

OmniBrain-0: A Unified Agentic Model for Embodied Intelligence

Xiaomi Embodied Intelligence Team × Tsinghua University × University of Macau

Abstract

General-purpose embodied intelligence requires robots to coordinate diverse physical skills over long horizons, monitor task progress, and recover from execution failures. Specialist robot policies alone do not ensure task-level reliability, while powerful multimodal agents depend on costly long-context reasoning and external experience that compact models cannot readily reproduce. We introduce **OmniBrain-0**, a lightweight unified embodied agent that brings manipulation and navigation into a shared decision framework. OmniBrain-0 orchestrates heterogeneous skills through a single high-level model, integrating task understanding, planning, progress tracking, and feedback-driven recovery. To transfer these capabilities into a compact model, we develop a progressive framework combining embodied pretraining, domain-specialized agent distillation, and multi-teacher on-policy distillation (MOPD). It first distills powerful teachers’ interaction experience into manipulation and navigation experts. Crucially, MOPD consolidates their decision capabilities within OmniBrain-0 itself, so that unification occurs in the learned decision model rather than only at the tool interface. We further introduce *Rubric-Guided Recursive Self-Improvement* (RSI), a data flywheel in which a Meta-Agent turns execution evidence into refined rubrics, step-level reminders, and verified trajectories for domain-expert updates before their reconsolidation into OmniBrain-0 through MOPD. We evaluate OmniBrain-0 on LIBERO-Pro, RoboCasa365, RoboTwin 2.0, and R2R-CE Val-Unseen, covering manipulation and navigation. We further evaluate OmniBrain-0 on three real-world manipulation tasks, demonstrating its applicability to physical robot execution. Results show strong performance across both domains and substantial gains over the compact zero-shot backbone, supporting effective transfer and consolidation of agentic planning and recovery capabilities across task domains.

Date: October 20, 2026

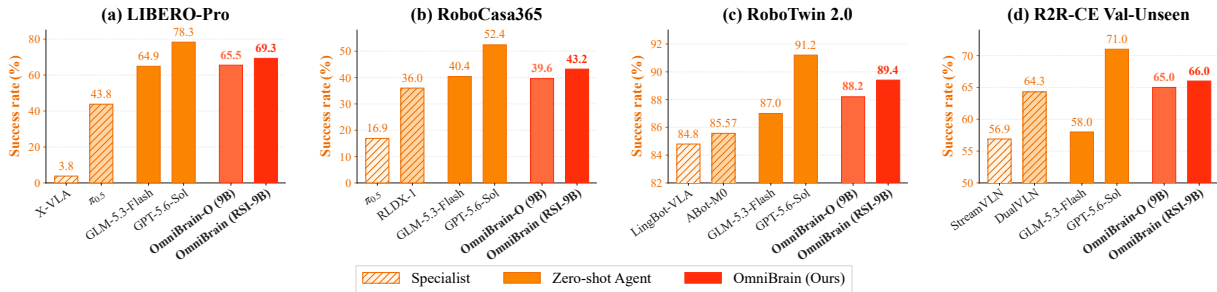


Figure 1 Cross-domain embodied-agent performance. Success-rate comparison across LIBERO-Pro, RoboCasa365, RoboTwin 2.0, and R2R-CE Val-Unseen. OmniBrain (Ours) unifies specialist manipulation and navigation skills in a compact agent, while RSI further enables iterative self-improvement from verified interaction feedback. Zero-shot teachers provide planning and recovery experience for capability distillation.

Contents

1	Introduction	3
2	Related Work	6
2.1	Specialist policies for manipulation and navigation	6
2.2	Tool-using embodied agents	6
2.3	Agent distillation and expert consolidation	6
2.4	Recursive self-improvement of embodied agents	6
3	Unified Embodied Agent Architecture	7
3.1	Problem Formulation and Overview	7
3.2	Skill Interfaces and Context Organization	7
3.3	Closed-Loop Planning and Failure Recovery	8
4	Embodied Capability Learning and Consolidation	9
4.1	Interaction Data Construction (Data Pipeline)	9
4.2	Embodied Pretraining	10
4.3	Domain-Specialized Agent Distillation	11
4.4	Multi-Expert Consolidation with MOPD	11
5	Rubric-Guided Recursive Self-Improvement	13
5.1	Rubric-Guided Reminder Policy	13
5.2	RSI Data Flywheel	14
6	Experiments and Analysis	16
6.1	Experimental Setup	16
6.2	Overall Performance	17
6.2.1	LIBERO-Pro	17
6.2.2	LIBERO-Plus	17
6.2.3	RoboCasa365	18
6.2.4	RoboTwin 2.0	19
6.2.5	R2R-CE	20
6.2.6	Spatial understanding and referring	21
6.2.7	Real-world manipulation	22
6.3	Ablation Studies	22
6.3.1	Ablation Study on the Reminder Policy	22
6.3.2	Ablation Study on the Data Processing Pipeline	23
6.3.3	Ablation Study on Experience Library	23
6.3.4	Ablation Study on Pretraining	23
6.3.5	Ablation Study on VLA skill	24
6.4	Qualitative Analysis	24
7	Conclusion and Limitations	29
	Contributions and Acknowledgments	30
A	Details of the Agentic System Design	36
B	Benchmarks and Evaluation Protocol	38
C	Datasets and Experimental Details	40
D	More Qualitative Examples	41

1 Introduction

General-purpose embodied intelligence requires diverse physical skills, long-horizon planning, skill composition, and closed-loop correction to fulfill task intent. Vision-language-action (VLA) models and vision-and-language navigation (VLN) policies advance manipulation and navigation [4, 22, 42, 46], but reliable individual skills do not guarantee task-level success. Under complex constraints, multi-stage dependencies, or distribution shifts, local action prediction may fail to maintain goal alignment, requiring feedback-driven reassessment and replanning [50, 60]. Large-model-based embodied agents offer a promising approach by exposing perception, specialist policies, and motion controllers as tools. A high-level model interprets instructions, decomposes tasks, orchestrates skills, and uses execution feedback to track progress and recover from failures, while low-level policies retain their physical interaction expertise [16, 28, 60]. This division of labor enables goal- and feedback-driven execution, but reliable coordination also requires the high-level model to reason about skill applicability, capability boundaries, and handoff states [15]. These shared task-level requirements motivate a common high-level decision mechanism for manipulation and navigation, spanning task understanding, skill orchestration, and feedback-driven adaptation.

However, successful task execution by powerful models does not by itself establish how their planning and recovery capabilities can be learned by a compact, unified agent. These models combine contextual reasoning, experience, and environmental feedback [15, 60], and compact agents must learn from this interaction experience how to decompose tasks, coordinate skills, track progress, and adapt to execution outcomes across domains. Repeated reasoning by powerful models during long-horizon interaction incurs computational overhead and decision latency, constraining deployment efficiency [9, 27, 29, 30, 33]. Distilling these capabilities into compact agents offers a way to improve deployment efficiency, but effective interaction data remain limited, particularly for states that require feedback-driven correction. Useful supervision must therefore preserve how tasks progress, how skills are coordinated, and how decisions change in response to execution feedback, including during recovery from failure.

Alongside effective learning and consolidation methods, we therefore need a scalable data construction mechanism that preserves informative task progression and recovery while continually expanding coverage of execution challenges encountered during interaction [7, 32]. These requirements motivate a unified learning framework that combines scalable interaction data construction with capability transfer and consolidation, enabling a compact agent to coordinate planning and recovery across manipulation and navigation.

First, capability transfer aims to learn effective decision behaviors from teacher interactions rather than reproduce teachers’ language outputs or reasoning processes [19]. Because embodied decisions alter the environment and subsequent observations, students must learn to interpret execution feedback, track progress, and adapt when plans become invalid. Fixed demonstrations alone may not provide sufficient supervision for appropriate decisions in states encountered during autonomous execution [1, 16]. Second, acquiring specialized capabilities and integrating them into a unified agent are distinct problems. Manipulation and navigation differ in scene distributions, skill interfaces, and task structures, so expertise learned by specialized planners must be further integrated for joint deployment [35].

Therefore, we introduce **OmniBrain-0**, a **lightweight, unified embodied agent for manipulation and navigation** (Figure 2). OmniBrain-0 integrates specialist policies and tools through a shared skill-invocation framework. A compact high-level planner performs task understanding, long-horizon planning, skill orchestration, state tracking, and failure recovery. Conditioned on observations, instructions, and interaction history, it selects skills and adapts to execution feedback, preserving low-level specialization while learning task decomposition and adaptive correction.

To realize these capability transformations, we develop a progressive training framework comprising *shared embodied initialization*, *specialized capability acquisition*, and *cross-domain capability integration*. First, we combine embodied, question-answering, and general language data with limited long-horizon interaction data to establish a shared initialization for a compact model. We then train manipulation and navigation planners from this initialization using quality-filtered interaction trajectories generated by multiple powerful models. By learning observation–decision–feedback relationships from canonical execution and successful recovery, these planners acquire the teachers’ planning and correction behaviors. Finally, we adopt multi-teacher on-policy distillation (MOPD) [35], using the domain experts to supervise student-generated decision sequences and integrate specialized capabilities into a unified student initialized from the shared pretrained model. The

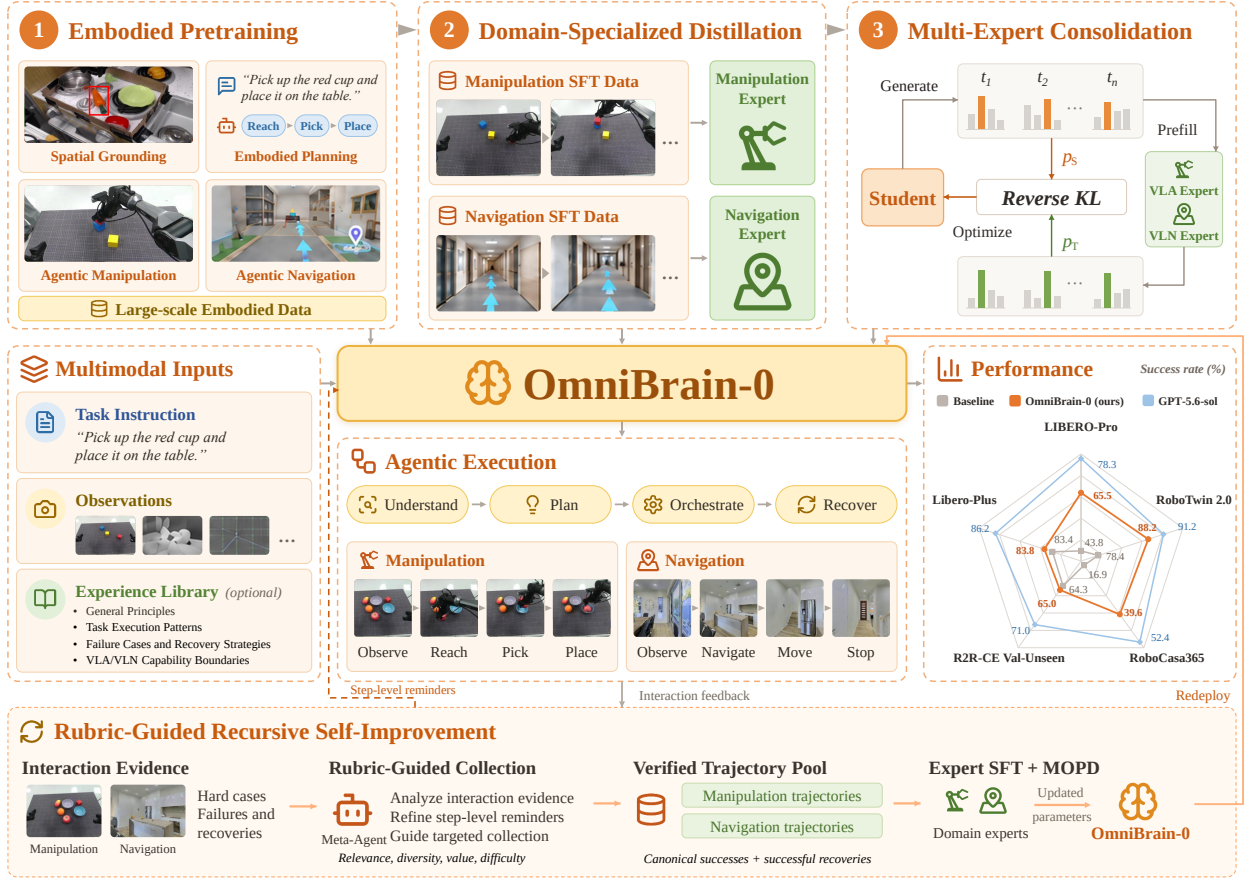


Figure 2 Overview of OmniBrain-0. *Top:* Three-stage training: embodied pretraining, domain-specialized distillation into manipulation and navigation experts, and multi-expert consolidation (MOPD) into a single compact agent. *Bottom:* Given a task instruction, observations, and an optional experience library, OmniBrain-0 understands, plans, orchestrates skills, and recovers from failures across manipulation and navigation. The radar chart reports success rates (%) on five benchmarks, comparing OmniBrain-0 with the baseline policy ($\pi_{0.5}$ for LIBERO-Pro, LIBERO-Plus, RoboTwin 2.0, and RoboCasa365; InternVLA-N1 for R2R-CE Val-Unseen) and the zero-shot teacher GPT-5.6-Sol.

resulting OmniBrain-0 thus unifies manipulation and navigation at the level of learned high-level decisions as well as the skill-invocation interface.

