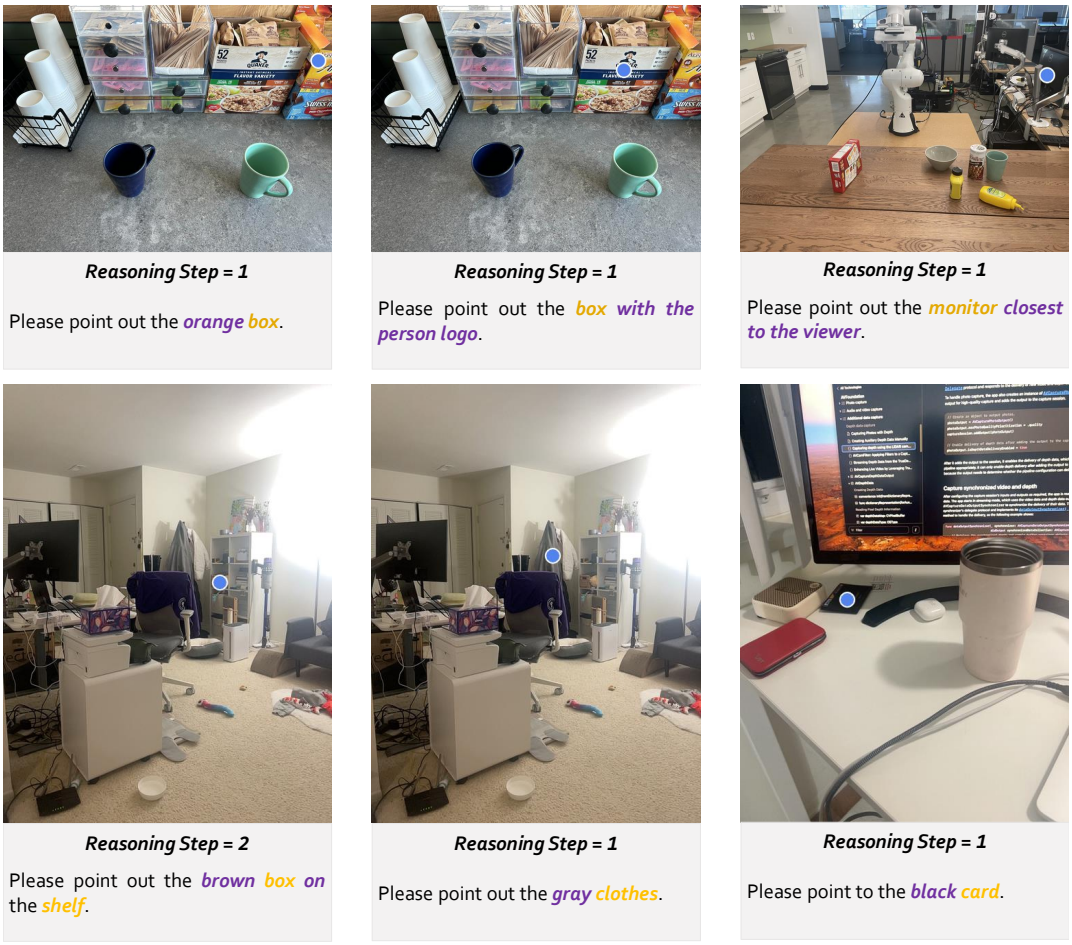
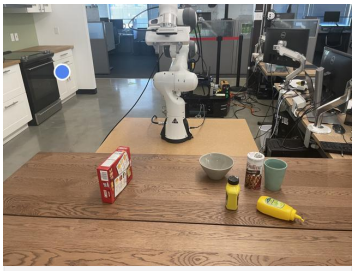


Figure 5 Pointing Examples of RoboBrain 2.0. The blue point represents the model's spatial referring prediction.



Reasoning Step = 1

Please point to the ***farthest white cabinet***.

Reasoning Step = 2

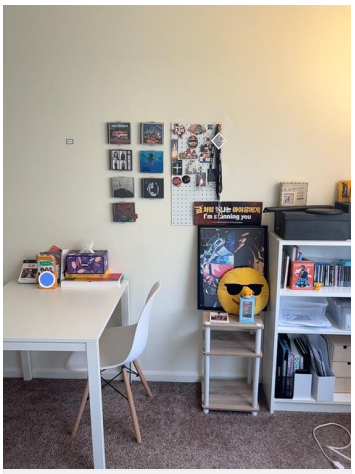
Please point to ***the top piece*** of ***paper*** on the ***white table***.

Reasoning Step = 2

Please point to the ***left pillow*** on the ***sofa***.

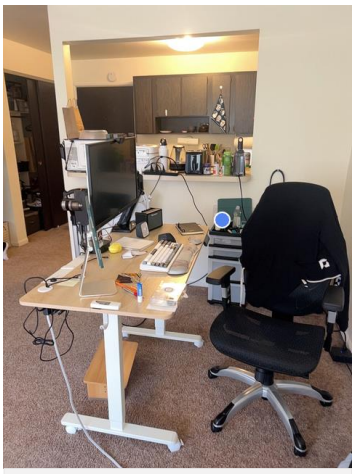
Reasoning Step = 3

Please point out the ***leftmost black object on the same platform*** as the ***micro-wave oven***.



Reasoning Step = 2

Please point out the ***orange box on*** the ***white table on the left***.

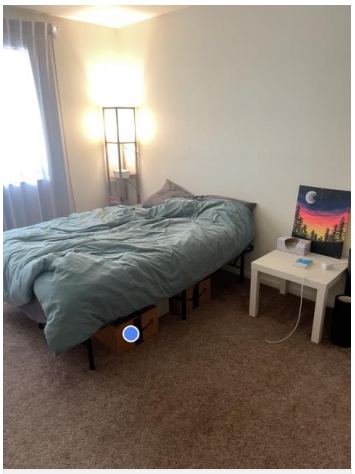


Reasoning Step = 3

Please point out the ***white cup on*** the ***shelf behind*** the ***chair***.

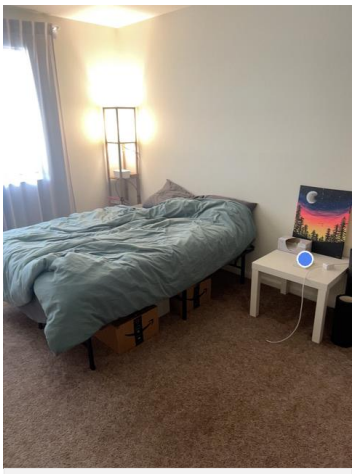
Reasoning Step = 2

Please point to the ***rightmost blue box*** on the ***refrigerator***.



Reasoning Step = 3

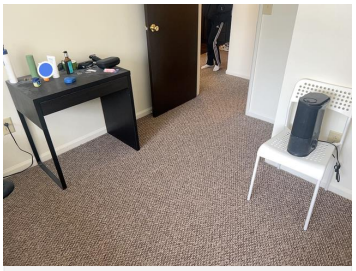
Please point out the ***cardboard box under*** the ***bed*** which is ***closest box to the viewer***.



Reasoning Step = 2

Please point out the ***blue object*** on the ***table***.

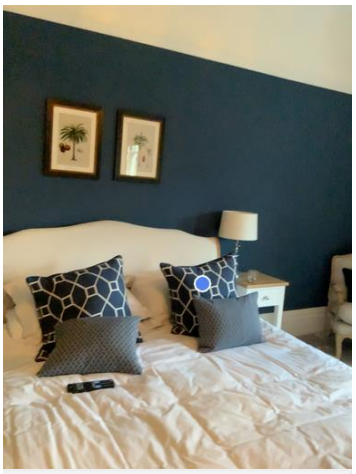
Figure 6 Pointing Examples of RoboBrain 2.0. The blue point represents the model's spatial referring prediction.



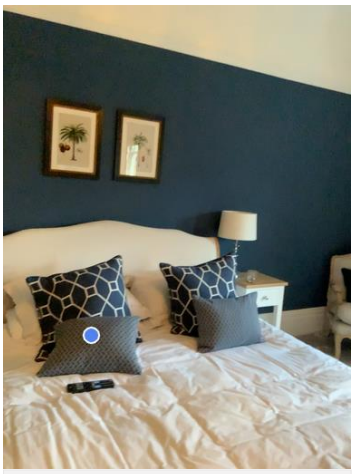
Reasoning Step = 3
Please point to the *white bottle on* the *table* that is *closest to* the *green bottle on the left*.

Reasoning Step = 2
Please point out the *second object from the left to the right on the nearest platform*.

Reasoning Step = 2
Please point out the *object on the right of* the *shovels*.

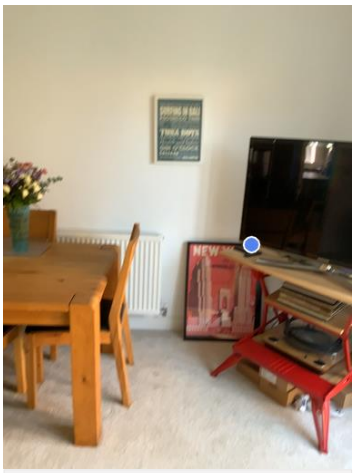


Reasoning Step = 2
Please point to the *pillow closest to the right nightstand*.



Reasoning Step = 2
Please point to the *pillow closest to* the *remote controller*.

Reasoning Step = 3
Please point out the *object between the white box and the farthest black pot*.



Reasoning Step = 3
Please point out the *black object* that is *on the same platform as the TV*.

Reasoning Step = 2
Please point out the *vase closest to* the *TV*.

Reasoning Step = 3
Please point to the *rightmost box at the bottom of* the *shelf*.

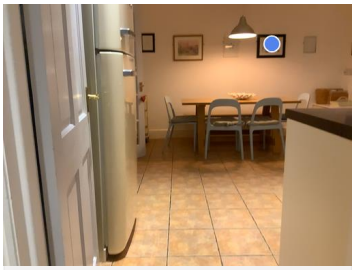
Figure 7 Pointing Examples of RoboBrain 2.0. The blue point represents the model's spatial referring prediction.

Reasoning Step = 1

Please point out the *second silver box from left to right*.

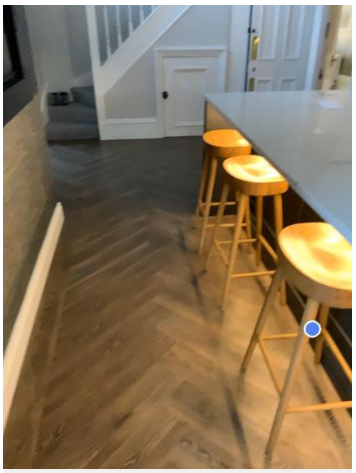
Reasoning Step = 1

Please point to the *wooden plate on the far left*.



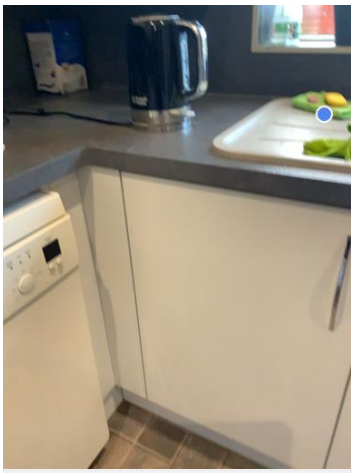
Reasoning Step = 3

Please point out the *black framed painting on the right of* the *lamp*.



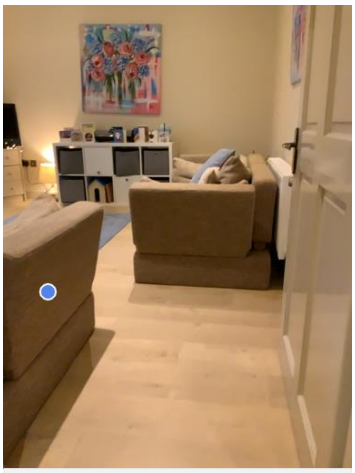
Reasoning Step = 1

Please point out the *chair closest from the viewer*.



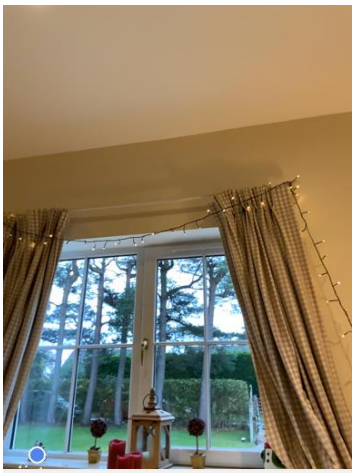
Reasoning Step = 2

Please point out the green *towel on the upper right with yellow object on top*.



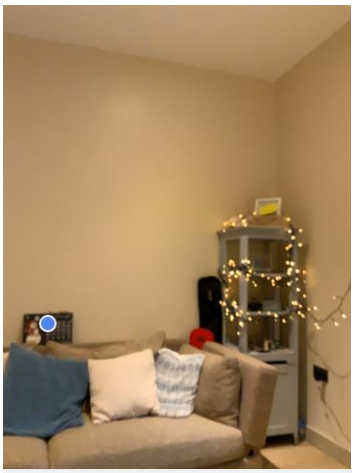
Reasoning Step = 1

Please point out the *brown sofa*, which is *the closest sofa to the viewer*.



Reasoning Step = 2

Please point out the *object on the windowsill farthest from the viewer*.



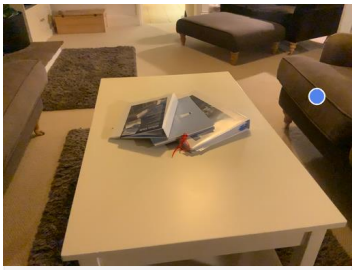
Reasoning Step = 2

Please point out the *black object which is farthest from the shelf*.

Reasoning Step = 1

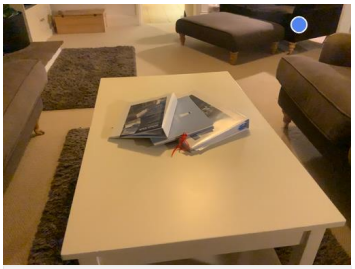
Please point out the *paper tube closest to the viewer*.

Figure 8 Pointing Examples of RoboBrain 2.0. The blue point represents the model's spatial referring prediction.