To support the scalable construction of effective interaction data, we introduce *Rubric-Guided Recursive Self-Improvement* (RSI), a closed-loop mechanism connecting data collection, execution-guidance refinement, and model training (Section 5). OmniBrain-0’s autonomous interactions reveal challenges in task progression, skill coordination, and failure recovery. A Meta-Agent analyzes execution evidence from successful and failed trajectories, derives execution rubrics, and refines step-level reminders to guide subsequent interaction and collection. Candidate reminders are validated through interaction with model parameters fixed, while successful, failed, and inefficient executions inform further refinement. Environment-verified canonical successes and successful recoveries are curated and structured to augment manipulation and navigation training pools, which are then used to update domain experts. MOPD consolidates the updated experts’ capabilities into the unified model, whose subsequent interactions provide evidence for the next round of guidance refinement and data collection. This process connects challenge discovery, execution guidance, trajectory verification, and capability learning, supporting the continued expansion of effective embodied interaction data. We evaluate OmniBrain-0 on manipulation and navigation benchmarks, with additional assessments of spatial understanding, spatial referring, and real-world manipulation.

Our main contributions are summarized as follows:

- **A unified agentic model for manipulation and navigation.** We introduce OmniBrain-0, a compact high-level agent that coordinates heterogeneous perception, manipulation, navigation, and motion skills through a shared decision model for planning, orchestration, and recovery.
- **Scalable interaction data construction for embodied capability transfer.** We construct structured trajectories that preserve observation–decision–feedback dependencies and recovery, and develop a closed-loop mechanism that combines challenge-driven collection, execution-guidance refinement, and environment verification to expand training data for compact domain-specialized agents.
- **Progressive specialization and multi-expert consolidation.** Embodied pretraining, domain-specialized distillation, and MOPD unify manipulation and navigation expertise while preserving low-level specialization.

2 Related Work

2.1 Specialist policies for manipulation and navigation

VLA models such as π_0 [5], $\pi_{0.5}$ [4], and LingBot-VLA [48, 49] learn multitask control from large-scale robot data and vision-language pretraining. These policies provide a foundation for reusable manipulation skills [42], while task-level coordination must also account for sub-goal dependencies, execution progress, and the conditions required by subsequent skills. Yet even $\pi_{0.5}$ and LingBot-VLA 2.0 degrade substantially on long-horizon tasks under object and initial-position shifts [49]. For navigation, NavGPT-2 [66] couples language reasoning with a navigation policy, and DualVLN (InternVLA-N1) [46] connects high-level planning to low-level execution in a dual-system design. Manipulation and navigation nevertheless involve different observations, action interfaces, and temporal structures, motivating a shared decision layer that can coordinate domain-specific execution. Rather than replacing these policies, OmniBrain-0 invokes them as skills and learns a shared high-level decision model across both domains.

2.2 Tool-using embodied agents

SayCan [2] and Code as Policies [28] ground task decomposition in executable skills, and Inner Monologue [16] feeds environmental feedback back into planning. This division between task reasoning and skill execution provides a basis for composing heterogeneous capabilities, with feedback enabling the planner to reassess whether intermediate outcomes satisfy the current goal [34]. Agent harnesses, including Harness VLA [60], RoboHarness [15], and Pigey [14], add state tracking, outcome verification, and recovery through agent loops. However, their orchestration relies on repeated reasoning by large models at deployment time. OmniBrain-0 instead internalizes planning, progress tracking, and recovery into a compact decision model learned from interaction experience. Building on this execution paradigm, we retain the tool interfaces and feedback loop while using curated interaction trajectories to supervise the high-level decisions made within it.

2.3 Agent distillation and expert consolidation

AgentTuning [55] and STEVE-2 [62] transfer agent capabilities via trajectory tuning and hierarchical distillation, but fixed teacher trajectories may not cover states encountered by students. For embodied capability transfer, useful supervision must preserve the relationship between observations, tool choices, and execution outcomes, including corrective decisions that lead to successful recovery. On-policy distillation mitigates this mismatch by supervising student-generated sequences [1, 31]; TurnOPD [68] adds turn-aware supervision for agents, and MOPD [35] merges domain-specialized teachers. Such consolidation complements trajectory-based specialization: the latter acquires domain-specific decision behaviors, while the former integrates complementary expert distributions within shared model parameters. In embodied settings, each decision alters subsequent observations and skill applicability; OmniBrain-0 applies multi-teacher distillation to consolidate manipulation and navigation expertise, including planning and recovery, into a single model.

2.4 Recursive self-improvement of embodied agents

Recursive self-improvement (RSI) uses successive interactions to refine an agent’s guidance or learned policy. Reflexion [40] stores verbal reflections in episodic memory, while ACE [59] evolves contextual playbooks through reflection and curation. ReST meets ReAct [3] iteratively trains agents on feedback-filtered trajectories. In robotics, EmbodiRSI [24] targets corrective data collection at policy weaknesses, and AdaRSI [43] combines harness refinement with selective executor fine-tuning. RE-0 [56] verifies local teacher corrections before distilling them into embodied code-generating agents. OmniBrain-0 connects rubric-guided reminder refinement with verified trajectory curation, domain-expert SFT, and MOPD for a decision model shared across manipulation and navigation. Reminders are validated without changing model parameters; subsequent training updates the agent, whose new interactions guide the next improvement round. This cycle retains successful recoveries while using failed episodes for diagnosis.

3 Unified Embodied Agent Architecture

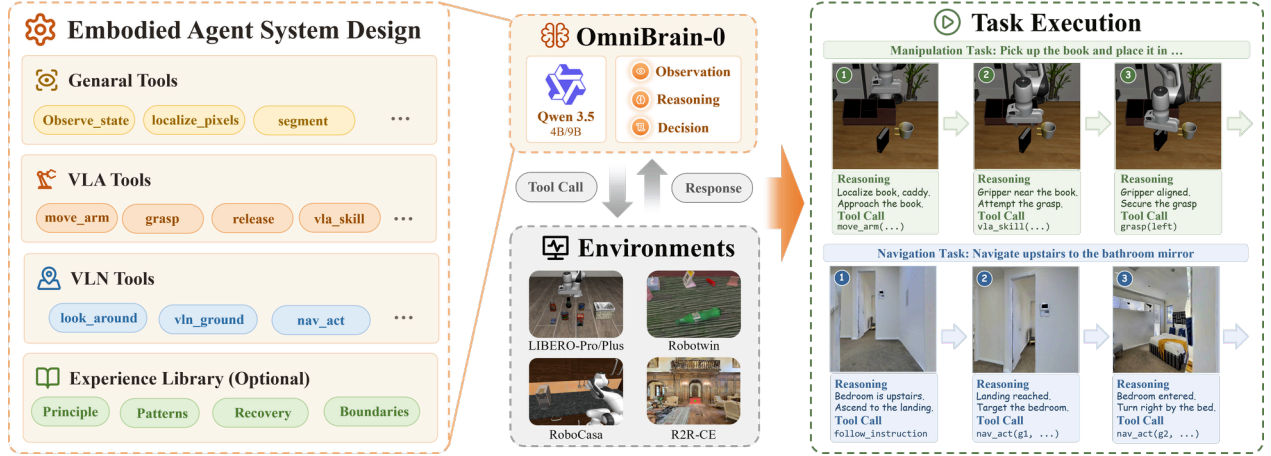


Figure 3 Agentic execution framework and distillation data pipeline. *Left:* General, VLA, and VLN tools with an experience library. *Center:* Expert models collect trajectories through closed-loop interaction with task environments. *Right:* Trajectory normalization, quality control, structured rewriting, and consistency filtering convert expert interaction records into an agentic training dataset.

3.1 Problem Formulation and Overview

We consider language-conditioned manipulation and navigation tasks, with tool invocation serving as a unified high-level action representation. Given a task instruction g , OmniBrain-0 selects tools based on current observations and interaction history, continually updating its decisions through environmental feedback. At decision step t , the observation and interaction context are defined as

$$o_t = (I_t, q_t, p_t), \quad \mathcal{H}_t = (h_t, \mathcal{M}), \quad (1)$$

where I_t denotes visual observations, q_t the robot’s proprioceptive state, and p_t task-status information exposed by the environment. The within-episode interaction history is denoted by h_t , while $\mathcal{M}$ represents an optional experience library. Decisions are conditioned only on information available at the current or previous steps, with the experience library providing priors for task organization and skill use.

Let $\mathcal{P}_e$ denote the available tool set, the high-level policy π_θ generates a tool identifier $k_t \in \mathcal{P}_e$ and invocation parameters ξ_t :

$$\begin{aligned} x_t &= (g, o_t, \mathcal{H}_t), \\ u_t &= (k_t, \xi_t) \sim \pi_\theta(\cdot \mid x_t), \quad k_t \in \mathcal{P}_e. \end{aligned} \quad (2)$$

where u_t the executed skill invocation. The selected tool performs the corresponding perception or control operation and returns a new observation o_{t+1} and execution feedback f_{t+1} , which condition subsequent decisions. The objective is to fulfill the instruction within a given execution budget, with completion determined by the environment’s task criteria. This formulation provides a unified high-level tool interface for manipulation and navigation, while preserving their distinct low-level action spaces and execution durations.

3.2 Skill Interfaces and Context Organization

The agentic execution framework of OmniBrain-0 follows the system design of Harness VLA [60], with its tools and experience library summarized in Figure 3 (left). OmniBrain-0 uses the general-purpose multi-modal model Qwen3.5-4B/9B [38] as a shared agent, decoupling task-level decision-making from specialized tool execution. The agent performs instruction understanding, long-horizon task decomposition, tool orchestration, and failure recovery, while execution modules provide perception, manipulation, navigation, and motion-control capabilities. A unified invocation-and-feedback interface connects the agent and execution

modules in a closed loop, enabling the same decision model to coordinate heterogeneous tools with different inputs and control mechanisms.

Unified tool abstraction. We encapsulate heterogeneous execution capabilities as parameterized semantic tools. Perception tools provide scene understanding and target localization; VLA tools handle manipulation and contact interactions; navigation tools support goal reaching through VLN policies or geometric navigators; and motion-control tools provide explicit pose and actuator adjustments. Each tool retains its native conditioning and low-level implementation while exposing a unified invocation-and-feedback interface. Thus, the agent shares a common high-level decision mechanism and interaction protocol rather than a single low-level control policy.

Context-driven tool orchestration. The agent jointly processes task instructions, current observations, and historical feedback to select subsequent tools and determine their invocation parameters. Interaction history supports task progress tracking, while an optional experience library provides priors on task organization, tool composition, and recovery strategies. Experience guides task-level organization, whereas online perception grounds tool invocation in the current execution state.

3.3 Closed-Loop Planning and Failure Recovery

At each turn, the agent selects one tool and makes the next decision only after receiving its execution feedback. High-level tool invocations are executed sequentially and connected through execution outcomes. The agent continuously updates its decisions based on observations and feedback, adapting target localization, invocation parameters, or tool selection when execution deviates from expectations. Detailed tool definitions and implementation specifications are provided in the Appendix A.

4 Embodied Capability Learning and Consolidation

Figure 5 summarizes our three-stage capability transfer framework: embodied pretraining establishes a shared initialization (Section 4.2), domain-specialized distillation transfers expert interaction experience into specialized decision-making capabilities (Section 4.3), and multi-teacher on-policy distillation consolidates these capabilities into a unified shared model (Section 4.4). This process separates capability acquisition from cross-domain integration, allowing the shared agent to inherit complementary planning, tool orchestration, and recovery capabilities while preserving the specialization of low-level execution tools.

4.1 Interaction Data Construction (Data Pipeline)

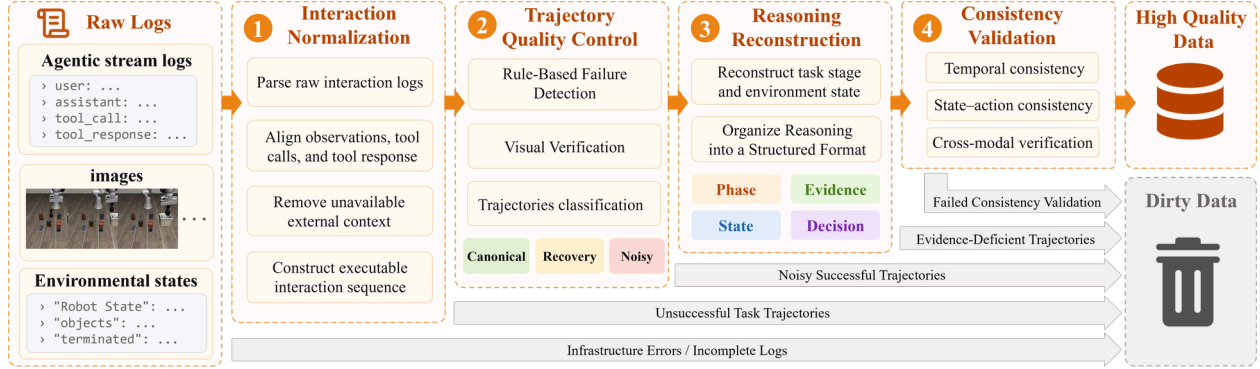


Figure 4 Interaction data construction. Raw interaction logs, images, and environment states pass through interaction normalization, trajectory quality control, reasoning reconstruction, and consistency validation. Canonical successes and successful recoveries form the training data; incomplete, unsuccessful, noisy, evidence-deficient, or inconsistent trajectories are discarded.

Expert Interaction Trajectories. We collect expert demonstration data by leveraging multiple powerful models, such as GPT-5.6-Sol, to execute manipulation and navigation tasks in closed-loop environments. Experts invoke tools within an execution framework that a Meta-Agent iteratively refines, continually adjusting decisions based on task objectives, online observations, and execution feedback. Each trajectory is organized as a temporally ordered, multi-turn multimodal sequence:

$$\tau = (g, x_0, y_0, u_0, f_1, x_1, \dots, y_{T-1}, u_{T-1}, f_T, x_T), \quad (3)$$

where y_t denotes the expert’s decision text, u_t the executed skill invocation, and f_{t+1} the corresponding execution feedback; x_t follows the definition in Section 3.1. We preserve the temporal relationships among observations, decisions, and feedback, ensuring that each decision is grounded in information available at that step.

Figure 4 details the conversion of expert interaction records into training trajectories through four steps: *interaction normalization*, *trajectory quality control*, *reasoning reconstruction*, and *consistency validation*. The pipeline aligns raw logs, images, and environment states into supervision compatible with the student’s observations and tool interfaces. It preserves the observation–action–feedback structure and retains both canonical successes and successful recoveries, while removing task-irrelevant information and unreliable trajectories. Dataset composition and splits are provided in the experimental setup and appendix.

Interaction Normalization. We parse raw interaction logs and reconstruct each episode as a temporally ordered sequence of *reasoning–tool call–tool feedback* units, retaining synchronized images and robot states at each decision step. Tool calls and feedback are normalized into a unified format, with visual observations aligned to the corresponding interaction steps. Context unavailable to the student is removed, while infrastructure errors and incomplete logs are filtered out to obtain executable interaction sequences.

Trajectory Quality Control. We use rule-based failure detection and visual verification to assess task outcomes and execution quality, discarding unsuccessful task trajectories. Successful trajectories do not necessarily provide equal training value. Canonical successful trajectories provide reliable demonstrations of standard task execution, while failure-recovery trajectories offer valuable state-dependent supervision by capturing how the agent diagnoses and corrects execution failures. In contrast, blind repetitions without new observations or state information may reinforce inefficient action patterns. We therefore categorize successful trajectories into three types:

- **Canonical trajectories:** The task is completed without tool failures, with perception, manipulation, and termination proceeding as expected.
- **Recovery trajectories:** One or more grasping, contact, or other execution failures occur, but the agent subsequently recovers through re-localization, gripper adjustment, re-grasping, or other corrective actions and eventually receives an environment success signal.
- **Noisy trajectories:** The agent repeatedly executes the same action without acquiring new observations or state information, reflecting blind retries rather than feedback-driven decision making.

Both canonical and recovery trajectories are retained, while noisy trajectories are discarded. Canonical trajectories provide standard execution demonstrations, while recovery trajectories capture the failure–diagnosis–correction–continuation process.

Reasoning Reconstruction. The raw reasoning text generated by the expert model may introduce unnecessary linguistic variation. We therefore rewrite each reasoning step into a structured representation using only information available at the corresponding decision step, following two principles: **state grounding**, which derives the state from tool outputs, robot states, and images, and **evidence traceability**, which excludes future information. Specifically, each reasoning step is represented by four fields:

- **Phase:** the current stage of task execution, such as localization, grasping, transport, placement, release, success verification, or recovery.
- **State:** the current interaction state, including the held object, gripper and grasp status, target and destination locations, and task completion status.
- **Evidence:** relevant visual observations and tool feedback supporting the current state assessment.
- **Decision:** the next tool to be invoked and its action description determined by the actual tool parameters.

We generate these fields by replaying each trajectory chronologically: tool feedback first updates the interaction state, while visual auditing verifies information not reliably available from tool outputs, such as object identity, grasp status, and placement. The *Phase*, relevant *Evidence*, and *Decision* are then determined from the resulting state and the actual next tool call. Failed actions are marked as recovery operations, while environment-confirmed completion produces the terminal state. For long-horizon tasks, the *State* additionally tracks the current sub-goal and its completion progress. The resulting structured reasoning is independent of the teacher’s wording and remains one-to-one aligned with each tool call.

Consistency Validation. We finally check temporal consistency, state–action consistency, and agreement between visual observations and textual evidence. These checks verify consistency among the state, evidence, phase, and tool call, including target/destination localization, action–decision alignment, and environment-verified termination. Evidence-deficient trajectories and those failing consistency validation are discarded, while canonical successes and successful recovery trajectories are retained as high-quality training data.

4.2 Embodied Pretraining

As shown in Figure 5 (left), we pretrain on a mixture of embodied question-answering data and long-horizon agent interaction data to establish a shared basis for embodied capabilities in manipulation and navigation. Embodied question answering provides supervision for scene understanding and task-related knowledge, while long-horizon interactions guide the model to organize decisions and use feedback in multi-turn tasks through temporal relationships among instructions, observations, tool invocations, and execution feedback. These

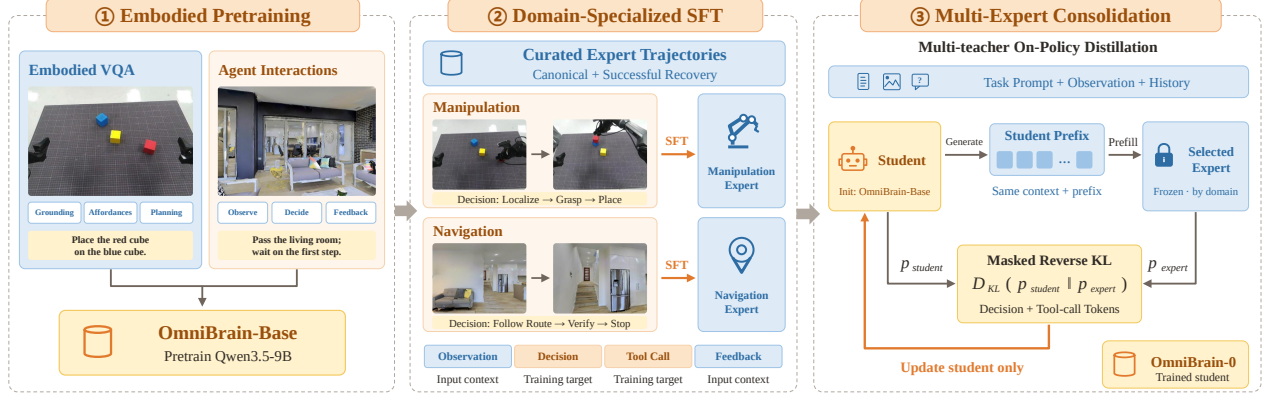


Figure 5 Three-stage training framework of OmniBrain-0. *Left:* Embodied VQA and agent interaction data establish the shared OmniBrain-Base initialization. *Middle:* Curated expert trajectories, including canonical successes and successful recoveries, supervise domain-specialized manipulation and navigation agents. *Right:* MOPD consolidates the frozen domain experts into a unified student through masked reverse-KL supervision on student-generated decision prefixes, updating only the student.

complementary sources connect static embodied understanding with dynamic interactive decision-making, providing shared knowledge and interaction priors for subsequent specialization. The resulting model, denoted as **OmniBrain-Base** with parameters θ_{base} , initializes both the domain teachers and the unified student.

4.3 Domain-Specialized Agent Distillation

As shown in Figure 5 (middle), we initialize domain-specialized teachers for manipulation and navigation from OmniBrain-Base and train them through supervised hard distillation on expert interaction trajectories. Each teacher learns goal understanding, skill orchestration, and recovery decisions appropriate to its domain’s task structure. These teachers are high-level agent models: they learn to organize and use skills rather than generate low-level actions.

Let ϕ_k denote the parameters of teacher k and $\mathcal{D}_k$ its training dataset. Given a context x comprising task instructions, observations, interaction history, and skill definitions, we serialize the teacher’s decision text and executed invocation into a target sequence z and minimize the autoregressive negative log-likelihood:

$$\mathcal{L}_{\text{expert}}^{(k)} = -\mathbb{E}_{(x,z) \sim \mathcal{D}_k} \left[\frac{1}{|z|} \sum_{j=1}^{|z|} \log \pi_{\phi_k}(z_j \mid x, z_{<j}) \right], \quad (4)$$

$$\phi_k^{(0)} = \theta_{\text{base}}.$$

Supervision is applied only to the agent’s decision text and tool invocations, while environmental observations and tool feedback serve as conditioning information. By jointly learning from standard execution and recovery trajectories, specialization transforms expert interaction experience into observation-conditioned, task-level decision-making capabilities, yielding domain teachers for subsequent cross-domain integration.

4.4 Multi-Expert Consolidation with MOPD

As shown in Figure 5 (right), we adopt multi-teacher on-policy distillation (MOPD) [35] to integrate domain teachers’ distributional supervision over student-generated decision prefixes. The unified student is initialized from OmniBrain-Base, and the domain-specialized agents obtained in Section 4.3 form a fixed teacher set:

$$\theta^{(0)} = \theta_{\text{base}}, \quad \mathcal{T} = \{\pi_{\phi_k}\}_{k=1}^K, \quad (5)$$

where K is the number of teachers. The student and teachers share the same model architecture and vocabulary. Each teacher provides domain-specific supervision, jointly optimizing the unified student.

Teacher Supervision on Student Prefixes. We sample domain k with weight ρ_k , where $\rho_k \geq 0$ and $\sum_k \rho_k = 1$. For a domain context $x \sim \mu_k$, a student decision sequence is sampled as $\hat{z} \sim \pi_{\bar{\theta}}(\cdot | x)$, where $\bar{\theta}$ denotes the student parameters used for sampling. The teacher provides token-level distributional supervision conditioned on the same context and the student-generated prefix:

$$\begin{aligned} p_{\theta,j}(v) &= \pi_{\theta}(v | x, \hat{z}_{<j}), \\ q_{k,j}(v) &= \pi_{\phi_k}(v | x, \hat{z}_{<j}), \quad v \in \mathcal{V}, \end{aligned} \tag{6}$$

where $\mathcal{V}$ is the shared vocabulary. At this stage, distribution alignment is conditioned on student-generated prefixes, allowing teacher supervision to adapt to the student’s generation behavior.