Reasoning Step = 1

Please point out the *sofa on the right side* that is *closest to the viewer*.



Reasoning Step = 1

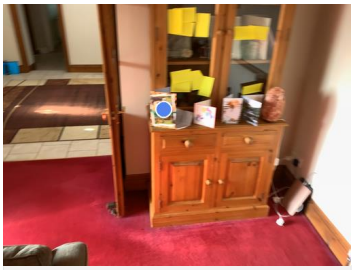
Please point out the *sofa farthest from the viewer*.

Reasoning Step = 2

Please point out the *painting hanging on* the *wall*.

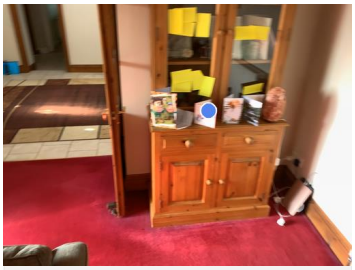
Reasoning Step = 2

Please point out the *blue toothbrush farthest from the faucet*.



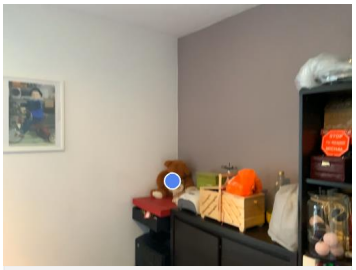
Reasoning Step = 2

Please point out the *card closest to the wooden door*.



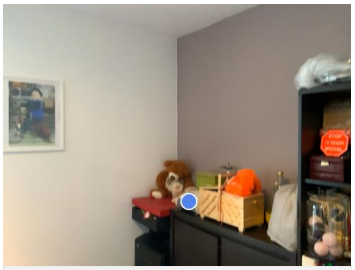
Reasoning Step = 2

Please point to the *third card from right to left on the cabinet*.



Reasoning Step = 1

Please point out the *brown object farthest from the viewer*.



Reasoning Step = 2

Please point out the *white object on* the *cabinet farthest from the viewer*.

Reasoning Step = 3

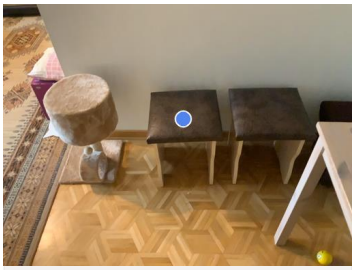
Please point out the *white object adjacent to the left side of* the *picture frame on* the *cabinet*

Reasoning Step = 2

Please point out the *closest red box* to the *blue box*.

Reasoning Step = 1

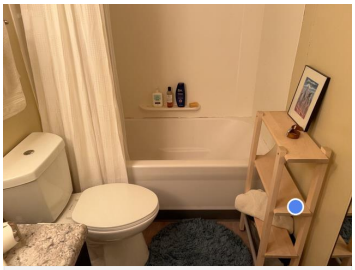
Please point out the *second closest cup to the viewer*.



Reasoning Step = 2

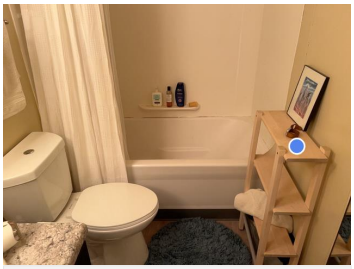
Please point out the *stool* which is *farthest from* the *white table*.

Figure 9 Pointing Examples of RoboBrain 2.0. The blue point represents the model's spatial referring prediction.



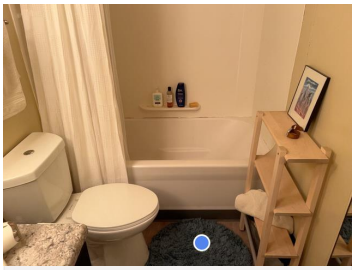
Reasoning Step = 2

Please point out the *free space on the second shelf of* the *wooden shelf*.



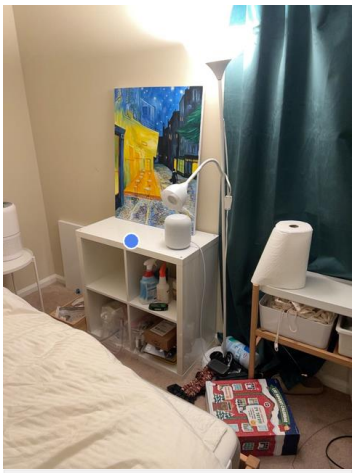
Reasoning Step = 3

Please point out the *free space in front of* the *brown object on* the *shelf*.



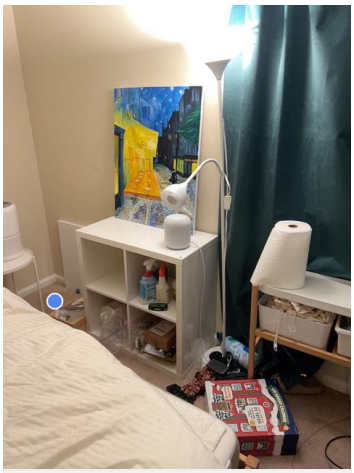
Reasoning Step = 3

Please point out the *free space between toilet* and *shelf*.



Reasoning Step = 2

Please point out the *free space on the white table at center*.

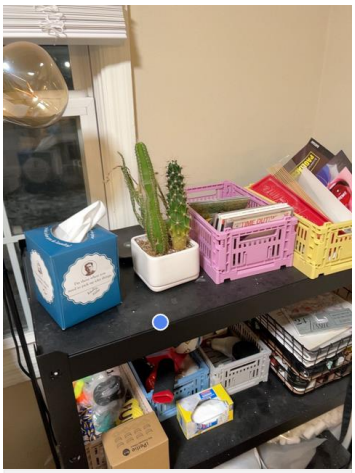


Reasoning Step = 2

Please point out the *empty area to the right of the leftmost stool*.

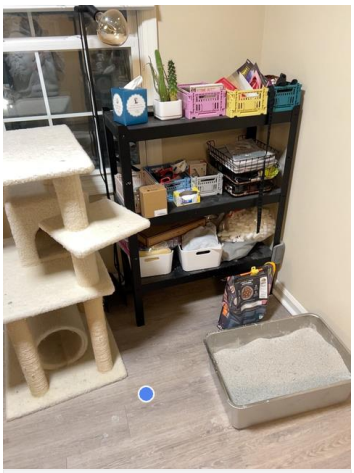
Reasoning Step = 4

Please point out the *free space in front of* the *blue box* which is *on the top of* the *shelf*.



Reasoning Step = 4

Please point out the *free space in front of* the *white vase* which is *on the top of* the *shelf*.



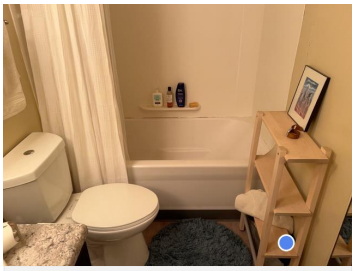
Reasoning Step = 3

Please point out the *free space between* the *cat tree* and *litter box*.

Reasoning Step = 2

Please point out the *free space in front of* the *litter box*.

Figure 10 Pointing Examples of RoboBrain 2.0. The blue point represents the model's spatial referring prediction.



Reasoning Step = 2

Please point out the *free space on the lowest shelf of* the *shelf*.

Reasoning Step = 3

Please point out the *free space between* the *black water bottle* and the *pot lid*.

Reasoning Step = 4

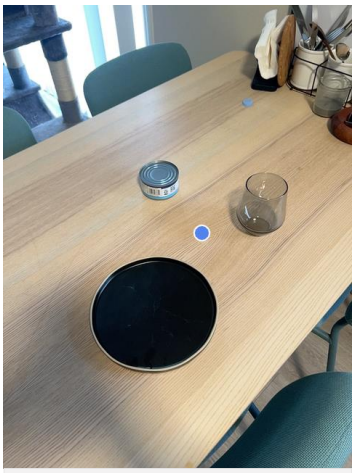
Please point out the *free space between the black water bottle*, the *pot lid*, and the *scissors*.

Reasoning Step = 2

Please point out the *free space on the right of the farthest pot*.

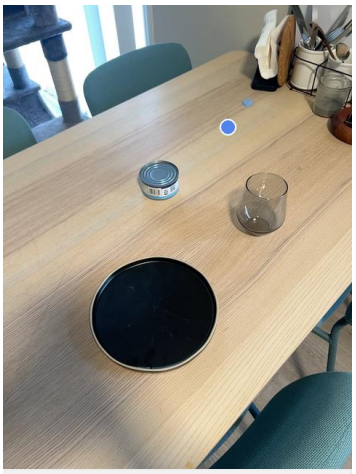
Reasoning Step = 2

Please point out the *free space inside the closest pot*.



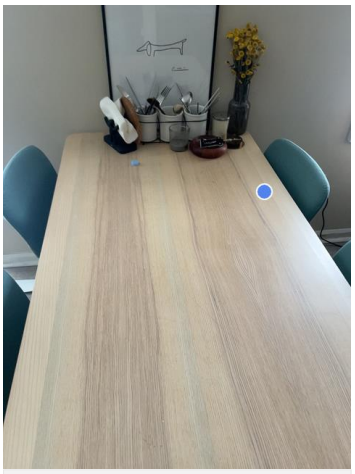
Reasoning Step = 4

Please point out the *free space between* the *black plate*, *blue can and closest water glass*.



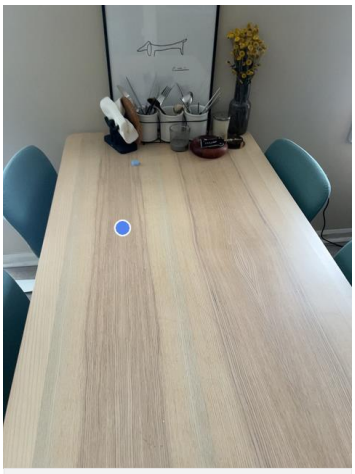
Reasoning Step = 2

Please point out the *free space in the top corner of* the *table*.



Reasoning Step = 3

Please point out the *free area on* the *table in facing direction of the second chair from the front on the right side*.

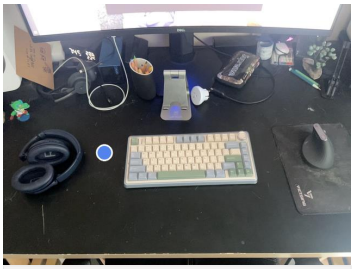


Reasoning Step = 3

Please point out the *free area on* the *table that the second chair from the front on the left side is directly facing*.

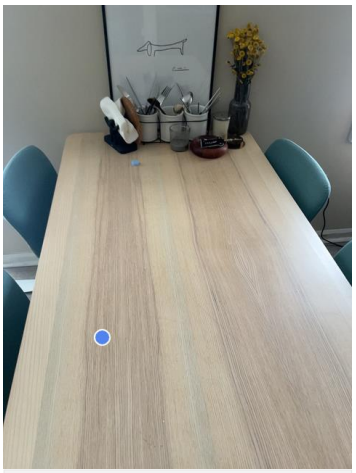
Figure 11 Pointing Examples of RoboBrain 2.0. The blue point represents the model's spatial referring prediction.

Reasoning Step = 3
Please point out the *free space between* the *scissors and* the *microwave*.

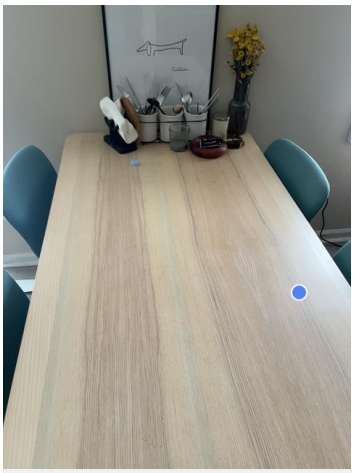


Reasoning Step = 4
Please point out the *free space between* the *headphones farthest from* the *monitor* and the *keyboard*.

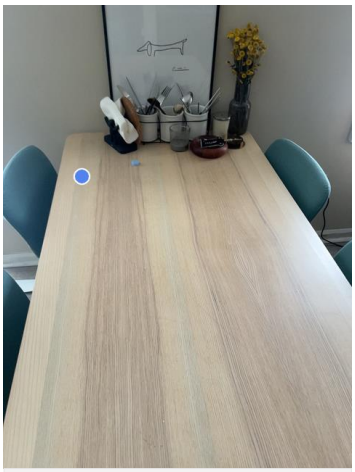
Reasoning Step = 5
Please point out the *free space between* the *black cloth box* to *the bottom-right* of the *monitor* and the *keyboard*.



Reasoning Step = 3
Please point out the *free area* on the *table* that the *first chair from the front on the left side is directly facing*.



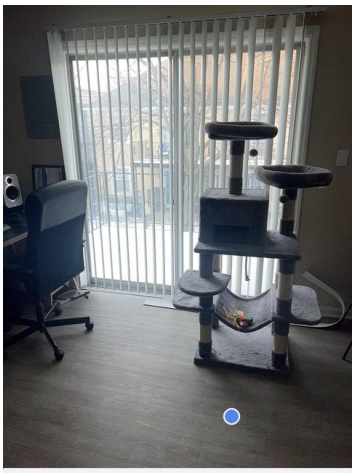
Reasoning Step = 3
Please point out the *free area* on the *table* that the *first chair from the front on the right side is directly facing*.



Reasoning Step = 2
Please point out the *free area in the top-left corner of* the *table*.

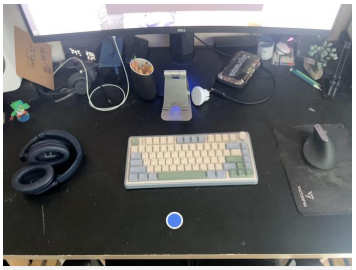
Reasoning Step = 3
Please point out the *free space on* the *stovetop in front of* the *black pot suitable for placing another pot*.

Reasoning Step = 3
Please point out the *free area between* the *black container for spatulas* and the *black object on its right side*.



Reasoning Step = 2
Please point out the *free space in front of* the *cat tree*.

Figure 12 Pointing Examples of RoboBrain 2.0. The blue point represents the model's spatial referring prediction.