Joint multi-domain distillation objective. We minimize the token-level reverse Kullback–Leibler (KL) divergence from the student to the corresponding expert:

$$\begin{aligned} \mathcal{L}_{\text{MOPD}}(\theta) &= \sum_{k=1}^K \rho_k \mathbb{E}_{\substack{x \sim \mu_k \\ \hat{z} \sim \pi_{\bar{\theta}}(\cdot | x)}} \left[\frac{1}{M} \sum_{j=1}^{|\hat{z}|} m_j D_{\text{KL}}(p_{\theta,j} \| q_{k,j}) \right], \\ M &= \sum_{j=1}^{|\hat{z}|} m_j > 0, \end{aligned} \tag{7}$$

where $m_j \in \{0, 1\}$ marks only valid agent decisions and tool-invocation outputs. The divergence is defined as

$$D_{\text{KL}}(p_{\theta,j} \| q_{k,j}) = \sum_{v \in \mathcal{V}} p_{\theta,j}(v) \log \frac{p_{\theta,j}(v)}{q_{k,j}(v)}. \tag{8}$$

This objective weights distributional discrepancies by student prediction probabilities, using the decision preferences of different teachers to constrain the student’s current output distribution. Teacher parameters and sampled prefixes remain fixed during optimization, and only the unified student parameters θ are updated. By decoupling specialization from cross-domain integration, we consolidate manipulation and navigation expertise into the lightweight **OmniBrain-0**, which orchestrates cross-domain skills with a single set of high-level agent parameters.

5 Rubric-Guided Recursive Self-Improvement

We introduce *Rubric-Guided Recursive Self-Improvement* (RSI) to support the scalable construction of effective embodied interaction data. A Meta-Agent derives step-level guidance from interaction evidence, while verified trajectories augment domain-expert SFT. MOPD consolidates the updated experts into OmniBrain-0, whose subsequent interactions support the next round of improvement.

5.1 Rubric-Guided Reminder Policy

Rubric-based policy update. Let $\mathcal{D}_k$ denote the self-collected trajectories at refinement round k , following the interaction representation in Equation 3. The initial collection includes both successful and failed manipulation and navigation episodes. A verifier assesses goal attainment and execution efficiency, providing outcome evidence for the Meta-Agent to analyze, diagnose, and revise execution guidance. Successful trajectories reveal effective task progression, while failed and inefficient trajectories expose unresolved execution challenges. The Meta-Agent relates these outcomes to the corresponding observations, decisions, and feedback, organizing relevant trajectory patches and rubric candidates into a shared set of robotics rubrics $\mathcal{U}_k$. New evidence supports adding or revising rubrics, while recurring supporting evidence reinforces applicable criteria. These rubrics capture principles for task prerequisites, skill ordering, progress verification, and recovery. Figure 6 summarizes rubric-based policy update and reminder-policy validation.

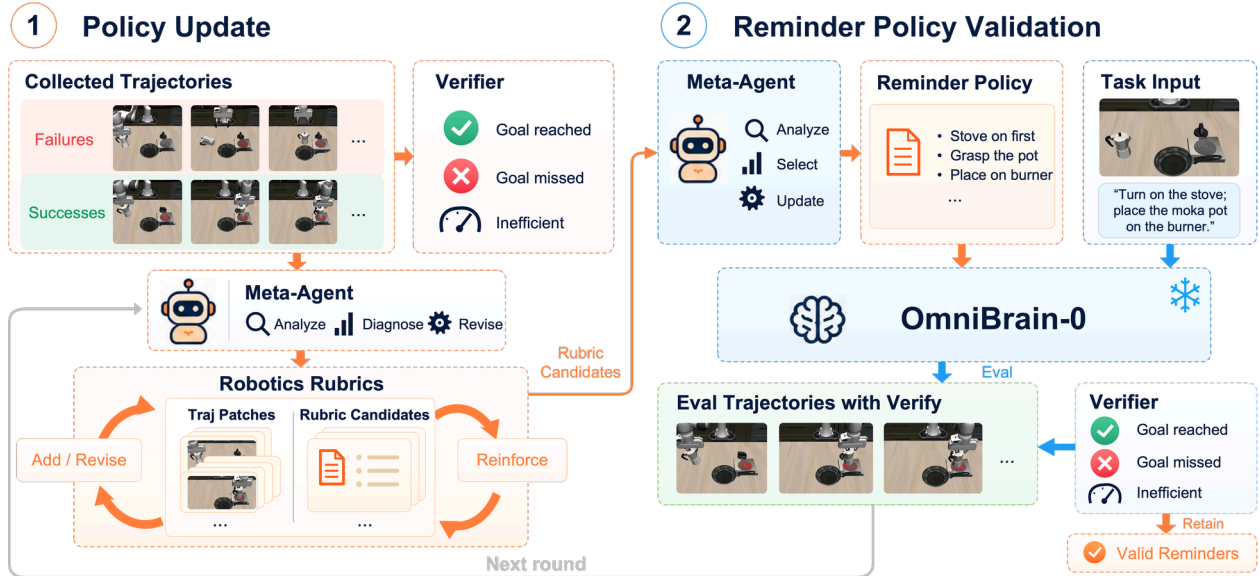


Figure 6 Rubric-based policy update and reminder-policy validation. *Left:* A verifier assesses collected successful and failed trajectories. The Meta-Agent analyzes execution evidence and adds, revises, or reinforces robotics rubrics supported by trajectory patches. *Right:* Rubric candidates inform the reminder policy, which conditions OmniBrain-0 together with the task input while model parameters remain fixed. Verified evaluation trajectories provide feedback for the next round, and effective reminders are retained as valid reminders.

Given the collected evidence and current rubrics, the update is expressed as

$$\mathcal{U}_{k+1} = \mathcal{A}(\mathcal{D}_k, \mathcal{U}_k), \quad \mathcal{R}_{k+1} = \mathcal{G}(\mathcal{U}_{k+1}), \quad (9)$$

where $\mathcal{A}$ denotes rubric analysis and revision, and $\mathcal{G}$ denotes reminder-policy construction, including candidate selection and validation. The rubrics organize execution criteria; the reminder policy translates selected criteria into concise guidance for the agent. The update changes the guidance supplied by the harness and retains the tool interfaces in Section 3.2.

Step-level reminder guidance. The Meta-Agent analyzes rubric candidates, selects relevant guidance, and updates the reminder policy. At every high-level decision step, the harness inserts a reminder before

OmniBrain-0 generates its next tool invocation. Using the context $x_t = (g, o_t, \mathcal{H}_t)$ from Section 3.1, the reminder and conditional decision are

$$r_t = \mathcal{R}_{k+1}(\mathcal{U}_{k+1}, x_t), \quad u_t \sim \pi_\theta(\cdot | x_t, r_t), \quad (10)$$

where r_t provides task- and state-dependent guidance on the prerequisites, action order, or execution evidence relevant to the current decision. Task input and reminder guidance jointly condition tool selection and invocation parameters. Subsequent observations and feedback update the interaction context, allowing guidance to remain aligned with task progress throughout the episode.

Reminder-policy validation. OmniBrain-0 executes evaluation trajectories under the candidate guidance with its parameters fixed. The verifier assesses goal attainment and execution efficiency, and the Meta-Agent uses the resulting evidence to identify valid reminders and revise ineffective guidance. Verified evaluation trajectories, including successful and failed executions, are returned for the next round of rubric-based policy update. Effective reminders are retained across rounds, enabling execution guidance to accumulate from observed interaction outcomes. The verified successful trajectories also provide candidates for the training-data process in Section 5.2. This stage refines the reminder policy through interaction feedback; model-parameter updates are performed through expert training and MOPD below.

5.2 RSI Data Flywheel

The RSI data flywheel connects reminder-guided interaction, verified trajectory collection, and the training framework in Section 4. As illustrated in Figure 7, the Meta-Agent coordinates analysis, guidance, curation, and improvement. Collected experience refines the reminder policy and enriches expert training, while the updated unified model generates the next round of interaction evidence.

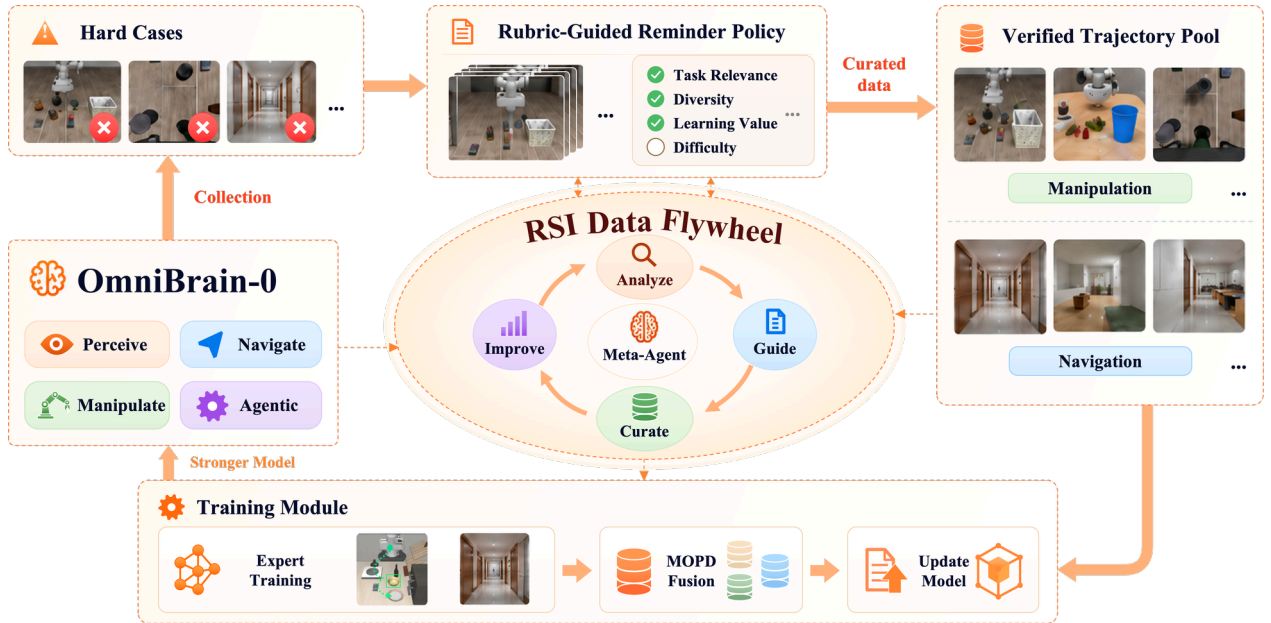


Figure 7 RSI data flywheel. OmniBrain-0 collects challenging cases that inform the rubric-guided reminder policy and trajectory curation. The Meta-Agent coordinates analysis, guidance, curation, and improvement, with task relevance, diversity, learning value, and difficulty considered during selection. Verified trajectories form manipulation and navigation pools for expert training. MOPD consolidates the updated experts into the unified model, which is redeployed for the next collection round.

Rubric-guided trajectory collection. OmniBrain-0 interactions reveal challenging cases that expose limitations in task progression, skill coordination, or recovery. These cases guide subsequent collection and reminder-policy refinement. The rubric-guided reminder policy from Section 5.1 supports further interaction on the identified challenges, while trajectory selection considers task relevance, diversity, learning value,

and difficulty. Together, guidance and curation focus collection on informative experience aligned with the current model’s capability needs.

Verified trajectory pool. Collected trajectories are verified and curated into manipulation and navigation pools. Failed and inefficient executions remain available for diagnosis, while expert training retains environment-verified canonical successes and successful recoveries under Section 4.1. The existing data pipeline normalizes interaction records and constructs consistent, state-grounded supervision, preserving the temporal alignment of observations, decisions, tool invocations, and feedback. This connects challenging cases to useful training experience through verified execution and data curation.

Let $\mathcal{T}_k^{(d)}$ denote trajectories collected for domain d at round k , and let $\mathcal{Q}$ denote verification, curation, and training-data processing. The selected trajectories augment the corresponding expert corpus:

$$\mathcal{D}_{\text{SFT},k+1}^{(d)} = \mathcal{D}_{\text{SFT},k}^{(d)} \cup \mathcal{Q}\left(\mathcal{T}_k^{(d)}\right), \quad (11)$$

where d indexes manipulation or navigation. Following Section 4.3, decision text and tool invocations are prediction targets, while observations and environment feedback provide conditioning information. The curated trajectories therefore return to the same expert-training stage as the original interaction data.

Expert training and MOPD consolidation. Domain-specialized experts are further trained on the augmented corpora using Equation 4. The resulting experts then provide domain-specific distributional supervision for MOPD under Equation 7. The MOPD fusion stage in Figure 7 follows this distillation procedure, transferring the refined experts’ decision capabilities into the unified OmniBrain-0. The updated model is redeployed for further collection, so subsequent challenges reflect its current capabilities.

Iterative improvement. New interactions supply the next evidence batch $\mathcal{D}_{k+1}$ for rubric revision and reminder-policy validation. Verified trajectories extend the expert corpora through Equation 11, while expert SFT and MOPD convert the selected experience into updated model parameters. The flywheel thus connects *analyze*, *guide*, *curate*, and *improve*: interaction evidence informs guidance, guidance supports collection, curation yields verified supervision, and training returns an updated model to interaction. Reminder-policy refinement and model learning remain connected through this shared flow of execution evidence and training data.

6 Experiments and Analysis

In this section, we first introduce the experimental setup, including the evaluation benchmarks and implementation details. We then analyze the performance of OmniBrain-0 and visualize its task execution. Ablation studies and detailed qualitative examples are presented in Sections 6.3 and 6.4.

6.1 Experimental Setup

Evaluation Benchmarks. We evaluate OmniBrain-0 on four manipulation benchmarks, one navigation benchmark, two spatial understanding and referring benchmarks, and a real-world suite: (1) **LIBERO-Pro** [67]: VLA evaluation under systematic perturbations of objects, states, instructions, and environments; (2) **LIBERO-Plus** [13]: Manipulation robustness across 10,030 task variants spanning seven perturbation dimensions; (3) **RoboCasa365** [36]: Household manipulation with 365 tasks for multitask and long-horizon evaluation; (4) **RoboTwin 2.0** [8]: Bimanual manipulation with randomized tasks and environments; (5) **R2R-CE** [23]: Continuous-environment vision-and-language navigation for indoor instruction following; (6) **EmbSpatial-Bench** [12]: Egocentric spatial-relation understanding from MP3D, AI2-THOR, and ScanNet images; (7) **RefSpatial-Bench** [65]: Spatial referring for object localization, placement, and unseen relation combinations; and (8) **Real-world**: Manipulation tasks include block stacking, apple placement, and drawer opening.

For manipulation, we report task-level success rates using the environment’s success conditions. LIBERO-Pro covers task and swap variants across the Spatial, Object, Goal, and LIBERO-10 suites, while LIBERO-Plus assesses robustness across seven perturbation dimensions. RoboCasa365 evaluation uses the Target-50 split, with results reported separately for Atomic-Seen, Composite-Seen, and Composite-Unseen tasks. RoboTwin 2.0 evaluates bimanual task execution under both Clean and Randomized conditions. For R2R-CE, we evaluate on a 100-episode subset of Val-Unseen, denoted as **rand100**, and report Success Rate (SR), Success weighted by Path Length (SPL), Oracle Success Rate (OSR), and Navigation Error (NE). The spatial benchmarks complement these closed-loop evaluations by measuring visual spatial understanding and referring accuracy, while the real-world suite measures task completion over repeated physical trials. Detailed task splits, episode counts, and evaluation protocols are provided in Appendix B.

Implementation Details. To collect training data for OmniBrain-0, we use multiple expert models, including GPT-5.6-Sol and GLM-5.3-Flash, to perform closed-loop embodied interactions across four simulated benchmarks and collect diverse trajectories. At each decision step, the expert receives visual observations and available state feedback, selects a tool and its parameters, and uses the execution feedback to inform subsequent decisions. The interaction records preserve task instructions, observations, decision text, tool calls, and environment feedback in temporal order. We use GPT-6.1-Sol as the Meta-Agent for rubric analysis and reminder-policy refinement in Section 5.1. We use GPT-5.6-Sol as the judge model for reminder-policy evaluation.

Following the tool-based execution paradigm of Harness VLA [60], the high-level planner is decoupled from low-level skill execution through a shared invocation-and-feedback interface. The execution layer exposes perception and motion primitives together with learned VLA/VLN skills, allowing the planner to invoke the capability appropriate to the current sub-goal while preserving the native action space of each skill. Depending on the benchmark and task, the available learned skills include $\pi_{0.5}$ [4], LingBot-VLA [48], RLDX-1 [20], and InternVLA-N1. The planner does not directly predict low-level actions; it selects a skill and specifies its invocation parameters, after which the resulting observations and execution status are returned as feedback for the next decision. This design enables the same task-level decision model to coordinate heterogeneous manipulation and navigation capabilities while retaining benchmark-specific execution modules.

We then filter and process these trajectories with our data pipeline (Section 4.1). Task completion is verified against environment feedback, and incomplete, invalid, or noisy records are filtered before constructing the training samples. The retained trajectories are normalized into multi-turn multimodal conversations, preserving the observation–decision–feedback sequence and aligning tool invocations with their outcomes. This yields 1,902, 427, 614, and 903 agentic training samples from LIBERO-Pro, RoboTwin 2.0, RoboCasa365, and R2R-CE, respectively. Supervision targets the planner’s decision text and tool invocations, with observations and environment feedback providing the conditioning context.

For embodied pretraining, we collect approximately 500K question-answering samples from ShareRobot [18] and RoboInter-VQA [26], covering spatial grounding, affordance understanding, and task planning. These samples provide visual-language supervision complementary to the multi-turn interaction data. OmniBrain-0 uses Qwen3.5-9B as its high-level multimodal planner, coordinating the perception and execution tools through the interfaces described in Section 3.2. Further details and training hyperparameters are provided in Appendix C.

6.2 Overall Performance

Unless otherwise specified, the quantitative analysis below refers to OmniBrain-0 (9B); the additional 4B and RSI rows reserve results for the smaller model and the model iterated through the RSI Data Flywheel, respectively.

6.2.1 LIBERO-Pro

Table 1 summarizes results on LIBERO-Pro across SPATIAL, OBJECT, GOAL, and LIBERO-10 under instruction-redirectation (T) and position-swap (S) perturbations, together with average task execution time. The results show that strong expert models achieve high success rates but incur substantial execution overhead. In contrast, OmniBrain-0 reaches 65.5% overall success with an average execution time of only 1.6 minutes, substantially improving over the zero-shot backbone while remaining far more efficient than large expert models. Its gains are consistent across multiple perturbation types, indicating improved robustness to task and spatial changes. It also outperforms larger zero-shot models, demonstrating that distilling agentic planning and recovery capabilities provides a favorable balance between task performance and execution efficiency.

Table 1 Success rates (%) on LIBERO-Pro across the SPATIAL, OBJECT, GOAL, and L10 suites under Task (T) and Swap (S) perturbations. Time denotes the average task completion time in minutes; “/” indicates unavailable runtime. † denotes zero-shot evaluation.

Method	Success Rate (%) ↑									Time ↓
	Spat-T	Spat-S	Obj-T	Obj-S	Goal-T	Goal-S	L10-T	L10-S	Overall	
Training-Based Skill Policies										
OpenVLA [22]	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	/
NORA [17]	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	/
π_0 [5]	0.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.3	/
MolmoAct [25]	0.0	0.0	0.0	6.0	0.0	0.0	6.0	0.0	1.5	/
X-VLA [63]	0.0	0.0	8.0	2.0	9.0	1.0	10.0	0.0	3.8	/
AtomVLA [41]	1.0	16.0	0.0	10.0	11.0	2.0	9.0	1.0	6.3	/
$\pi_{0.5}$ [4]	41.0	52.0	56.0	69.0	42.0	36.0	46.0	8.0	43.8	0.6
Zero-Shot Agents and Task-Level Baselines										
Qwen3.5-4B [†]	6.0	10.0	10.0	17.0	21.0	14.0	6.0	1.0	10.6	7.5
Qwen3.5-9B [†]	6.0	11.0	28.0	13.0	13.0	20.0	22.0	12.0	15.6	6.2
Qwen3.8-27B [†]	13.0	18.0	55.0	48.0	24.0	17.0	24.0	9.0	26.0	9.2
GLM-5.3-Flash [†]	68.0	67.0	76.0	80.0	69.0	<u>71.0</u>	<u>52.0</u>	<u>36.0</u>	64.9	8.5
GPT-5.6-Sol[†]	92.0	81.0	95.0	97.0	84.0	83.0	57.0	37.0	78.3	13.8
OmniBrain-0 (4B)	77.0	69.0	90.0	92.0	66.0	57.0	33.0	15.0	62.4	<u>1.5</u>
OmniBrain-0 (9B)	78.0	75.0	<u>93.0</u>	<u>96.0</u>	64.0	52.0	44.0	22.0	65.5	1.6
OmniBrain-0 (RSI)	<u>84.0</u>	<u>78.0</u>	92.0	95.0	<u>70.0</u>	61.0	48.0	26.0	<u>69.3</u>	1.6

6.2.2 LIBERO-Plus

Table 2 summarizes results on LIBERO-Plus under seven perturbation dimensions, namely camera viewpoints, robot initial states, language instructions, light conditions, background textures, sensor noise, and object

layout. The benchmark comprises 10,030 task variants. Each reported zero-shot model result is based on 30 episodes per perturbation dimension, yielding 210 evaluated episodes per model.

Table 2 Success rates (%) on LIBERO-Plus. Zero-shot model results are computed from 30 episodes per perturbation dimension (210 episodes per model). [†] denotes zero-shot evaluation.

Method	Success Rate (%) $\uparrow$							
	Camera	Robot	Language	Light	Back-ground	Noise	Layout	Total
<i>Training-Based Skill Policies</i>								
OpenVLA [22]	0.8	3.5	23.0	8.1	34.8	15.2	28.5	15.6
NORA [17]	2.2	37.0	65.1	45.7	58.6	12.8	62.1	39.0
π_0 [5]	13.8	6.0	58.8	85.0	81.4	79.0	68.9	53.6
OpenVLA-OFT [21]	56.4	31.9	79.5	88.7	93.3	75.8	74.2	69.6
ABot-M0 [52]	60.4	67.9	86.4	96.2	91.6	<u>86.4</u>	82.6	80.5
$\pi_{0.5}$ [4]	69.2	74.2	83.3	94.2	95.0	83.3	<u>85.0</u>	83.4
ACoT-VLA [64]	<u>72.6</u>	82.6	<u>87.5</u>	97.7	<u>96.5</u>	87.8	88.1	<u>86.6</u>
<i>Zero-Shot Agents and Task-Level Baselines</i>								
Qwen3.5-4B [†]	16.7	10.0	10.0	20.0	13.3	26.7	20.0	16.7
Qwen3.5-9B [†]	6.7	6.7	10.0	13.3	6.7	13.3	13.3	10.0
Qwen3.8-27B [†]	30.0	13.3	16.7	10.0	13.3	23.3	13.3	17.1
GLM-5.3-Flash [†]	86.7	66.7	90.0	83.3	86.7	80.0	73.3	77.1
GPT-5.6-Sol[†]	72.5	<u>79.2</u>	88.3	<u>95.0</u>	98.3	83.3	86.7	86.2
OmniBrain-0 (4B)	70.8	73.3	84.2	90.0	93.3	81.7	82.5	82.3
OmniBrain-0 (9B)	70.0	75.8	85.8	92.5	95.8	82.5	84.2	83.8
OmniBrain-0 (RSI)	70.8	76.7	85.0	94.2	96.7	83.3	<u>85.0</u>	84.5

6.2.3 RoboCasa365

Table 3 summarizes the evaluation results on the RoboCasa365 benchmark across Atomic-Seen, Composite-Seen, and Composite-Unseen scenarios.

Table 3 Success rates (%) on RoboCasa365 over the Atomic-Seen, Composite-Seen, and Composite-Unseen task splits. OmniBrain-0 (RSI) denotes the model obtained through iterative updates in the RSI Data Flywheel (Section 5.2). [†] denotes zero-shot evaluation.

Method	Success Rate (%) $\uparrow$			
	Atomic-Seen	Composite-Seen	Composite-Unseen	Overall
<i>Training-Based Skill Policies</i>				
Diffusion Policy [10]	15.7	0.2	1.3	6.1
π_0 [5]	34.6	6.1	1.1	14.8
$\pi_{0.5}$ [4]	39.6	7.1	1.2	16.9
Giga World-Policy 0.1 [53]	44.4	11.8	2.9	20.7
GR00T N1.5 [37]	50.7	14.8	2.7	23.9
RLDX-1 [20]	67.6	27.9	8.5	36.0
<i>Zero-Shot Agents</i>				
Qwen3.5-4B [†]	27.0	5.8	0.0	11.7
Qwen3.5-9B [†]	28.3	6.3	0.0	12.2
Qwen3.8-27B [†]	37.2	11.3	2.5	17.8
GLM-5.3-Flash [†]	74.4	28.8	13.8	40.4
GPT-5.6-Sol[†]	81.1	45.0	27.5	52.4
OmniBrain-0 (4B)	60.0	30.0	7.5	33.6
OmniBrain-0 (9B)	71.1	32.5	11.3	39.6
OmniBrain-0 (RSI)	<u>74.4</u>	<u>36.3</u>	<u>15.0</u>	<u>43.2</u>

GPT-5.6-Sol still demonstrates a high performance ceiling and exceptional task generalization, achieving top success rates across all task splits. Meanwhile, our OmniBrain-0 exhibits an impressive and well-balanced overall capability: it achieves an overall success rate of 39.6%, outperforming prior end-to-end VLA baselines such as RLDX-1, achieving $3\times$ the success rate of zero-shot Qwen3.5-9B, and approaching the performance of GLM-5.3-Flash.