Reasoning Step = 4

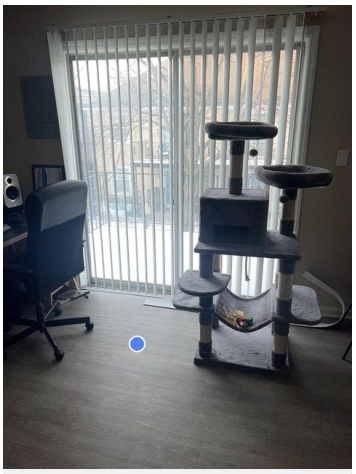
Please point out the *free space on* the *table between* the *keyboard* and the *viewer*.

Reasoning Step = 4

Please point out the *free space on* the *toilet between* the *blue bottle* and the *red can*.

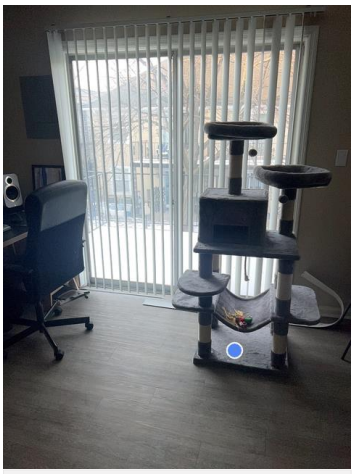
Reasoning Step = 3

Please point out the *free space between* the *bathtub and* the *toilet*.



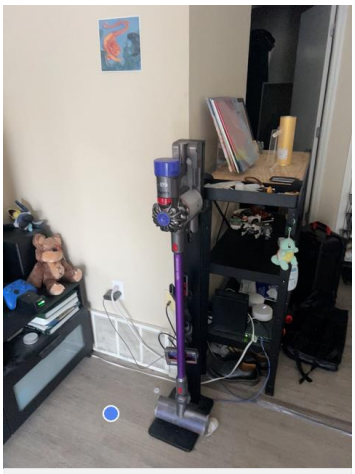
Reasoning Step = 3

Please point out the *free space between* the *cat tree* and the *chair*.



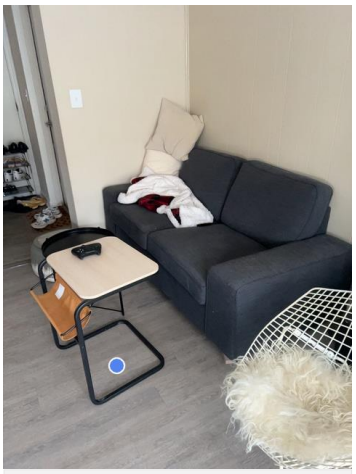
Reasoning Step = 2

Please point out the *free space on the lowest shelf of* the *cat tree*.



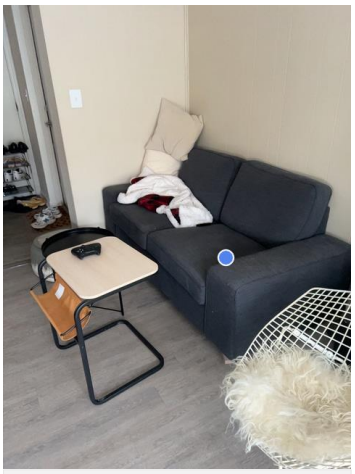
Reasoning Step = 3

Please point out the *free space between* the *purple vacuum cleaner* and the *cabinet on the left*.



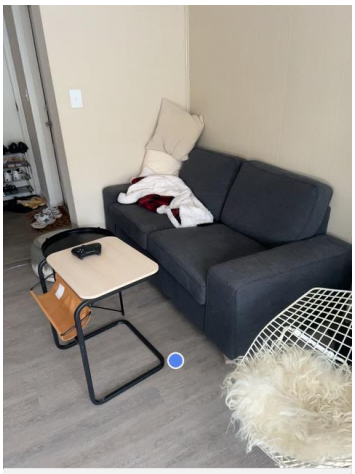
Reasoning Step = 2

Please point out the *free space below* the *table*.



Reasoning Step = 2

Please point out the *free space on* the *sofa cushion*.

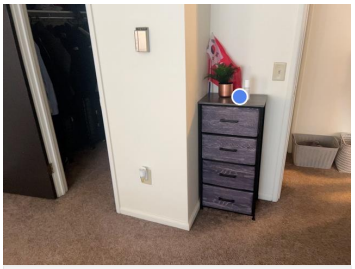


Reasoning Step = 3

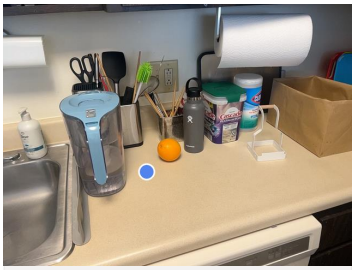
Please point out the *free space between* the *table* and the *sofa*.

Figure 13 Pointing Examples of RoboBrain 2.0. The blue point represents the model's spatial referring prediction.

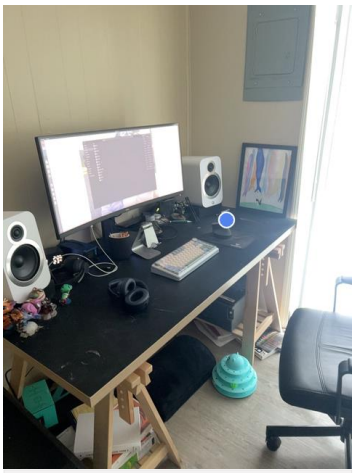
Reasoning Step = 3
Please point out the *free space on* the *cabinet* to *in front of* the *brown vase*.



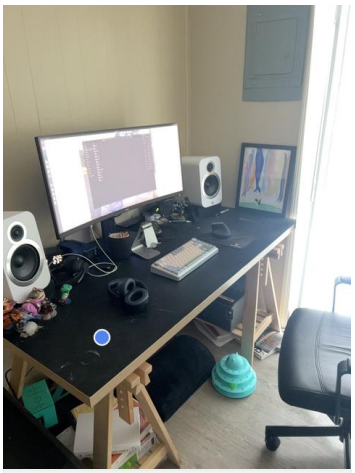
Reasoning Step = 4
Please point out the *free space on* the *cabinet between* the *brown vase* and *white bottle*.



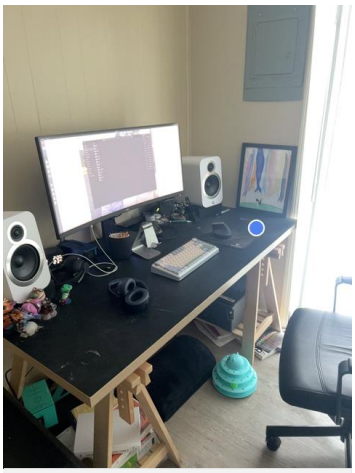
Reasoning Step = 3
Please point out the *free spot between the blue water kettle and* the *orange*.



Reasoning Step = 5
Please point out the *free space* on the *table between* the *speaker* to *the right of* the *monitor and* the *mouse*.



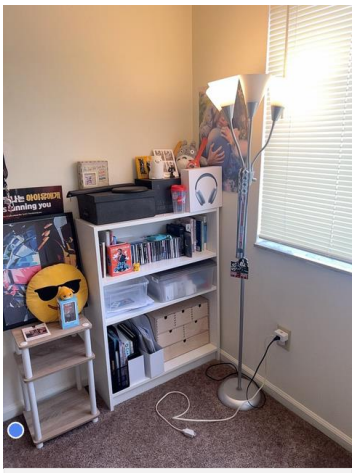
Reasoning Step = 2
Please point out the *free space on the corner of* the *black table* that is *closest to the viewer*.



Reasoning Step = 5
Please point out the *free space on the right part* of the *table between* the *mouse and* the *picture frame*.

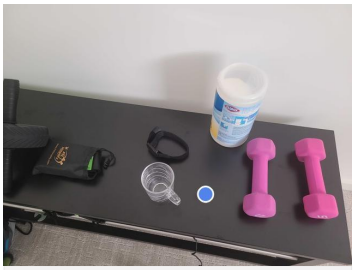
Reasoning Step = 4
Please point out the *free space* on the *table between* the *pillow and* the *brown bowl*.

Reasoning Step = 3
Please point out the *free area on* the *stovetop to the left of* the *pot*.



Reasoning Step = 2
Please point out the *free space on the left of* the *brown shelf*.

Figure 14 Pointing Examples of RoboBrain 2.0. The blue point represents the model's spatial referring prediction.

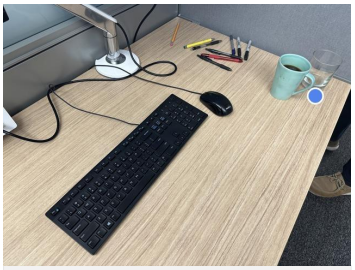


Reasoning Step = 5
Please point out the *free spot on* the *table to the left of* the *two pink dumbbells where another dumbbell can be placed at an equal distance*.

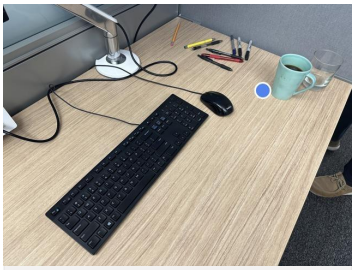
Reasoning Step = 4
Please point out the *free spot*, *equidistant from* both the *blue bowl* and the *red bowl*, and *between* them, *where another bowl can be placed*.

Reasoning Step = 4
Please point out the *free spot behind* the *pink cup*, such that *distance to* the *pink cup* is *equal to distance from* the *pink cup to* the *red bowl*.

Reasoning Step = 4
Please point out the *free space midway between the first and second green cups from the left*.

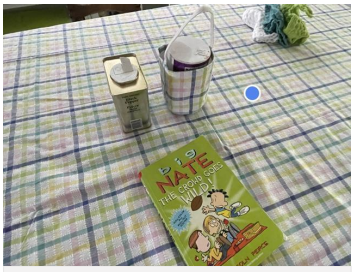


Reasoning Step = 2
Please point out the *free area in the direction of the handle of the rightmost green cup*.



Reasoning Step = 3
Please point out the *free space between* the *mouse and* the *green cup*.

Reasoning Step = 2
Please point out the *free space in the direction of the handle of the second closest cup to the viewer*.



Reasoning Step = 4
Please point out the *free spot to the right of the cloth bag, where an object of the same size can be placed at an equal*.

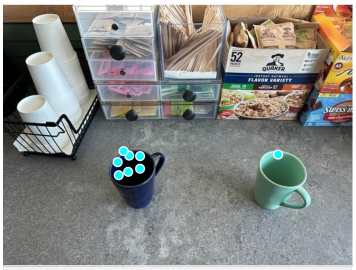
Reasoning Step = 2
Please point out the *free space in the direction of the handle of the transparent glass cup*.

Reasoning Step = 2
Please point out the *free space in the facing direction* of the *purple bag*.

Reasoning Step = 2
Please point out the *free space in the facing direction of the orange box*.

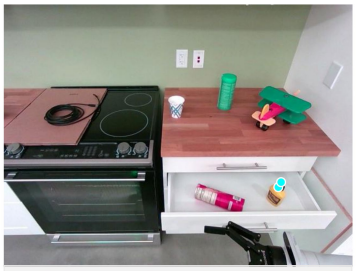
Reasoning Step = 3
Please point out the *free space between the red box on the left and the black box*.

Figure 15 Pointing Examples of RoboBrain 2.0. The blue point represents the model's spatial referring prediction.



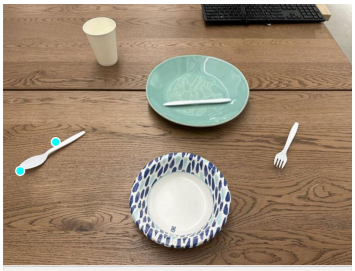
Reasoning Step = 1

What part of a *mug* *holds the liquid* inside for drinking?



Reasoning Step = 1

Which part of the *yellow bottle* can be *removed to access its content*?



Reasoning Step = 1

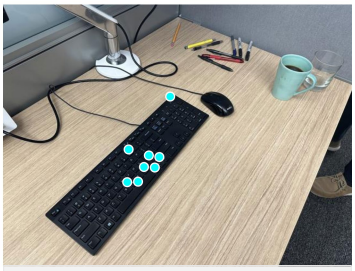
What *utensil* can be used to *scoop and transfer food to your mouth*?

Reasoning Step = 1

What part of a *mug* should be *gripped to lift it*?

Reasoning Step = 1

What *object* can be used to *hold and drink beverages*?



Reasoning Step = 1

What *object* can be used to *input text and commands into a computer by pressing its keys*?

Reasoning Step = 1

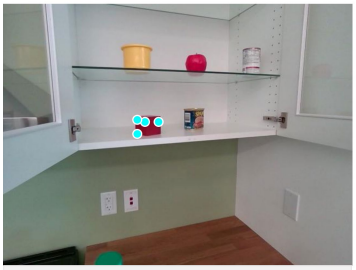
What part of the *cabinet* should be *pulled to open it*?

Reasoning Step = 1

Which part of the *bowl* can be *filled with soup or salad*?

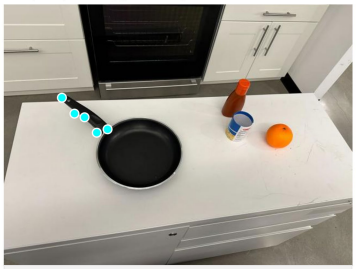
Reasoning Step = 1

What part of a *mug* should be *gripped to lift it*?



Reasoning Step = 1

What *object* can be stacked or used as a *building block in a structure*?



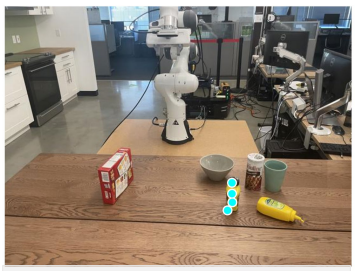
Reasoning Step = 1

What part of *pan* should be *gripped to lift it safely*?

Reasoning Step = 1

What part of a *faucet* should be *turned to control the water flow*?

Figure 16 Pointing Examples of RoboBrain 2.0. The objects or their parts are pointed according to their affordances queried in the instruction.



Reasoning Step = 2

Mark the *yellow bottle nearest to the bowl* in the image.

Reasoning Step = 2

Determine several points *on plate nearest to the cup* in the image.

Reasoning Step = 2

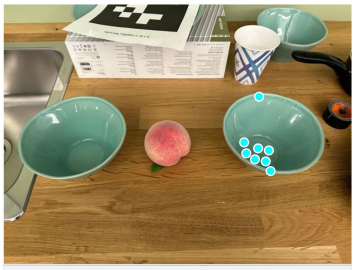
Indicate the *plate nearest to the cup* in the image.

Reasoning Step = 1

Highlight the *middle mug* in the image.

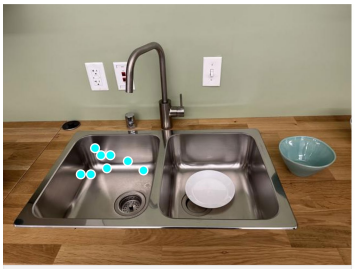
Reasoning Step = 1

Find several points on the *front can* in the image.



Reasoning Step = 2

Identify the *bowl right to the peach* in the image.

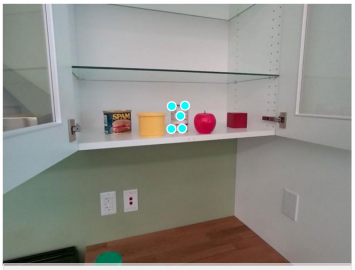


Reasoning Step = 1

Mark the *left sink* in the image.

Reasoning Step = 2

Highlight several points *on the notebook near to the plastic container* in the image.



Reasoning Step = 2

Spot several points *on the can left next to the apple* in the image.

Reasoning Step = 3

Identify the *cup right to the bowl below the cabinet* in the image.

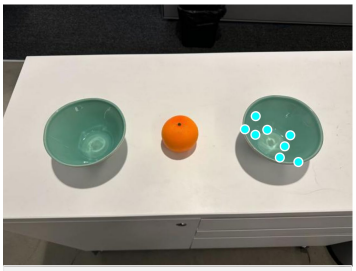
Reasoning Step = 1

Highlight the *rightmost fruit* in the image.

Reasoning Step = 2

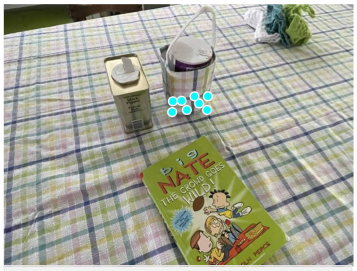
Determine several points *on the plate in the right sink* in the image.

Figure 17 Pointing Examples of RoboBrain 2.0. The objects referred by spatial relations or object attributes are pointed out.



Reasoning Step = 2

Highlight several points ***on the bowl right to the orange*** in the image.



Reasoning Step = 2

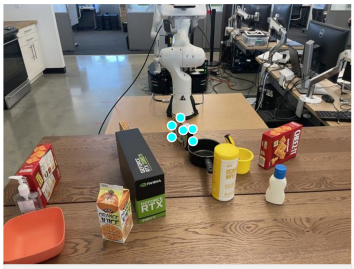
Highlight the ***object right to can*** in the image.

Reasoning Step = 1

Mark the ***rightmost object*** in the image.

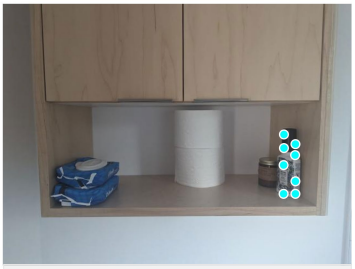
Reasoning Step = 2

Pinpoint the ***chair right to the table*** in the image.



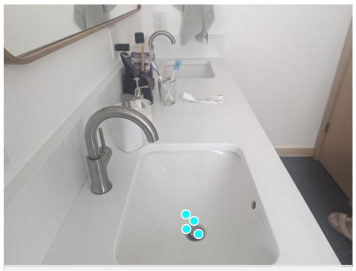
Reasoning Step = 2

Recognize several points ***on the box nearest to the pan*** in the image.



Reasoning Step = 2

Pinpoint the ***bottle in front of the can*** in the image.

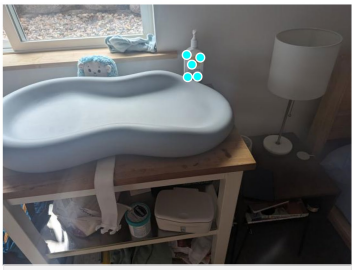


Reasoning Step = 1

Identify several points in the ***front sink*** in the image.

Reasoning Step = 2

Pinpoint the ***box left to the yellow can*** in the image.



Reasoning Step = 2

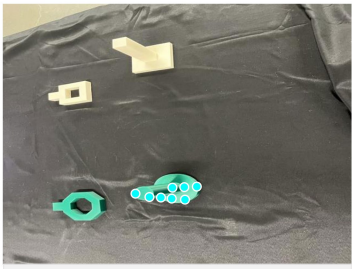
Highlight the ***bottle left to the lamp*** in the image.

Reasoning Step = 1

Indicate several points on the ***top drawer*** in the image.

Reasoning Step = 2

Highlight the ***box behind the pan*** in the image.



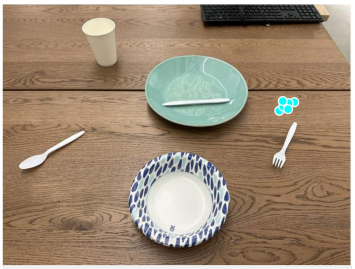
Reasoning Step = 2

Locate several points ***on the building block in the bottom-right corner*** in the image.

Figure 18 Pointing Examples of RoboBrain 2.0. The objects referred by spatial relations or object attributes are pointed out.

Reasoning Step = 3
Identify several spots *within the vacant space* that's *between the two mugs*.

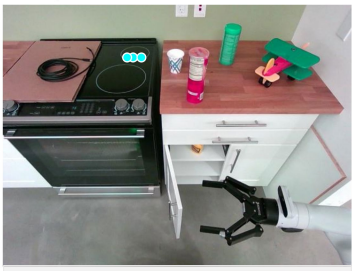
Reasoning Step = 2
Locate several points *within the vacant space* positioned to *the left of the yellow mustard bottle*.



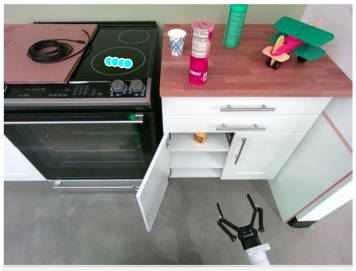
Reasoning Step = 3
Locate several points *within the vacant area* that is situated to *the right of the teal plate*.

Reasoning Step = 3
Select one or more locations *within the vacant area* that is *in front of the mug in the middle*.

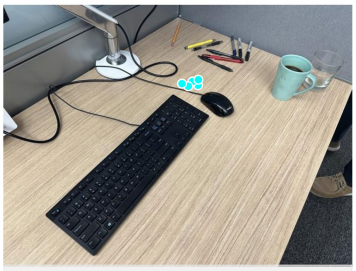
Reasoning Step = 3
Locate a few places *in the free space between the orange* and *the plastic cup*.



Reasoning Step = 2
Locate several points *within a vacant area on the back side of the stove*.

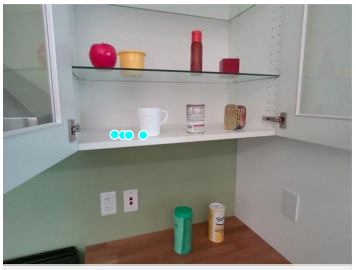


Reasoning Step = 2
Locate several points *within a vacant area on the front portion of the stove*.



Reasoning Step = 2
Locate a few spots *within the unoccupied space behind the mouse*.

Reasoning Step = 2
Locate a few spots *within the unoccupied area inside the cabinet*.



Reasoning Step = 2
Locate several spots *within the unoccupied area beneath the apple*.

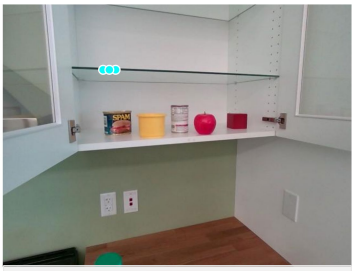
Reasoning Step = 3
Locate several spots *within the vacant area* that is *in front of the bowl on the left*.

Reasoning Step = 2
Locate several spots *within the vacant area* that is *in front of the teal bowl*.

Figure 19 Pointing Examples of RoboBrain 2.0. The free space indicated by spatial relations and the referenced objects are pointed out.

Reasoning Step = 2
Locate several points *within the vacant area* that lies *before the plastic container*.

Reasoning Step = 2
Locate several points *within the vacant area* that is *in front of the blue cup*.



Reasoning Step = 3
Locate several spots *within the vacant space* situated *above the leftmost item*.

Reasoning Step = 2
Pinpoint several spots *within the vacant area* located to *the right-hand of the green container*.

Reasoning Step = 4
Identify several points *within the vacant area* that lies *between the blue cup* and *the teal bowl on the table*.

Reasoning Step = 4
Locate a few points *within the unoccupied space* that lies *before the leftmost fruit on the table*.

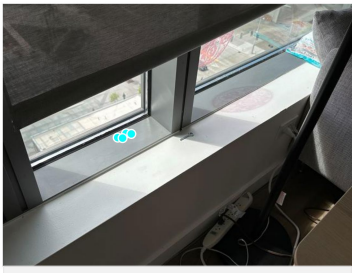
Reasoning Step = 2
Locate a few points *within the vacant space to the left of the frying pan*.

Reasoning Step = 2
Locate several spots *within the vacant area* situated to the *left side of the orange*.

Reasoning Step = 3
Locate several spots *in a vacant area next to the white mug*.

Reasoning Step = 2
Locate several points *within the vacant space* situated *on the left part of the cabinet shelf*.

Reasoning Step = 2
Locate several points *within a vacant area on the front side of the table*.



Reasoning Step = 3
Find a few points *in the free space in front of the window on the left*.

Figure 20 Pointing Examples of RoboBrain 2.0. The free space indicated by spatial relations and the referenced objects are pointed out.

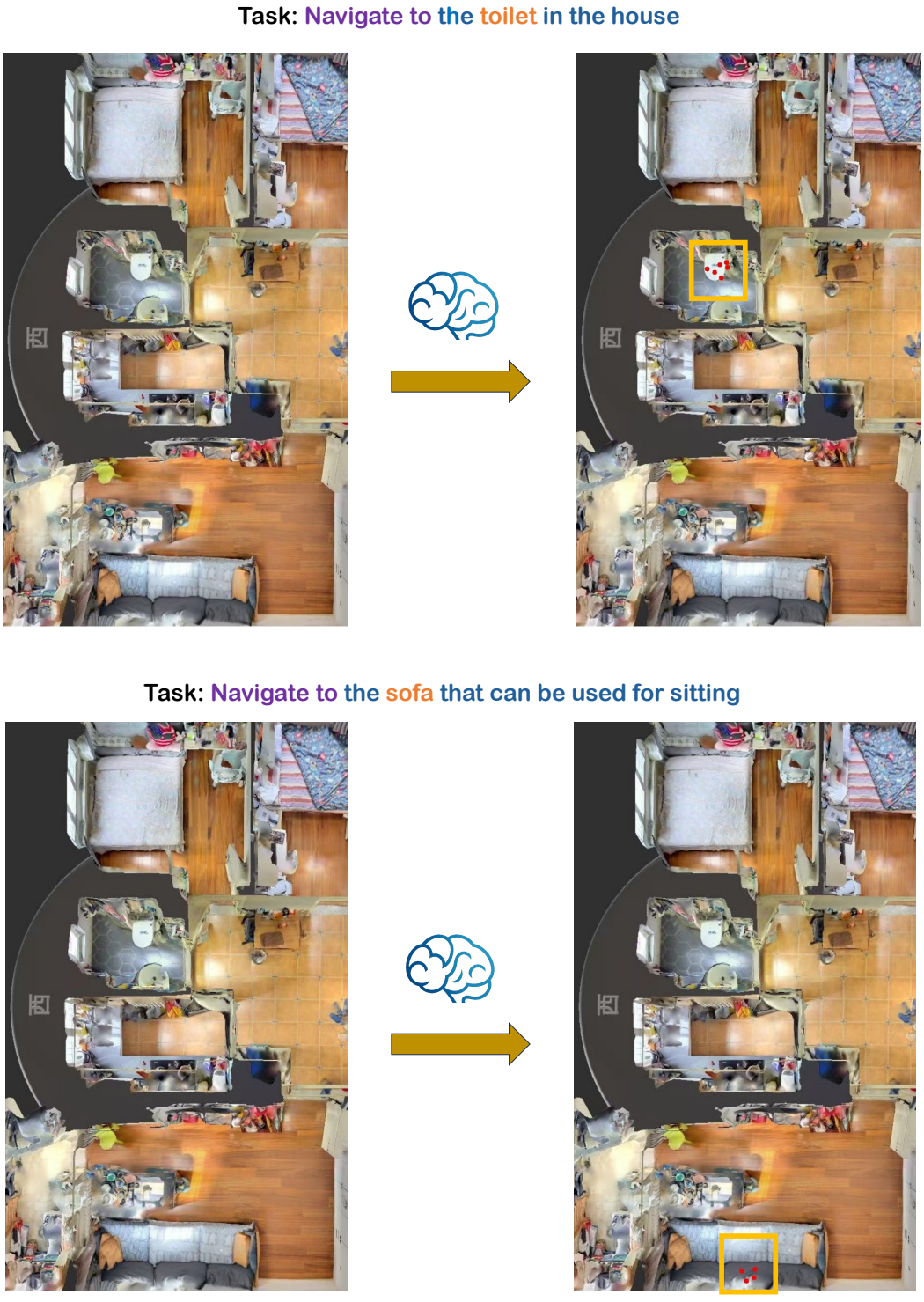


Figure 21 Pointing Examples of RoboBrain 2.0. The free space indicated by spatial relations and the referenced objects are pointed out.