6.2.4 RoboTwin 2.0

Table 4 summarizes the RoboTwin evaluation results across **Clean** and **Randomized** task configurations. Our OmniBrain-0 achieves a success rate of 88.8% under the Clean setup and maintains 85.6% under environmental randomization, reaching an overall score of 87.2%. This performance surpasses strong baselines such as LingBot-VLA and GLM-5.3-Flash, while maintaining a compact model size and high execution efficiency.

Table 4 Success rates (%) on RoboTwin 2.0. **Clean** denotes the standard setting with fixed scene configurations, while **Randomized** applies domain randomization. [†] denotes zero-shot evaluation.

Method	Success Rate (%) $\uparrow$		
	Clean	Randomized	Overall
<i>Training-Based Skill Policies</i>			
X-VLA [63]	72.8	41.2	57.0
$\pi_{0.5}$ [4]	81.2	75.6	78.4
LingBot-VLA [48]	86.0	83.6	84.8
ABot-M0 [52]	86.06	85.08	85.57
Qwen-VLA [44]	86.10	87.20	86.65
<i>Zero-Shot Agents</i>			
Qwen3.5-4B [†]	59.2	62.4	60.8
Qwen3.5-9B [†]	62.8	60.4	61.6
Qwen3.8-27B [†]	68.4	67.6	68.0
GLM-5.3-Flash [†]	87.6	86.4	87.0
GPT-5.6-Sol[†]	92.0	90.4	91.2
OmniBrain-0 (4B)	88.0	86.4	87.2
OmniBrain-0 (9B)	88.8	87.6	88.2
OmniBrain-0 (RSI)	<u>90.4</u>	<u>88.4</u>	<u>89.4</u>

6.2.5 R2R-CE

Table 5 summarizes navigation performance using Success Rate (SR), Success weighted by Path Length (SPL), Oracle Success Rate (OSR), and Navigation Error (NE). Our evaluations use the 100-episode **rand100** subset of R2R-CE Val-Unseen. Direct performance comparisons are restricted to methods evaluated on the same episode subset. Among zero-shot planners, GPT-5.6-Sol achieves the highest SR (71.0%) and OSR (75.0%) and the lowest NE (3.11 m), reflecting the performance ceiling of strong general-purpose models on this task. Compared with the zero-shot Qwen3.5-9B sharing the same backbone, OmniBrain-0 improves SR from 34.0% to 65.0% and SPL from 26.9% to 54.8%, while reducing NE from 6.57 m to 3.78 m. It also outperforms zero-shot Qwen3.8-27B, a larger model, and zero-shot GLM-5.3-Flash on SR, SPL, and NE. Compared with specialist VLN models, OmniBrain-0 achieves a higher SR than all of them, slightly exceeding the strongest one, DualVLN (65.0% vs. 64.3%), and attains a lower NE (3.78 m vs. 4.05 m), although its SPL remains lower (54.8% vs. 58.5%). These results indicate that distillation effectively transfers the navigation decision-making capability of strong models into a compact model.

Table 5 Results on R2R-CE Val-Unseen. Published specialist-policy results are shown for reference under their original evaluation protocols. Our agent evaluations use the 100-episode **rand100** subset. Success Rate (SR), Success weighted by Path Length (SPL), and Oracle Success Rate (OSR) are reported as percentages; Navigation Error (NE) is reported in meters. [†] denotes zero-shot evaluation.

Method	SR $\uparrow$	SPL $\uparrow$	OSR $\uparrow$	NE $\downarrow$
<i>Training-Based Skill Policies</i>				
NaVid [57]	37.4	35.9	49.1	5.47
Uni-NaVid [58]	47.0	42.7	53.3	5.58
AO-Planner [6]	47.0	33.0	59.0	5.55
ActiveVLN [61]	52.9	45.7	62.1	5.34
StreamVLN [47]	56.9	51.9	64.2	4.98
Progress-Think [45]	60.1	53.6	63.6	4.68
DualVLN [46]	64.3	58.5	70.7	4.05
<i>Zero-Shot Agents</i>				
Qwen3.5-4B [†]	18.0	13.2	41.0	7.74
Qwen3.5-9B [†]	34.0	26.9	48.0	6.57
Qwen3.8-27B [†]	41.0	31.6	58.0	6.12
GLM-5.3-Flash [†]	58.0	35.5	<u>73.0</u>	4.31
GPT-5.6-Sol[†]	71.0	<u>57.5</u>	75.0	3.11
OmniBrain-0 (4B)	55.0	45.9	58.0	5.01
OmniBrain-0 (9B)	65.0	54.8	71.0	3.78
OmniBrain-0 (RSI)	<u>66.0</u>	55.2	<u>73.0</u>	<u>3.70</u>

6.2.6 Spatial understanding and referring

Tables 6 and 7 evaluate spatial capabilities of the 4B and 9B models. EmbSpatial-Bench measures recognition of egocentric spatial relations through multiple-choice questions, while RefSpatial-Bench requires predicting points for object localization and placement from spatially constrained instructions. These evaluations complement closed-loop task success by examining the spatial understanding and grounding that support embodied decisions.

Table 6 Spatial question-answering accuracy (%) on EmbSpatial-Bench. MP3D, AI2-THOR, and ScanNet denote the three source environments; Overall reports accuracy across all evaluation questions. The benchmark covers six egocentric spatial relations: above, below, left, right, close, and far.

Method	Accuracy (%) $\uparrow$			
	MP3D	AI2-THOR	ScanNet	Overall
InternVL3-8B [69]	76.3	<u>74.2</u>	78.5	76.3
Cambrian-S-7B [51]	71.3	72.9	74.3	72.8
Qwen3.5-4B	77.7	71.8	77.1	75.5
Qwen3.5-9B	78.6	72.1	81.2	77.2
OmniBrain-0 (4B)	<u>81.1</u>	73.0	<u>83.2</u>	<u>79.0</u>
OmniBrain-0 (9B)	84.0	76.8	84.3	81.6

Table 7 Spatial referring accuracy (%) on RefSpatial-Bench. Location evaluates target-object localization, Placement evaluates free-space selection, and Unseen evaluates novel spatial-relation combinations.

Method	Accuracy (%) $\uparrow$			
	Location	Placement	Unseen	Overall
SpaceLLaVA [39]	5.8	4.3	4.0	4.8
RoboPoint [54]	22.9	9.3	8.4	13.9
Molmo-7B [11]	21.9	12.9	12.2	15.9
RoboRefer-2B-RFT [65]	<u>52.0</u>	54.0	41.6	49.8
Qwen3.5-4B	46.0	20.0	26.0	31.1
Qwen3.5-9B	65.0	41.0	36.4	<u>48.4</u>
OmniBrain-0 (4B)	43.0	46.0	<u>37.7</u>	42.6
OmniBrain-0 (9B)	<u>52.0</u>	<u>48.0</u>	41.6	47.7

6.2.7 Real-world manipulation

Figure 8 summarizes results on three real-world manipulation tasks: block stacking, apple placement, and drawer opening. OmniBrain-0 achieves a macro-average success rate of 73.3%, outperforming $\pi_{0.5}$, which achieves 58.3% across the three tasks. The improvement is primarily driven by block stacking, where success increases from 50% to 75%, followed by apple placement (40% to 60%), while performance on drawer opening is matched at 85%. GPT-5.6-Sol (zero-shot) achieves a higher macro-average of 86.7%, with the largest gaps to OmniBrain-0 on block stacking (90% vs. 75%) and apple placement (75% vs. 60%). These results support the applicability of the compact unified agent to the evaluated physical manipulation tasks, although further improvements are needed to match the teacher’s task-level performance.

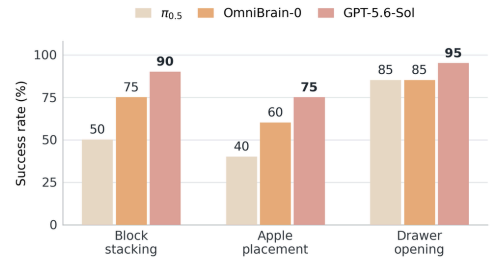


Figure 8 Real-world manipulation performance. Comparison of the standalone baseline $\pi_{0.5}$, the compact unified agent OmniBrain-0, and GPT-5.6-Sol (zero-shot). Success rates are reported as percentages.

6.3 Ablation Studies

6.3.1 Ablation Study on the Reminder Policy

Table 8 isolates the inference-time contribution of the reminder policy refined through the iterations in Section 5.1. With the model checkpoint fixed, we compare execution without reminders and with state-dependent reminders inserted before each high-level decision. The comparison uses the same task splits, seeds, tool interfaces, and execution budgets for both conditions, measuring whether the iteratively refined guidance improves task success independently of model-parameter updates.

Table 8 Inference-time ablation of the iteratively refined reminder policy with OmniBrain-0 (9B). The two conditions differ only in whether reminders precede each high-level decision. Scores are success rates (%): Overall for LIBERO-Pro, RoboCasa365, and RoboTwin 2.0; Total for LIBERO-Plus; and SR on Val-Unseen for R2R-CE.

Method	LIBERO-Pro Overall	LIBERO-Plus Total	RoboCasa365 Overall	RoboTwin 2.0 Overall	R2R-CE Val-Unseen SR
w/o reminder policy	65.5	83.8	39.6	88.2	65.0
w/ reminder policy	71.6	85.1	46.6	89.2	—

6.3.2 Ablation Study on the Data Processing Pipeline

The proposed data processing pipeline is designed to transform expert interaction data into a format suitable for training compact models. The ablation results on LIBERO-Pro in Table 9 demonstrate the impact of key components of the pipeline. When the final training data excludes expert failure-recovery trajectories and retains only a single successful trajectory, the model trained on this ablated dataset suffers an 9% drop in success rate. This degradation highlights the value of failure-recovery trajectories, which provide supervision for actively selecting corrective actions in response to concrete failure evidence, such as failed grasps or object drops during transportation. Furthermore, when the expert-generated reasoning is used directly as free-form text rather than being structurally rewritten, the model trained on the resulting data also exhibits a substantial performance drop. This is because expert models typically produce lengthy and unconstrained reasoning, and directly using such reasoning to construct long-horizon training trajectories for compact models can lead to overfitting to the linguistic style of the expert model, while weakening the learning of visual grounding and decision-making logic.

Table 9 Ablation study on the data pipeline with OmniBrain-0 (9B). Success rates (%) on LIBERO-Pro.

Method	Success Rate (%) ↑								
	Spat-T	Spat-S	Obj-T	Obj-S	Goal-T	Goal-S	L10-T	L10-S	Overall
w/o structured reasoning	45.0	42.0	76.0	70.0	28.0	23.0	11.0	8.0	37.9
w/o recovery data	<u>63.0</u>	<u>60.0</u>	<u>85.0</u>	<u>87.0</u>	<u>56.0</u>	<u>48.0</u>	<u>35.0</u>	<u>18.0</u>	<u>56.5</u>
OmniBrain-0 (full)	78.0	75.0	93.0	96.0	64.0	52.0	44.0	22.0	65.5

6.3.3 Ablation Study on Experience Library

Table 10 evaluates the effect of the Experience Library on agent execution, using OmniBrain-0 (9B) with and without access to the library. The library consistently improves the success rate across all three benchmarks, with gains of 27.7, 6.8, and 26.4 points on LIBERO-Pro, RoboTwin 2.0, and RoboCasa365, respectively. By providing reusable priors from previously collected cross-episode experiences, the Experience Library helps the agent avoid repeated exploration and make more informed execution decisions. Since only successful trajectories are retained for training, the Experience Library also increases the yield of canonical and recovery trajectories available for distillation into OmniBrain-0.

Table 10 Ablation study on the experience library with OmniBrain-0 (9B). Success rates (%) on LIBERO-Pro, RoboTwin 2.0, and RoboCasa365.