A.2 Examples for Affordance

The affordance task assesses RoboBrain 2.0's understanding of object functionalities and interaction possibilities. For example, when asked "What part of a mug holds the liquid for drinking?" the model correctly identifies the interior of the mug as the part that holds the liquid. In another example, the instruction "Which part of a handbag can be grasped to carry it?" is accurately answered by identifying the handle of the handbag. These examples showcase the model's ability to reason about object affordances, making it capable of understanding how objects can be interacted with in the real world. As shown in Figure 22-Figure 23, the model demonstrates its proficiency in identifying functional parts of objects and their potential uses.

Reasoning Step = 2

Please identify the *affordance area* for *sitting on the bench*

Reasoning Step = 2

Please identify the *affordance area* for *ridding the motorcycle.*

Reasoning Step = 2

Please identify the *affordance area* for *opening the bottle.*

Reasoning Step = 3

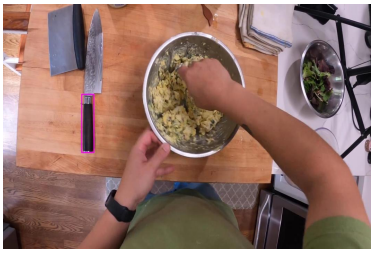
Please identify the *affordance area* for *washing the toothbrush.*

Reasoning Step = 2

Please identify the *affordance area* for *holding the cup.*

Reasoning Step = 2

Please identify the *affordance area* for *holding the wine glass.*



Reasoning Step = 2

Please identify the *affordance area* for *holding the knife.*

Reasoning Step = 1

Please identify the *affordance area* for *typing on the laptop.*

Reasoning Step = 2

Please identify the *affordance area* for *pushing the bicycle.*

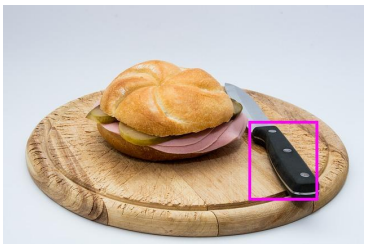
Figure 22 Affordance Examples of RoboBrain 2.0. The purple bounding boxes denote the actionable affordance areas for specific tasks.

Reasoning Step = 2

Please identify the *affordance area* for *pouring the wine glass.*

Reasoning Step = 3

Please identify the *affordance area* for *lying on the bench.*

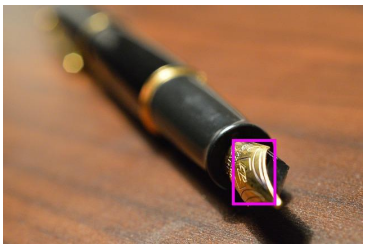


Reasoning Step = 2

Please identify the *affordance area* for *holding the knife.*

Reasoning Step = 2

Please identify the *affordance area* for *opening the bottle.*



Reasoning Step = 2

Please identify the *affordance area* for *the pen to write.*

Reasoning Step = 2

Please identify the *affordance area* for *picking up the suitcase.*

Reasoning Step = 2

Please identify the *affordance area* for *sipping the cup.*

Reasoning Step = 2

Please identify the *affordance area* for *holding the cup.*

Reasoning Step = 2

Please identify the *affordance area* for *opening the refrigerator.*

Reasoning Step = 2

Please identify the *affordance area* for *holding the cup.*

Reasoning Step = 2

Please identify the *affordance area* for *holding the cup.*

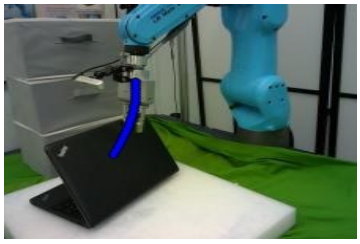
Reasoning Step = 2

Please identify the *affordance area* for *sitting on the bicycle.*

Figure 23 Affordance Examples of RoboBrain 2.0. The purple bounding boxes denote the actionable affordance areas for specific tasks.

A.3 Examples for Trajectory

The trajectory task evaluates the model's ability to predict and navigate paths based on given instructions. For instance, given the instruction "Please provide the trajectory to move the robot arm to grasp the apple," RoboBrain 2.0 generates a smooth and efficient path for the robot arm to follow. The model's trajectory predictions are accurate and take into account the spatial constraints and obstacles in the environment, demonstrating its proficiency in spatial and temporal reasoning for navigation tasks. As shown in Figure 24-Figure 25, the model effectively plans and executes trajectories that are both optimal and collision-free.

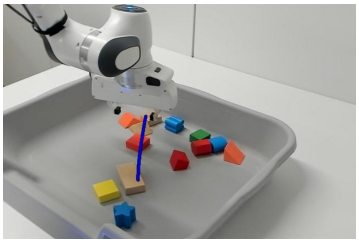


Reasoning Step = 1
Please predict the *key trajectory points* for *closing the laptop.*

Reasoning Step = 3
Please predict the *key trajectory points* for *making a piece of toast with the oven.*

Reasoning Step = 2
Please predict the *key trajectory points* for *moving apple near green can.*

Reasoning Step = 2
Please predict the *key trajectory points* for *picking water bottle from bottom shelf of fridge.*



Reasoning Step = 1
Please predict the *key trajectory points* for *picking bin.*

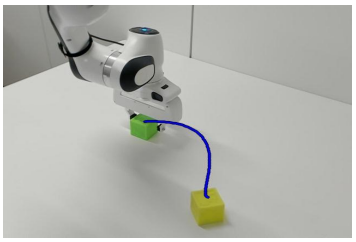
Reasoning Step = 2
Please predict the *key trajectory points* for *placing the mushroom into the metal pot.*

Reasoning Step = 2
Please predict the *key trajectory points* for *placing the mushroom on the blue towel.*

Reasoning Step = 2
Please predict the *key trajectory points* for *placing the orange pot to the left of the potato.*

Reasoning Step = 2
Please predict the *key trajectory points* for *placing the pan on the far right edge of the table.*

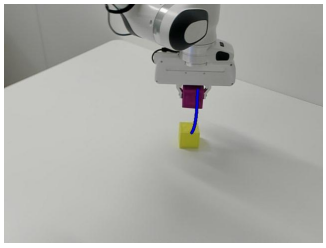
Figure 24 Trajectory Examples of RoboBrain 2.0. The blue trajectories, composed of key trajectory points, represent the model-predicted paths for task completion.



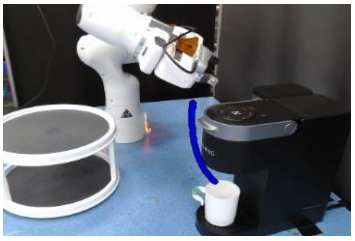
Reasoning Step = 2
Please predict the *key trajectory points* for *moving green cube to the top of yellow cube.*

Reasoning Step = 1
Please predict the *key trajectory points* for *closing middle drawer.*

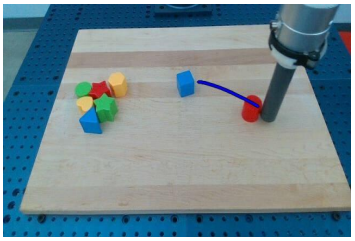
Reasoning Step = 1
Please predict the *key trajectory points* for *closing top drawer.*



Reasoning Step = 1
Please predict the *key trajectory points* for *destacking purple yellow cube.*



Reasoning Step = 2
Please predict the *key trajectory points* for *making a cup of coffee with the Keurig machine.*



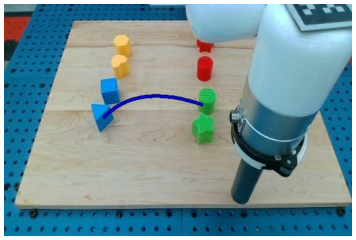
Reasoning Step = 2
Please predict the *key trajectory points* for *moving red circle closer towards blue cube.*

Reasoning Step = 2
Please predict the *key trajectory points* for *moving red spoon to just below green towel.*

Reasoning Step = 2
Please predict the *key trajectory points* for *moving red spoon to upper right corner of table.*

Reasoning Step = 2
Please predict the *key trajectory points* for *moving rxbar chocolate near coke can.*

Reasoning Step = 2
Please predict the *key trajectory points* for *moving spatula onto the towel.*



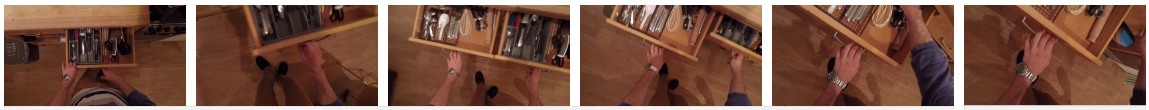
Reasoning Step = 2
Please predict the *key trajectory points* for *moving blue triangle below the green star*

Reasoning Step = 2
Please predict the *key trajectory points* for *picking green can from middle shelf of fridge.*

Figure 25 Trajectory Examples of RoboBrain 2.0. The blue trajectories, composed of key trajectory points, represent the model-predicted paths for task completion.

A.4 Examples for EgoPlan2

The EgoPlan2 task focuses on planning daily activities from an egocentric perspective. For instance, given the instruction "Plan the steps to prepare a cup of coffee," RoboBrain 2.0 outlines a detailed sequence of actions, including locating the coffee machine, fetching the coffee beans, and following the steps to brew the coffee. The model's ability to break down complex tasks into actionable steps demonstrates its proficiency in task decomposition and sequential planning. As shown in Figure 26-Figure 28, the model effectively plans and executes multi-step tasks, showcasing its capabilities in long-horizon planning and task execution.



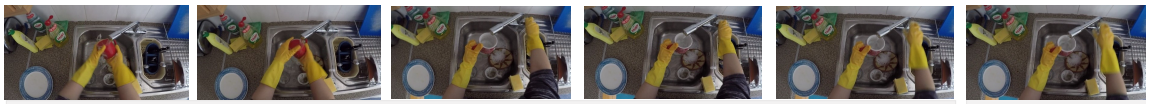
Question: I am tasked with ***Move the saucepan and light up the candle***. The task's progress is demonstrated in the provided video. What should be my next action?
(A). move saucepan ***(B). close drawer*** (C). adjust hob (D). open drawer

Thinking: ***From the visual input, I observe an open drawer containing various utensils such as spoons and a whisk. My focus remains on completing the task of moving the saucepan and lighting a candle. The current step involves handling objects from this drawer. Based on this, the logical next action is to close the drawer after retrieving the necessary item (likely utensils) for cooking or handling the saucepan, ensuring workspace orderliness and safety.***

The available options include: (A) Moving the saucepan, (B) Closing the drawer, (C) Adjusting the hob, and (D) Opening another drawer. Given that the drawer is already open and no further items appear necessary at the moment, closing it aligns with maintaining organization and preventing obstruction during subsequent steps.

Therefore, the most reasonable action is option (B), closing the drawer.

Answer: (B). close drawer



Question: My current task is to ***Wash and rinse two mugs***. The task's progress is demonstrated in the provided video. What should be my next action?
(A). put mug ***(B). rinse mug*** (C). pick up mug (D). close tap

Thinking: ***From the visual input, I observe that I am positioned near a sink within what appears to be a kitchen environment. My task involves washing and rinsing two mugs. Currently, I am holding one red mug and a sponge, actively cleaning it. Other items like a blue plate are visible on the counter nearby, suggesting this is part of a larger cleaning task.***

The sink contains various other dishware, including another mug and additional kitchen utensils, indicating my current focus on washing multiple mugs. Given the visual context, the next logical step after cleaning the mug would be to rinse it thoroughly before placing it aside or moving on to wash the second mug.

Considering the options provided—(A) put mug, (B) rinse mug, (C) pick up mug, (D) close tap—the most immediate and sensible action aligns with rinsing the mug to ensure it is properly cleaned and ready for further steps, such as drying or placing it away.

Therefore, based on my visual analysis of the scene and the task at hand, the appropriate next action is to rinse the mug, which corresponds to option (B).

Answer: (B). rinse mug

Figure 26 EgoPlan Examples of RoboBrain 2.0. The key reasoning process and final answer related to specific tasks are depicted.

Question: I am in charge of ***Take mozzarella, open the bag, spread it onto the dough and add salt***, and the video reveals the task's advancement, What is the most rational next move?
(A). spread mozzarella onto dough (B). put down knife (C). take mozzarella (D). open mozzarella bag

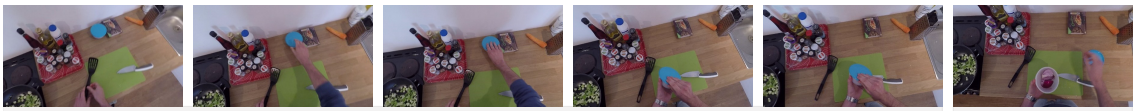
Thinking: ***From the visual input, I recognize that my task involves preparing ingredients for a dish on a wooden countertop. I observe a loaf of mozzarella, which is currently being held by me, and an open packet containing additional mozzarella slices placed near the edge of the countertop. A flat, baked surface with sliced mozzarella arranged neatly appears to be the target area for spreading the cheese.***

Given this setup, my next logical step is to apply the cheese directly onto the prepared flat surface with the slices of mozzarella already laid out, ensuring even distribution over the crust. This aligns with the objective of adding cheese as part of the preparation process.

Comparing my options:
- Option (A) "spread mozzarella onto dough" directly addresses the current task at hand, as it involves positioning the cheese appropriately on the target area.
- Option (B) "put down knife" doesn\'t fit the immediate need since the task focuses on spreading rather than cutting.
- Option (C) "take mozzarella" is redundant, as I already have the mozzarella in hand.
- Option (D) "open mozzarella bag" is unnecessary since the bag is already open, and the focus is on applying the cheese.

Thus, based on the visual scene analysis and task requirements, the most rational next move is to spread the mozzarella onto the dough. Therefore, the selected action is (A) "spread mozzarella onto dough.