Method	Success Rate (%) ↑		
	LIBERO-Pro	RoboTwin 2.0	RoboCasa365
w/o experience library	<u>50.6</u>	<u>84.4</u>	<u>26.0</u>
w/ experience library	78.3	91.2	52.4

6.3.4 Ablation Study on Pretraining

Table 11 evaluates the contribution of Embodied Pretraining by comparing OmniBrain-0 with and without the pretraining stage. Removing pretraining consistently degrades performance across all three benchmarks, with drops of 0.6, 1.8, and 3.2 points on LIBERO-Pro, RoboTwin 2.0, and RoboCasa365, respectively. These results demonstrate that embodied pretraining provides transferable knowledge that benefits diverse manipulation tasks and improves generalization across different embodied environments. The larger gains on RoboTwin 2.0 and RoboCasa365 further suggest that pretraining is particularly beneficial for benchmarks involving more diverse task configurations and interaction patterns.

Table 11 Ablation study on embodied pretraining with OmniBrain-0 (9B). Success rates (%) on LIBERO-Pro, RoboTwin 2.0, and RoboCasa365.

Method	Success Rate (%) $\uparrow$		
	LIBERO-Pro	RoboTwin 2.0	RoboCasa365
w/o pretraining	<u>64.9</u>	<u>85.4</u>	<u>36.4</u>
OmniBrain-0 (full)	65.5	87.2	39.6

6.3.5 Ablation Study on VLA skill

As shown in Table 12, we further investigate the impact of using different low-level VLA models for `vla_skill` execution on OmniBrain-0 using RoboTwin 2.0. Across both **Clean** and **Randomized** settings, OmniBrain-0 consistently improves over the corresponding standalone VLA models, with gains of 4.0/2.4 over $\pi_{0.5}$ and 2.8/2.0 percentage points LingBot-VLA, respectively. This demonstrates that OmniBrain-0 can effectively exploit agentic planning and skill scheduling to improve the overall performance beyond that of the underlying VLA skill model. At the same time, the choice of low-level VLA remains important: replacing $\pi_{0.5}$ with the stronger LingBot-VLA increases the overall success rate from 81.6% to 87.2%, indicating that the capability of low-level skills directly influences the performance ceiling of the unified agentic system.

Table 12 Ablation study on the low-level VLA skill used by OmniBrain-0 (9B). Success rates (%) on RoboTwin 2.0. The full model uses LingBot-VLA as its VLA skill.

Method	Success Rate (%) $\uparrow$		
	Clean	Randomized	Overall
$\pi_{0.5}$ as <code>vla_skill</code>	<u>85.2</u>	<u>78.0</u>	<u>81.6</u>
LingBot-VLA as <code>vla_skill</code>	88.8	85.6	87.2

6.4 Qualitative Analysis

Figures 9, 10, 11, and 12 present representative executions on LIBERO-Pro, RoboTwin 2.0, RoboCasa365, and R2R-CE, respectively. Each sequence pairs selected observations with the agent’s reasoning and tool calls, showing how task progress and execution feedback inform subsequent decisions. The manipulation cases illustrate object grounding, skill composition, and recovery, while the navigation cases show landmark-based sub-goal execution. Figure 14 further shows when reminder guidance is triggered during manipulation. Complete manipulation traces and additional navigation comparisons are analyzed in Appendix D.

LIBERO-Pro. Figure 9 illustrates both failure recovery and goal-directed placement.

In the stove task, the initial grasp fails and the robot reaches the stove without the moka pot (stages 2–3). The agent uses this execution evidence to revise the action order: it activates the stove, refines the grasp cue, re-grasps the pot, and transports it onto the burner before verifying completion. This sequence shows a change in the next decision following a failed skill invocation. In the book-placement task, the agent localizes the book and the caddy, secures and lifts the book, aligns it with the back compartment, and releases it before checking the goal. Together, the examples show how perception, VLA skills, and explicit arm motions are coordinated through a shared task state.

RoboTwin 2.0. Figure 10 examines sequential object placement and articulated-object interaction.

In the colored-block task, the agent tracks which blocks have been grasped and placed as it arranges red, green, and blue blocks in a row. Intermediate decisions maintain grasp state during transport and align each new block with those already placed. The microwave task requires a different interaction pattern: the agent localizes the handle, establishes a grasp, and follows the door’s swing while maintaining contact and clearance. These cases illustrate how the high-level model selects and coordinates skills according to the current object state, using placement verification for one task and continuous handle interaction for the other.

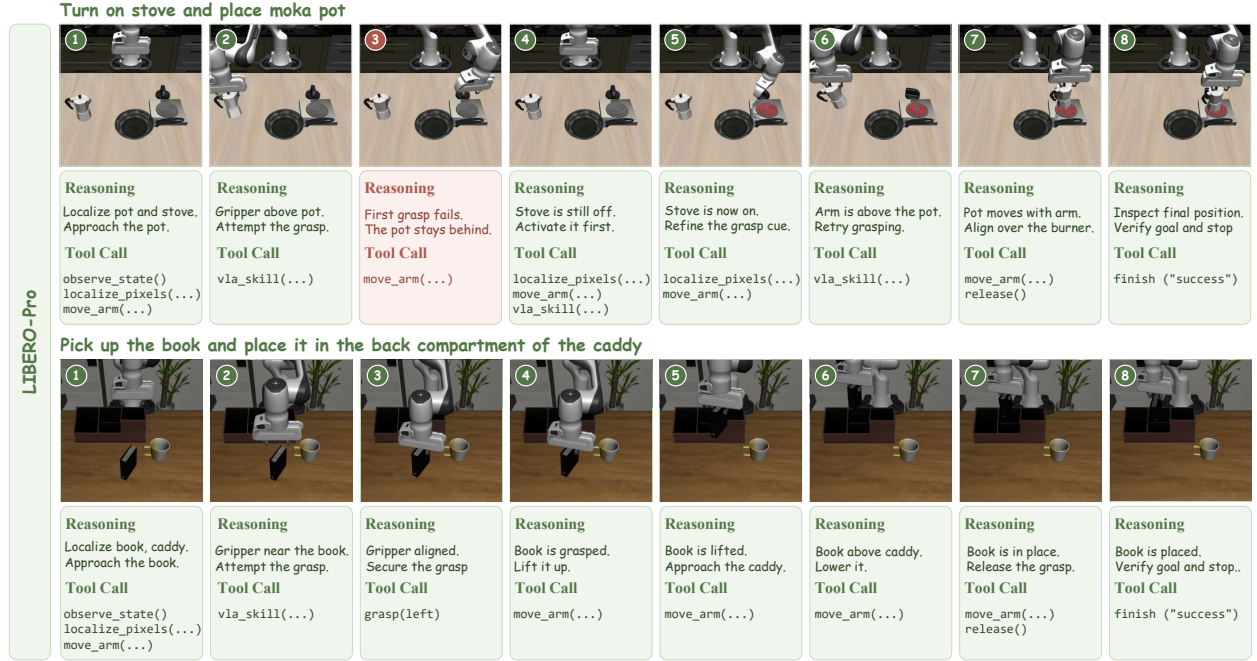


Figure 9 Examples of OmniBrain-0 on LIBERO-Pro. Top: turning on the stove and placing a moka pot on it, with red annotations highlighting the failed grasp. Bottom: placing a book in the back compartment of a caddy.



Figure 10 Examples of OmniBrain-0 on RoboTwin 2.0. Top: arranging colored blocks in a row. Bottom: opening a microwave.

RoboCasa365. Figure 11 shows household tasks with dependent sub-goals and changes in manipulation context.

In the vegetable task, an off-target pepper prompts renewed localization and a grasp retry before the agent places it in the pot, handles the next vegetable, and introduces the utensil. The correction is tied to the observed object state rather than an unchanged repetition of the previous action. In the bread-basket task, the agent first gains access through the cabinet, grasps and transfers the bread into the basket, and then

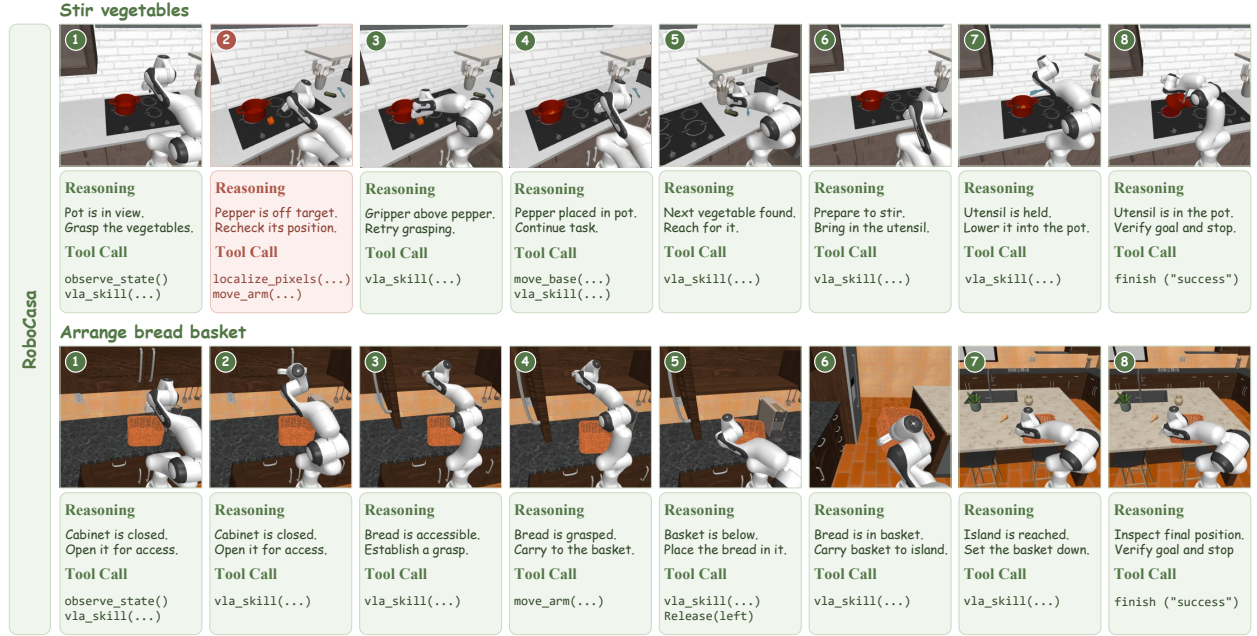


Figure 11 Examples of OmniBrain-0 on RoboCasa365. Top: stirring vegetables. Bottom: arranging a bread basket. Each sequence shows eight selected stages with observations, reasoning, and tool calls; red annotations highlight a failed attempt in the vegetable task.

moves the basket to the island. Tracking these intermediate outcomes allows it to progress from access and object handling to transport of the assembled result.

R2R-CE. Figure 12 illustrates instruction decomposition and landmark-grounded navigation.

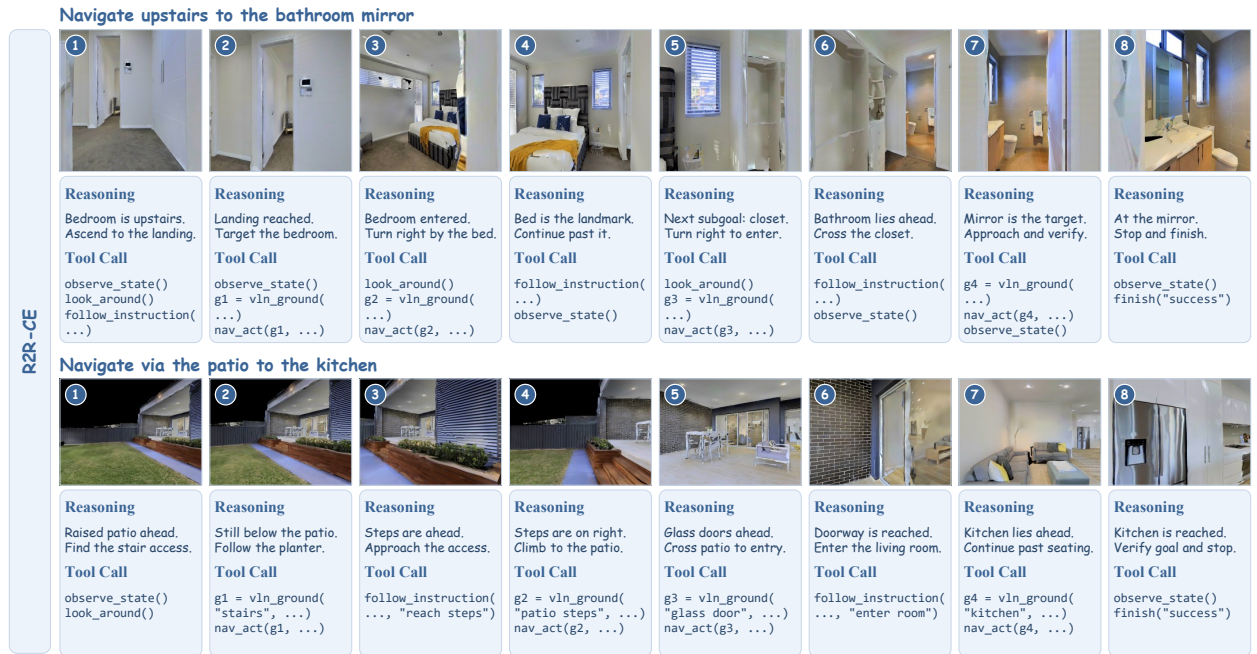


Figure 12 Examples of OmniBrain-0 on R2R-CE Val-Unseen. Top: navigating upstairs to a bathroom mirror. Bottom: navigating via a patio to a kitchen. Each sequence shows eight selected stages, pairing observations with subgoal reasoning and navigation tool calls.