Answer: (A). spread mozzarella onto dough



Question: My assignment is to ***Preparing onion and boiling water, with the task's progress evident in the video***. What should I do next? Please provide a reasonable and succinct recommendation.
(A). take onion (B). open container (C). take knife (D). open tap

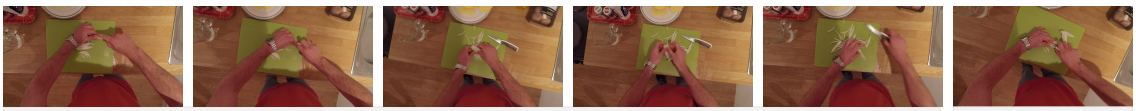
Thinking: ***From the visual input, the bowl containing onions is held and is the primary focus of the scene. The onion pieces are visible inside the bowl, indicating that it is already opened and ready for use. The surrounding context includes kitchen items such as a spatula, a knife on the counter, and a grater nearby, but these are not directly related to the task here. My task involves preparing an onion, which requires handling or slicing the onion contents. Given this, the logical next step would be to focus on utilizing the already accessible onion by taking it out of the bowl or using it for cooking.***

The options provided include (A) taking the onion itself, (B) opening a container (which has been done), (C) taking a knife (not relevant at this stage since the onion is already handled), and (D) opening the tap (irrelevant as no water preparation is observed). Among these, option (A) aligns most directly with the immediate need to prepare the onion.

Therefore, based on the observed situation, the most reasonable next action is to take the onion from the bowl, ensuring readiness for further steps in the cooking process.

Answer: (A). take onion

Figure 27 EgoPlan Examples of RoboBrain 2.0. The key reasoning process and final answer related to specific tasks are depicted.



Question: As I am in the process of *Prepare and slice more onions*, with my progress visible in the video. what do you suggest as my next move? (A). take knife (B). turn knife (C). throw onion skin into bin *(D). continue cutting onion*

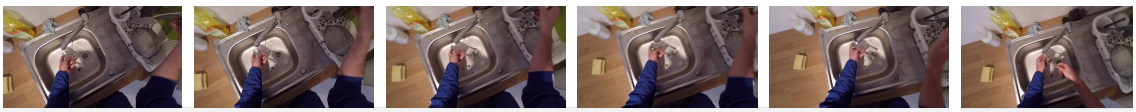
Thinking: ***From the visual input, the person is focused on slicing an onion on a green cutting board. The onion's white layers have been separated, and some skin or layers are visible on the cutting surface. To maintain safety and efficiency, continuing to slice the onion is logical. The person holds the onion steady, and their fingers are positioned to maintain control during further cuts.***

Analyzing the current task, the objective remains to chop the onion neatly without causing any unnecessary mess. The presence of a knife suggests readiness for this action. With the onion already sliced partially, the next logical step is to complete the task by continuing to slice.

Considering the options provided:
(A) indicates taking the knife again, which might be redundant given its current proximity.
(B) suggests turning the knife, which may not be necessary unless there's an issue with the orientation.
(C) proposes throwing away the onion skin, which could happen after cutting but not as part of the ongoing slicing process.
(D) confirms the need to continue cutting the onion, aligning well with the observed task progression.

Based on the visual scene and task context, option (D) to continue cutting the onion is the appropriate choice, as it supports the process of preparing and slicing the onion efficiently.

Answer: (D). continue cutting onion



Question: Given my responsibility to *Clean the sink and counter*, the video shows the progress What is the most logical next step?
(A). take spoon *(B). rinse spoon* (C). take cloth (D). rinse cloth

Thinking: ***From the visual input, I am observing a kitchen environment where cleaning activities are taking place. The focus is on a sink area that contains various cleaning tools, including utensils and dishes. A spoon is being held over the sink, which is filled with water. The faucet is nearby, suggesting its use for rinsing or further cleaning actions.***

Given the current task of cleaning, the spoon appears to be in the process of being washed, as it is already oriented towards the water source. The fact that this spoon is already being handled indicates readiness for further actions, such as rinsing or placing it elsewhere after cleaning. The surrounding environment includes a drying rack holding other utensils, reinforcing the notion of completing cleaning tasks sequentially.

Analyzing the sequence and progression of cleaning actions, the spoon is currently within a position conducive to rinsing under running water. This action is the next logical step before moving on to other items or utensils, to ensure all surfaces are appropriately cleaned for safe handling or placement. The other options—such as taking a cloth or additional utensils—do not align with the immediate visual context, as the spoon is already in an appropriate state for this activity.

Therefore, based on the visual analysis, the most logical next step is to rinse the spoon (Option B). This action ensures proper cleaning and aligns with the observed process within the image.

Answer: (B). rinse spoon

Figure 28 EgoPlan Examples of RoboBrain 2.0. The key reasoning process and final answer related to specific tasks are depicted.

A.5 Examples for Close-Loop Interaction

Close-loop interaction examples showcase RoboBrain 2.0's ability to engage in interactive reasoning with feedback. For example, in a scenario where the model is asked to "Find a muff cup and pour coffee into it," it not only needs to navigate and search for the mug multiple times within the task environment but also must operate the coffee machine based on feedback to complete the pouring process. This iterative process highlights the model's capability to refine its actions based on real-time feedback, ensuring more accurate and reliable performance in interactive tasks. As shown in Figure 29-Figure 32, the model demonstrates its ability to adapt and improve its responses through iterative feedback loops.

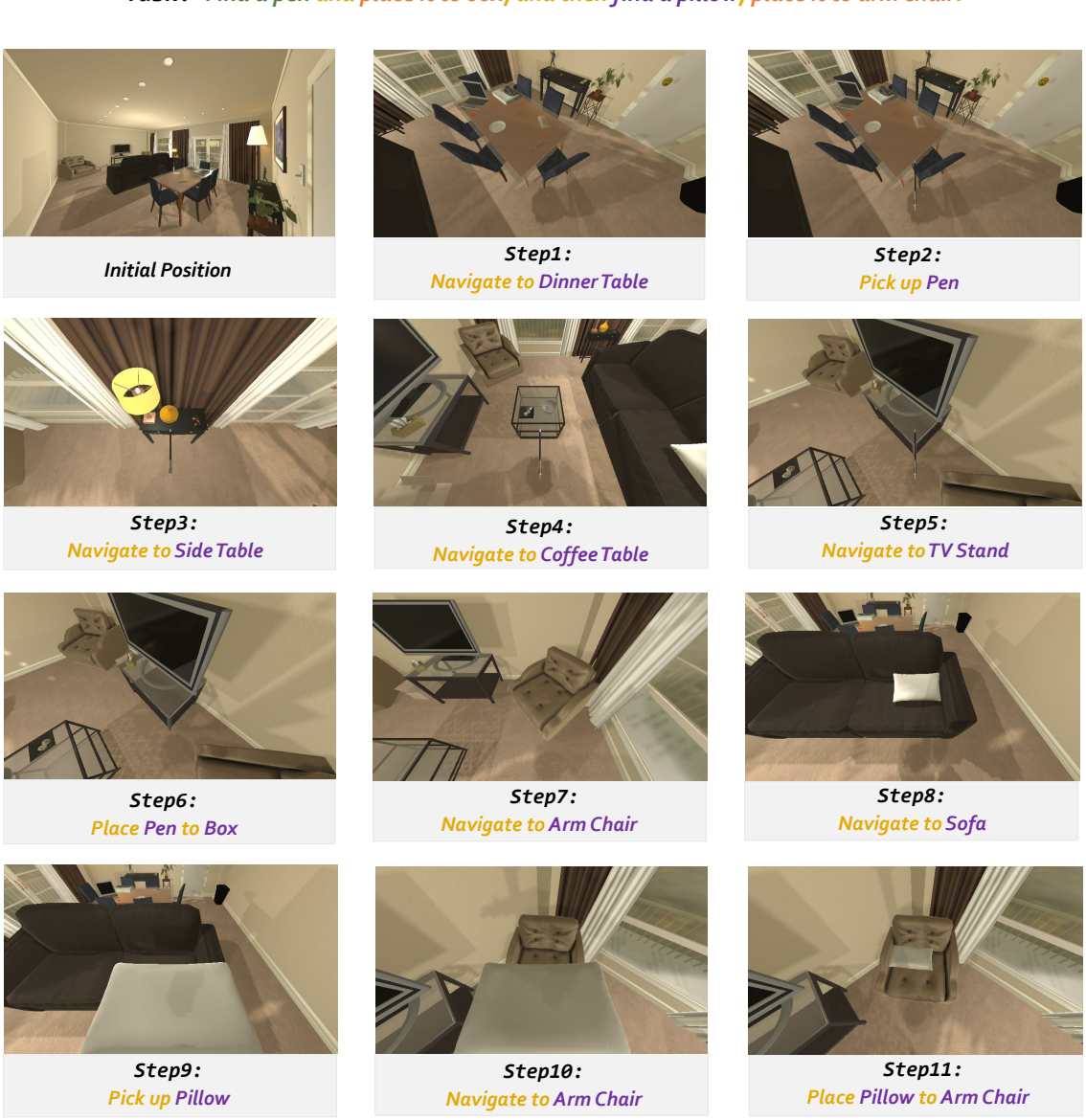


Figure 29 Close-loop planning Examples of RoboBrain 2.0. The key planning steps related to specific tasks are depicted.

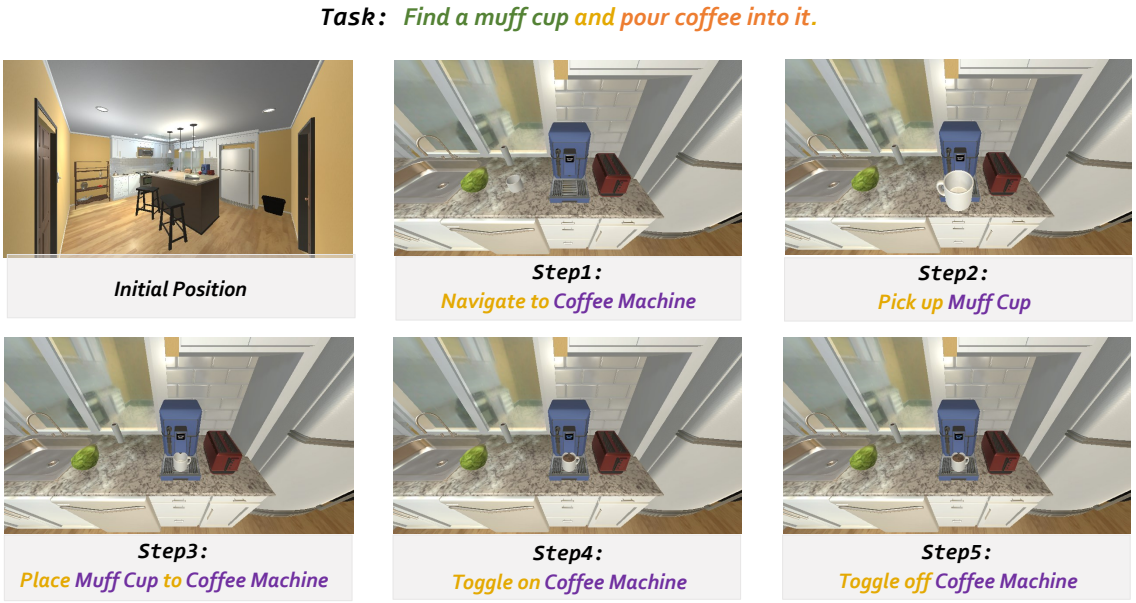


Figure 30 Close-loop planning Examples of RoboBrain 2.0. The key planning steps related to specific tasks are depicted.

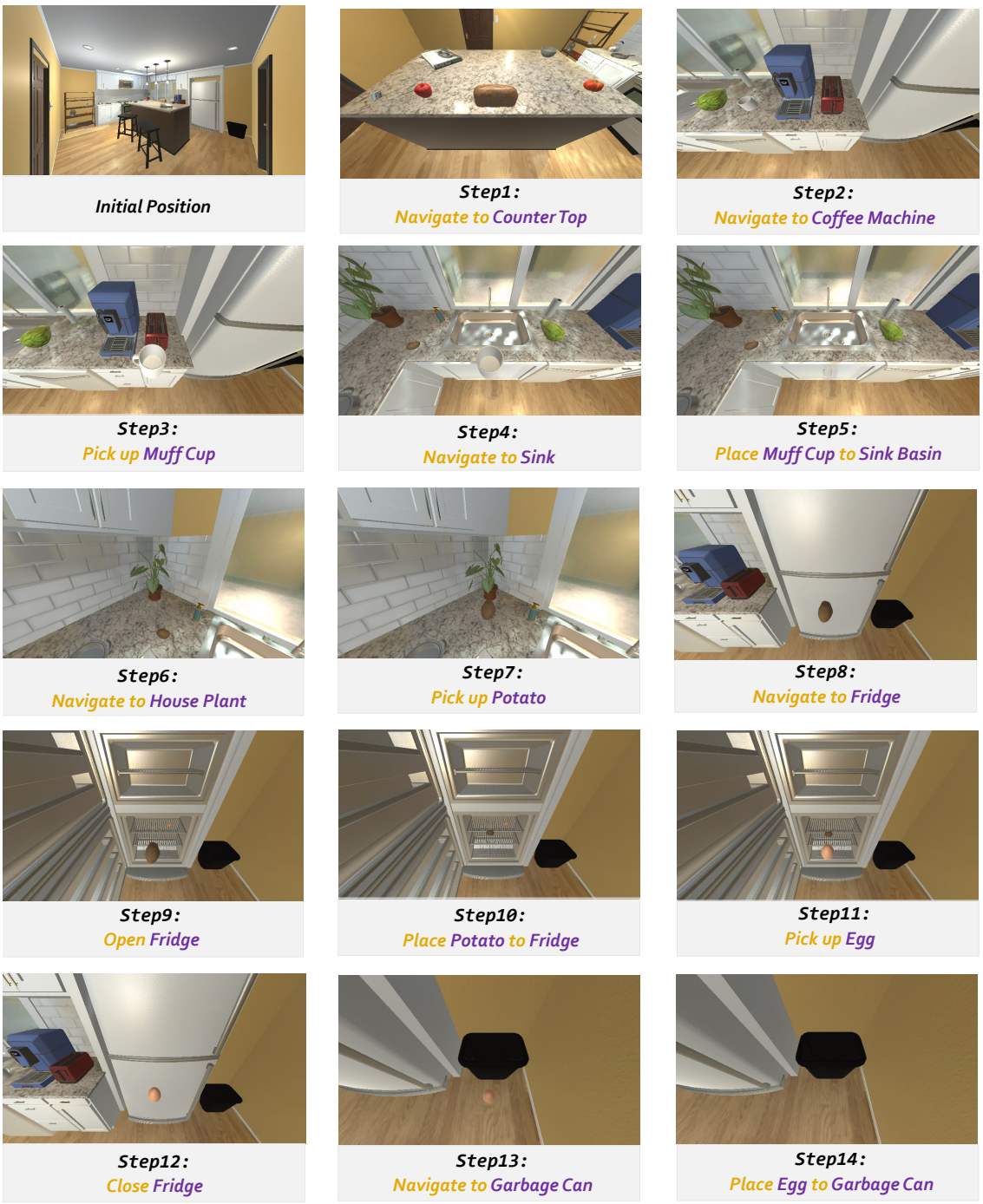


Figure 31 Close-loop planning Examples of RoboBrain 2.0. The key planning steps related to specific tasks are depicted.

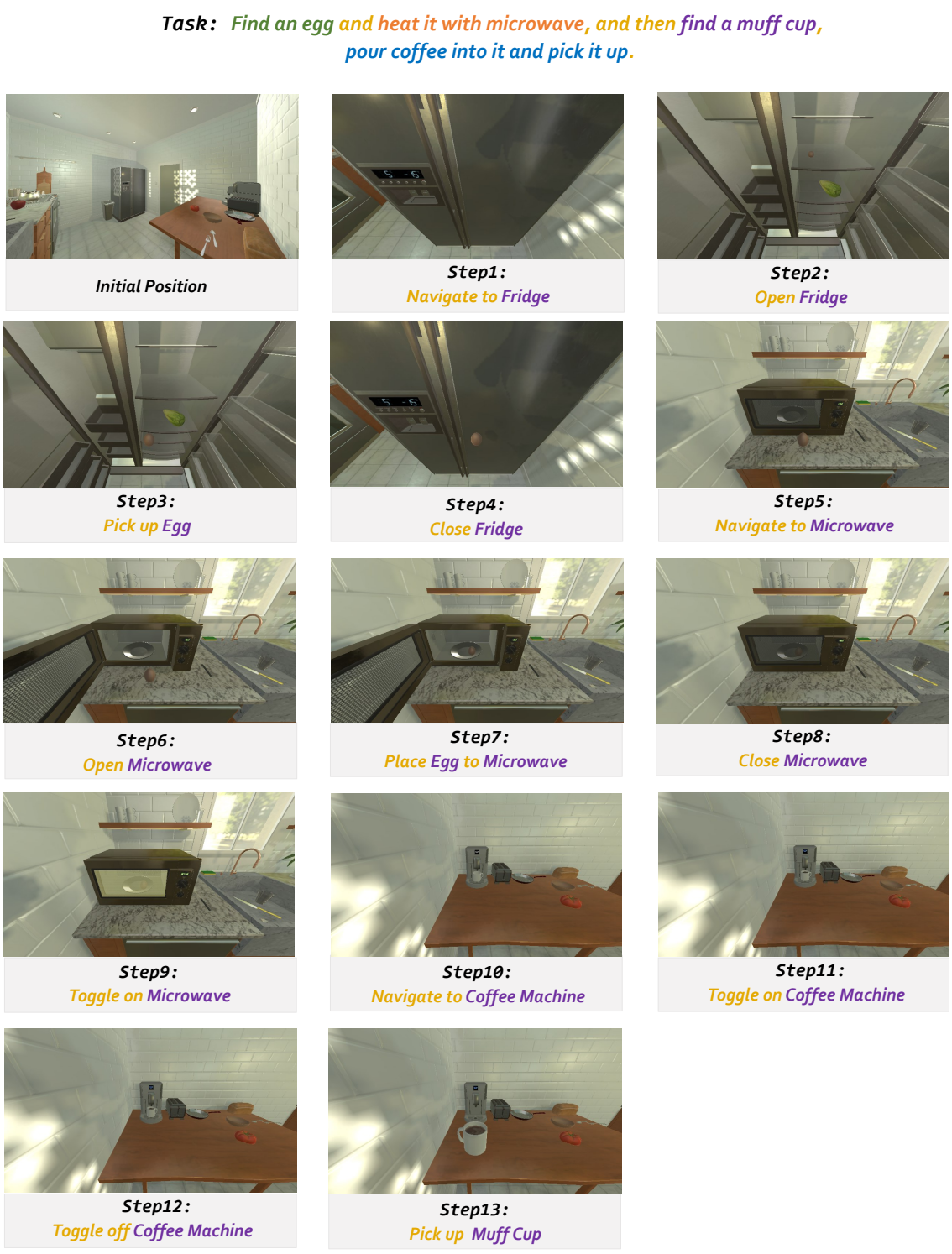


Figure 32 Close-loop planning Examples of RoboBrain 2.0. The key planning steps related to specific tasks are depicted.

A.6 Examples for Multi-Robot Planning

In multi-robot planning scenarios, RoboBrain 2.0 coordinates the actions of multiple robots to achieve a common goal. For example, in a supermarket scenario, the model plans the movements of multiple robots to efficiently restock shelves. The planning involves assigning specific tasks to each robot, coordinating their movements to avoid collisions, and ensuring that the overall goal is achieved in a timely manner. These examples highlight the model's advanced capabilities in multi-agent coordination and long-horizon planning.

As shown in Figure 33, the model demonstrates its ability to orchestrate complex multi-robot activities with high precision and efficiency. In the restaurant setting (Figure 33(a)), a Unitree G1 humanoid and Agilex dual-arm robot collaborate on burger preparation and delivery for the command "I'm hungry and order a normal burger," with RoboBrain 2.0 performing scene-aware task decomposition. The household scenario (Figure 33(b)) features a Realman single-arm and Agilex dual-arm robot executing commands like "Give me an orange and a knife." In the supermarket (Figure 33(c)), RoboBrain 2.0 assists customers with gift selection by analyzing dimensions and bag compatibility, coordinating the Realman robot for gift placement and the Agilex executing VLA-cerebellum skills like "open the gift bag." Please refer to RoboOS [61] for more details.

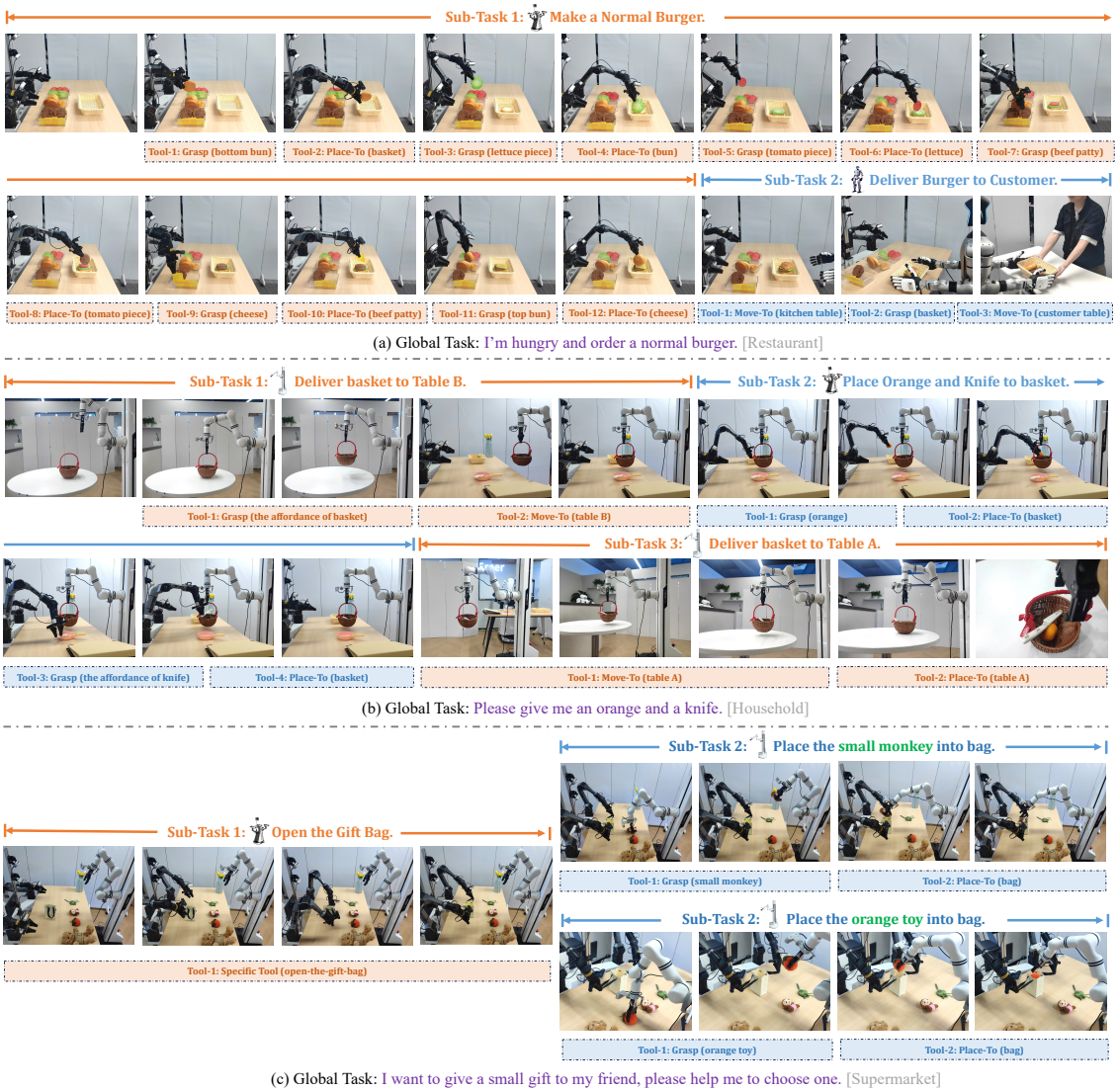


Figure 33 We showcase multi-robot collaboration in three scenarios: (a) Restaurant: Unitree G1 and Agilex robots prepare burgers. (b) Household: Realman and Agilex robots fetch items. (c) Supermarket: Robots coordinate gift selection and packaging.

A.7 Examples for Synthetic Benchmarks

Synthetic benchmarks are used to evaluate RoboBrain 2.0's performance on a variety of spatial and temporal reasoning tasks. For instance, in the BLINK benchmark, which assesses depth perception and spatial relation understanding, the model achieves high accuracy in identifying the relative positions and distances of objects. In the CV-Bench benchmark, which evaluates 3D spatial understanding, RoboBrain 2.0 demonstrates its ability to accurately process and reason about 3D scenes. These synthetic benchmarks provide a comprehensive evaluation of the model's capabilities across different reasoning dimensions. As shown in Figure 34-Figure 35, the model consistently performs well across various synthetic benchmarks, showcasing its robust abilities.

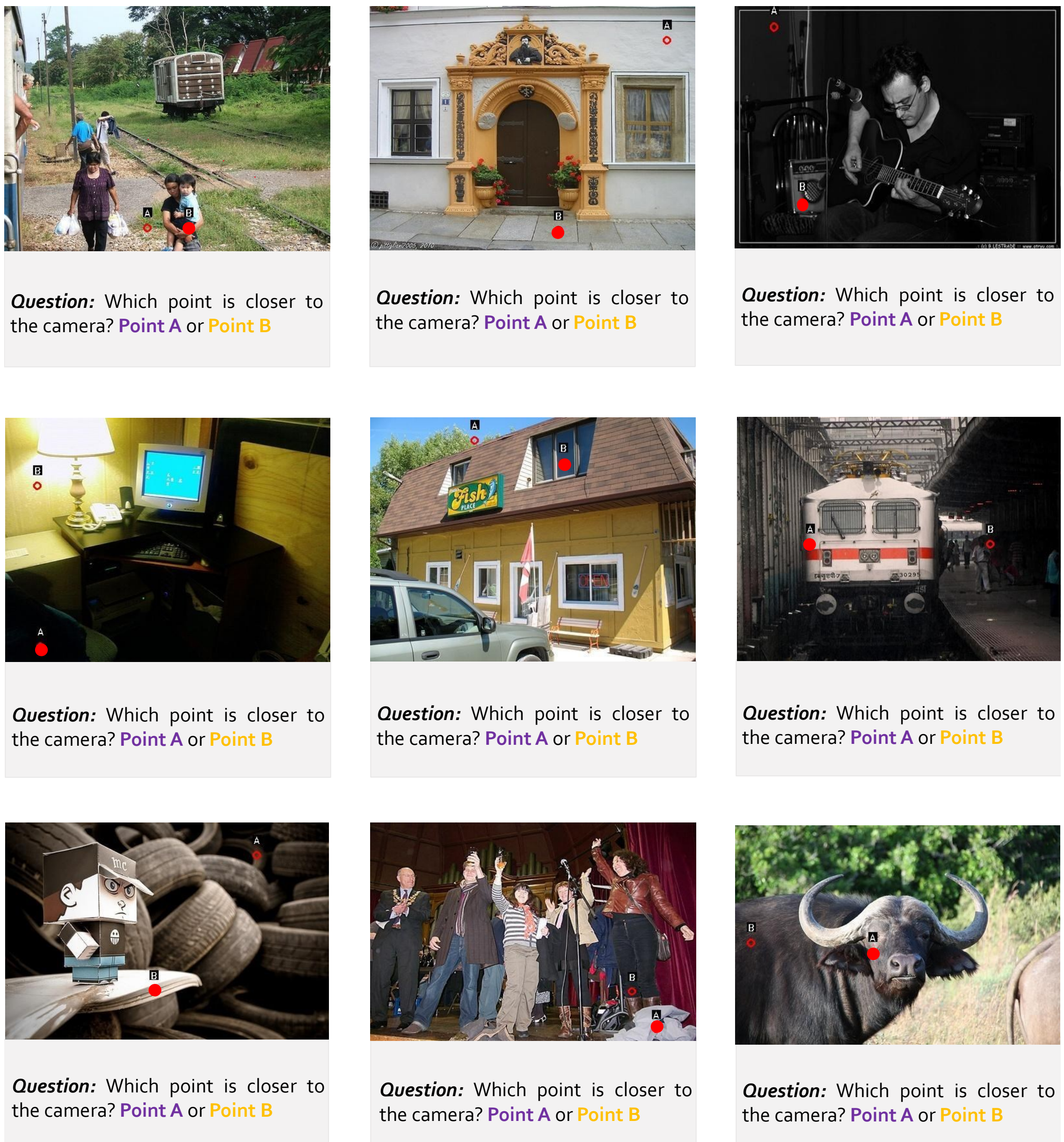


Figure 34 CVbench Benchmark Examples of RoboBrain 2.0. The solid circle in the diagram represents the selected point.

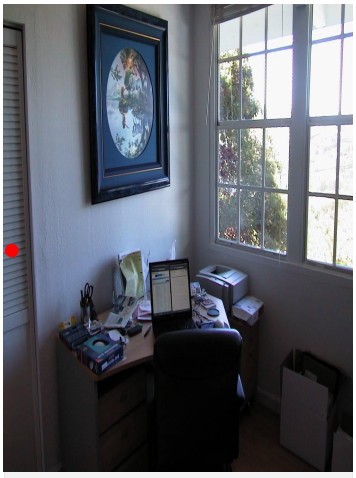
Question: How many televisions are in the image? Point out and answer.
Answer: *1*

Question: How many table lamps are in the image? Point out and answer.
Answer: *1*

Question: How many pictures are in the image? Point out and answer.
Answer: *1*

Question: How many walls are in the image? Point out and answer.
Answer: *1*

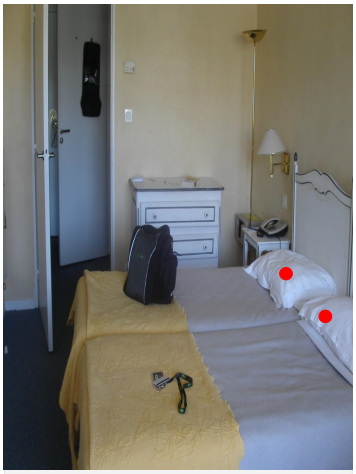
Question: How many curtains are in the image? Point out and answer.
Answer: *2*



Question: How many blinds are in the image? Point out and answer.
Answer: *1*

Question: How many curtains are in the image? Point out and answer.
Answer: *2*

Question: How many walls are in the image? Point out and answer.
Answer: *2*



Question: How many pillows are in the image? Point out and answer.
Answer: *2*

Figure 35 BLINK Benchmark Examples of RoboBrain 2.0. The solid circle in the diagram represents the selected object.

B Prompts Details

This section outlines the system prompts for various spatial understanding and planning tasks assigned to a robot with advanced visual and analytical capabilities. Each task requires simulating visual-spatial reasoning, leveraging visual inputs as if directly perceiving the scene, and generating step-by-step reasoning processes within `<think></think>` tags, with answers in `<answer></answer>` tags. Reasoning is kept concise (200–500 words) and follows a five-stage process tailored to each task. Instructions emphasize direct visual-spatial language, avoiding abstract references to input data (e.g., "based on the description") and maintaining the robot's role.

B.1 Spatial Understanding: Coordinates -- Pointing

The robot is tasked with identifying specific points within an image based on visual criteria, such as locating points in a vacant area on a delineated plane. The reasoning process includes:
Object Analysis Observe the object's shape, size, and spatial relationships (e.g., a red rectangular border delineating a plane with vacant patches).
Capability Assessment Relate visual processing capabilities to identifying vacant regions and pinpointing coordinates.
Contextual Relevance Focus on the task requirement to select points within the vacant area.
Verification Ensure selected coordinates lie within boundaries and are distinct.
Point Conclusion Output coordinates as a list of tuples, justified by visual analysis.
Example Prompt: Locate points within a vacant area on a plane outlined by a red border. Your answer should be formatted as a list of tuples, i.e. $[(x_1, y_1), (x_2, y_2), \dots]$, where each tuple contains the x and y coordinates of a point satisfying the conditions above. The coordinates should indicate the normalized pixel locations of the points in the image.
Example Output w/ Thinking: `<think> ... </think><answer>`$[(296, 282), (321, 256), \dots]$`</answer>`.

B.2 Spatial Understanding: Coordinates -- Trajectory

The robot predicts a sequence of key trajectory points to achieve a goal, such as reaching an object. The reasoning process includes:
Object Analysis Identify the target object's properties and spatial relationships (e.g., a banana on a plate with potential obstacles nearby).
Capability Assessment Use joint control to plan smooth end-effector paths, avoiding obstacles.
Contextual Relevance Ensure the trajectory aligns with the goal (e.g., reaching the banana).
Verification Confirm the path avoids obstacles and reaches the target.
Trajectory Conclusion Output trajectory points as $[[x_1, y_1], [x_2, y_2], \dots]$, justified by visual and kinodynamic analysis.
Example Prompt: You are a robot using the joint control. The task is "Reach for a banana on a plate". Please predict up to 10 key trajectory points to complete the task. Your answer should be formatted as a list of tuples, i.e. $[[x_1, y_1], [x_2, y_2], \dots]$, where each tuple contains the x and y coordinates of a point."
Example Output w/ Thinking: `<think> ... </think><answer>`$[[116, 114], [153, 97], \dots]$`.</answer>`.

B.3 Spatial Understanding: Bounding Box -- Affordance

The robot identifies an affordance area for interaction with an object, such as grasping a handle. The reasoning process includes:
Object Analysis Describe the object's shape, size, and material properties (e.g., a blue coffee mug with a handle, inferred as ceramic from sheen).
Capability Assessment Assess gripper compatibility with the object's features (e.g., handle size vs. gripper opening).
Contextual Relevance Align with the task goal (e.g., preparing coffee by grasping the mug).
Verification Confirm the affordance area suits the interaction and is within reach.
Affordance Conclusion Output the bounding box as $[x_{min}, y_{min}, x_{max}, y_{max}]$, justified by visual compatibility.

Example Prompt: You are a robot using the joint control. The task is "hold a coffee mug". Please predict a possible affordance area of the end effector.
Example Output w/ Thinking: `<think> ... </think><answer>`$[915, 408, 1109, 533]$`.</answer>`.

B.4 Spatial Understanding: Freeform Q&A -- General Spatial Analysis

The robot answers questions about spatial relationships or action outcomes based on one or more images. The reasoning process includes:
Scene Perception Detail prominent features and their spatial arrangement (e.g., a metallic gripper above a green book on a shelf).
Task Objective Interpretation Clarify the question's focus (e.g., predicting the outcome of a gripper's trajectory).
Focused Visual Analysis Examine relevant scene elements or project actions (e.g., a yellow trajectory toward a lower shelf).
Relational Reasoning Synthesize observations to form a hypothesis, evaluating provided options.
Conclusion Derivation Output the answer, justified by visual evidence and logical reasoning.
Example Prompt: Predict the outcome of a gripper following a yellow trajectory. Options: (A) place book on lower shelf; (B) place book on upper shelf.
Example Output w/ Thinking: `<think> ... </think><answer>`(A)`</answer>`.

B.5 Temporal Understanding: Long-horizon Planning

The robot determines the next action in a task (e.g., cooking) based on a sequence of images and the current view. The reasoning process includes:
Task Progress Analysis Interpret completed actions from the sequence (e.g., onions peeled and sliced on a cutting board).
Current Scene Analysis Describe the current view's objects and state (e.g., frying pan on hob, oil container nearby).
Contextual Relevance Align with the task goal (e.g., cook onions by preparing the pan).
Action Option Evaluation Assess options for suitability (e.g., pour oil vs. peel onion, considering onions are already prepared).
Next Action Conclusion Output the next action, justified by visual evidence and task flow.
Example Prompt: Prepare and cook onions; choose the next action (options: pour oil, turn up hob, etc.).
Example Output w/ Thinking: `<think> ... </think><answer>`Pour oil.`</answer>`.

B.6 Temporal Understanding: Closed Loop Conversation

The robot answers a question within a conversation history, leveraging prior visual inputs and responses. The reasoning process includes:
Task Progress Recall Recap previous actions and their outcomes (e.g., opened the fridge to access ingredients).
Initial Analysis Focus on current visual input relevant to the question (e.g., a coffee machine on the countertop).
Contextual Relevance Align with the current task goal (e.g., flipping the coffee machine switch).
Action Option Evaluation Assess options for logical progression based on history and current state.
Next Action Conclusion Output the action, justified by visual evidence and conversation context.
Example Prompt: The task is "Flip the coffee machine switch after opening the fridge." After you have finished <action> , you can see <image>, and the feedback of final action is xxx. What is your next action?
Example Output w/ Thinking: `<think> ... </think><answer>`Toggle on Coffee Machine.`</answer>`.

B.7 Temporal Understanding: Multi-Robot Planning

The robot coordinates actions with other robots to achieve a common goal, devided into global task decomposition and agent-based tool-calling.
Example Prompt for Global Task Decomposition: Please Refer to Figure 36.
Example Prompt for Agent-based Tool-calling: Please Refer to Figure 37.
Example Output w/ Thinking: `<think> ... </think><answer>`Graph of TaskFlow`</answer>`.

System Prompt for Global Task Decomposition

 # You are a robotics expert specializing in task decomposition.

Your role is to decompose tasks into subtasks based on the task description and assign them to different robots for execution.

Example 1:

Current Robot: realman_1, singlearm_1, doublearm_1

Current Task: All the robots go to the table and bring an apple to the fridge respectively.

Your answer:

```json
[
  {{'robot_id': 'realman_1', 'subtask': 'go to the table and bring an apple to the fridge.', 'subtask_order': '0'}},
  {{'robot_id': 'singlearm_1', 'subtask': 'go to the table and bring an apple to the fridge.', 'subtask_order': '0'}},
  {{'robot_id': 'doublearm_1', 'subtask': 'go to the table and bring an apple to the fridge.', 'subtask_order': '0'}},
]
```

Example 2:

Current Robot: realman_1, doublearm_1

Current Task: realman take the basket from table_1 to table_2, then doublearm take the apple into basket in table_2, then realman take the basket back to table_1.

Your answer:

```json
[
  {{'robot_id': 'realman_1', 'task': 'bring the basket from table_1 to table_2.', 'task_order': '0'}},
  {{'robot_id': 'doublearm_1', 'task': 'pick an apple into the basket.', 'task_order': '1'}},
  {{'robot_id': 'realman_1', 'task': 'bring the basket from table_2 to table_1.', 'task_order': '2'}},
]
```

Note: 'subtask_order' means the order of the sub-task.

If the tasks are not sequential, please set the same 'task_order' for the same task. For example, if two robots are assigned to the two tasks, both of which are independance, they should share the same 'task_order'. If the tasks are sequential, the 'task_order' should be set in the order of execution. For example, if the task_2 should be started after task_1, they should have different 'task_order'.

 # Now it's your turn !!!

We will provide more scenario information and robot information. Based on the following robot information and scene information, please break down the given task into sub-tasks, each of which cannot be too complex, make sure that a single robot can do it. It can't be too simple either, e.g. it can't be a sub-task that can be done by a single step robot tool. Each sub-task in the output needs a concise name of the sub-task, which includes the robots that need to complete the sub-task. Additionally you need to give a 200+ word reasoning explanation on subtask decomposition and analyze if each step can be done by a single robot based on each robot's tools!

The output format is as follows, in the form of a JSON structure:

```
{
  "reasoning_explanation": xxx,
  "subtask_list": [
    {"robot_id": xxx, "subtask": xxx, "subtask_order": xxx},
    {"robot_id": xxx, "subtask": xxx, "subtask_order": xxx},
    {"robot_id": xxx, "subtask": xxx, "subtask_order": xxx},
  ]
}
```

Robot Information:

Robot in Scene: {Robot List}. Robot positional states:{Robotic Memory}. Robot available tools:{Robotic Tool Libraries}

Scene Information:

{Scene Graph}

The task to be completed is:{Global Task}. Your output answer:

Figure 36 Prompt for global task decomposition.

System Prompt for Agent-based Tool Calling

 # You are an expert assistant who can solve any task using tool calls.

You will be given a task to solve as best you can. To do so, you have been given access to some tools.

The tool call you write is an action: after the tool is executed, you will get the result of the tool call as an "observation".
This Action/Observation can repeat N times, you should take several steps when needed.

You can use the result of the previous action as input for the next action.
The observation will always be a string: it can represent a file, like "image_1.jpg".
Then you can use it as input for the next action. You can do it for instance as follows:

Observation: "image_1.jpg"

Action:
{{ "name": "image_transformer", "arguments": {{"image": "image_1.jpg"}}}}

To provide the final answer to the task, use an action blob with "name": "final_answer" tool. It is the only way to complete the task, else you will be stuck on a loop. So your final output should look like this:

Action:
{{"name": "final_answer", "arguments": {{"answer": "insert your final answer here"}}}} "arguments": "image.png"}}

 # Here are a few examples using notional tools:

Task: "What is the result of the following operation: 5 + 3 + 1294.678?“

Action:
{{ "name": "python_interpreter", "arguments": {{"code": "5 + 3 + 1294.678"}}}}
Observation: 1302.678

Action:
{{ "name": "final_answer", "arguments": "1302.678" }}

 # Above example were using notional tools that might not exist for you. You only have access to these tools:

{%- for tool in tools.values() %}
- {{ tool.name }}: {{ tool.description }}
  Takes inputs: {{tool.inputs}}
  Returns an output of type: {{tool.output_type}}
{%- endfor %}

 # Here are the rules you should always follow to solve your task:

1. ALWAYS provide a tool call, else you will fail.
2. Always use the right arguments for the tools. Never use variable names as the action arguments, use the value instead.
3. Call a tool only when needed: do not call the search agent if you do not need information, try to solve the task yourself.
If no tool call is needed, use final_answer tool to return your answer.
4. Never re-do a tool call that you previously did with the exact same parameters.

 # Now Begin! If you solve the task correctly, you will receive a reward of $1,000,000.

Task: {Subtask}
The tool you have used are: {Tool-Calling_History}
Observation: {Observation}
Your next action is:

Figure 37 Prompt for agent-based tool calling.