For the bathroom-mirror route, the agent ascends to the landing and uses the bedroom, bed, and closet as successive landmarks before approaching the mirror and verifying the stopping condition. For the patio-to-kitchen route, it first identifies the stair access, follows the planter to the steps, reaches the raised patio, and enters through the glass doors before continuing through the living room to the kitchen. The selected stages show how observation and panoramic inspection support the choice of grounded navigation targets and short-horizon instructions. Progress is assessed after each sub-goal, so the final stop is conditioned on reaching the requested endpoint. A separate recovery case and comparisons with InternVLA-N1 are provided in Figures D.4 and D.5 in the appendix.

Real-world performance. To further evaluate the embodied planning and sim-to-real generalization capabilities of OmniBrain-0, we directly deploy the model trained in simulation environments to the real world without additional training.

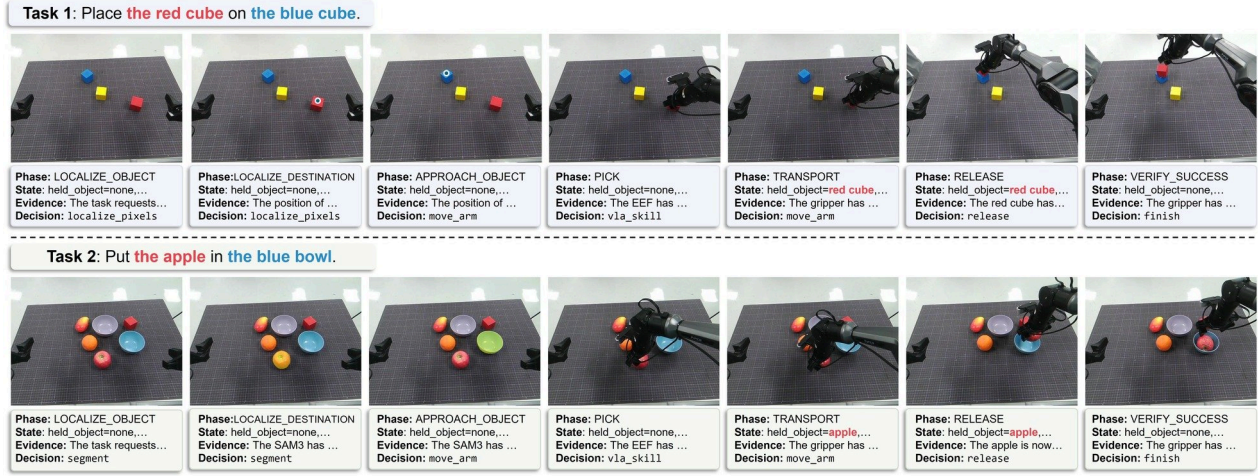


Figure 13 Examples of OmniBrain-0 on Real-World Tasks. We deploy OmniBrain-0 to a real-world robot for evaluation on real-world tasks. The figure illustrates the closed-loop observation–action–feedback execution process of OmniBrain-0 on two representative real-world tasks: *Place the red cube on the blue cube.* and *Put the apple in the blue bowl.*

Figure 13 presents two representative real-world manipulation tasks, where OmniBrain-0 performs closed-loop task execution by dynamically coordinating perception, motion, and VLA-based skills. Rather than following a fixed action sequence, the agent selects different perception strategies according to the task—for example, using spatial querying for the cube-placement task and segmentation for the apple-placement task—and continuously updates the execution state after each skill. The transition from *held_object=None* to *held_object=red cube* or *apple* further demonstrates that subsequent decisions are conditioned on intermediate execution outcomes, enabling adaptive skill composition and feedback-driven task completion.

Reminder-guided execution. Figure 14 illustrates the inference-time reminder mechanism introduced in Section 5.1.

The reminder policy is checked against the current interaction context before the agent selects its next tool invocation. In this example, the initial *move_arm* and *vla_skill* decisions proceed without reminders. Guidance is triggered at step 2 before *grasp* and again at step 3 before *move_arm*, concentrating the intervention around grasping and the transition to subsequent motion. The later transport motions, release, and final *finish* decision proceed without further reminders. This pattern illustrates how the policy supplies guidance at selected states within an ongoing task, while OmniBrain-0 continues to generate the tool calls. Aligning the observations, trigger states, and tool sequence makes the timing of reminder guidance explicit within closed-loop execution.

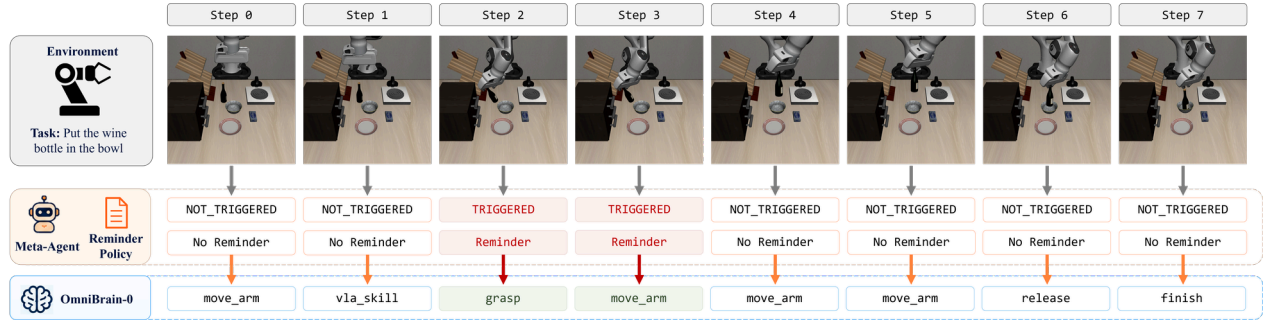


Figure 14 Reminder-guided manipulation. A wine-bottle placement example aligning visual observations (top), reminder-policy trigger states (middle), and OmniBrain-0’s tool invocations (bottom). For the task *Put the wine bottle in the bowl*, reminders are triggered at steps 2 and 3 before **grasp** and **move_arm**, respectively; the remaining displayed steps proceed without reminders. The sequence continues through bottle transport, release, and task termination.

7 Conclusion and Limitations

We presented **OmniBrain-0**, a lightweight unified embodied agent that coordinates manipulation and navigation through a shared high-level decision model. By combining embodied pretraining, domain-specialized agent distillation, and MOPD-based expert consolidation, OmniBrain-0 transfers interaction experience from powerful agents into a single compact planner while preserving the specialization of low-level skills. Experiments across manipulation and navigation benchmarks demonstrate consistent improvements over the compact zero-shot backbone and strong cross-domain task performance. These results suggest a practical path toward learning unified agentic capabilities for long-horizon embodied decision-making. Future work will further scale task diversity, improve cross-domain generalization, and extend evaluation to broader real-world settings.

Contributions and Acknowledgments

Core Contributors

- Name^{*,†}
- Name^{*}
- Name [✉]

Contributors

- Name
- Name

^{*}*Equal core contribution.* [†]*Project Leads.* [✉]*Corresponding author.*

Acknowledgments

We thank the teams and maintainers of the models, datasets, benchmarks, simulation environments, and robotic platforms used in this work. These resources were used for research and evaluation purposes.

References

- [1] Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, volume 2024, pages 21246–21263, 2024.
- [2] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.
- [3] Renat Aksitov, Sobhan Miryoosefi, Zonglin Li, Daliang Li, Sheila Babayan, Kavya Kopparapu, Zachary Fisher, Ruiqi Guo, Sushant Prakash, Pranesh Srinivasan, Manzil Zaheer, Felix Yu, and Sanjiv Kumar. ReST meets ReAct: Self-improvement for multi-step reasoning LLM agent. *arXiv preprint arXiv:2312.10003*, 2023. URL <https://arxiv.org/abs/2312.10003>.
- [4] Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, brian ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In Joseph Lim, Shuran Song, and Hae-Won Park, editors, *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 17–40. PMLR, 27–30 Sep 2025. URL <https://proceedings.mlr.press/v305/black25a.html>.
- [5] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0 : A Vision-Language-Action Flow Model for General Robot Control. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.010.
- [6] Jiaqi Chen, Bingqian Lin, Xinmin Liu, Lin Ma, Xiaodan Liang, and Kwan-Yee K. Wong. Affordances-oriented planning using foundation models for continuous vision-language navigation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 23568–23576, 2025. doi: 10.1609/aaai.v39i22.34526.
- [7] Qimao Chen, Fang Li, Yuechen Luo, Zehan Zhang, Haiyang Sun, Fangzhen Li, Bing Wang, Guang Chen, Yang Ji, Jiong Deng, Hongwei Xie, Hangjun Ye, Long Chen, and Yi Zhang. DriveReward: A comprehensive dataset and generative vision-language reward model for autonomous driving. *arXiv preprint arXiv:2606.08525*, 2026. URL <https://arxiv.org/abs/2606.08525>.
- [8] Tianxing Chen, Zanzin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025.
- [9] William Chen, Suneel Belkhale, Suvir Mirchandani, Karl Pertsch, Danny Driess, Oier Mees, and Sergey Levine. Training strategies for efficient embodied reasoning. In Joseph Lim, Shuran Song, and Hae-Won Park, editors, *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 365–391. PMLR, 27–30 Sep 2025. URL <https://proceedings.mlr.press/v305/chen25a.html>.
- [10] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 2024.
- [11] 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 vision-language models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 91–104. IEEE, 2025.
- [12] Mengfei Du, Binhao Wu, Zejun Li, Xuanjing Huang, and Zhongyu Wei. EmbSpatial-Bench: Benchmarking spatial understanding for embodied tasks with large vision-language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 346–355, 2024. doi: 10.18653/v1/2024.acl-short.33. URL <https://aclanthology.org/2024.acl-short.33/>.

- [13] Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. LIBERO-Plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025. URL <https://arxiv.org/abs/2510.13626>.
- [14] Liane Galanti, Dhruv Shah, and Tri Dao. Addressing the orchestration gap in generalist robots via physical agency. *arXiv preprint arXiv:2607.21725*, 2026.
- [15] Jinbang Huang, Yuanzhao Hu, Zhiyuan Li, Ran Qi, Yixin Xiao, Zhanguang Zhang, Mark Coates, Tongtong Cao, and Yingxue Zhang. Roboharness: Memory-driven orchestration of heterogeneous robot policies for long-horizon planning. *arXiv preprint arXiv:2607.18060*, 2026.
- [16] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.
- [17] Chia-Yu Hung, Qi Sun, Pengfei Hong, Amir Zadeh, Chuan Li, U-Xuan Tan, Navonil Majumder, and Soujanya Poria. NORA: A small open-sourced generalist vision language action model for embodied tasks. *arXiv preprint arXiv:2504.19854*, 2025.
- [18] 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. *arXiv preprint arXiv:2502.21257*, 2025.
- [19] Minki Kang, Jongwon Jeong, Seanie Lee, Jaewoong Cho, and Sung Ju Hwang. Distilling llm agent into small models with retrieval and code tools, 2025. URL <https://arxiv.org/abs/2505.17612>.
- [20] Dongyoung Kim, Huiwon Jang, Myungkyu Koo, Suhyeok Jang, Taeyoung Kim, et al. Rldx-1 technical report. *arXiv preprint arXiv:2605.03269*, 2026.
- [21] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *Proceedings of Robotics: Science and Systems (RSS)*, 2025. doi: 10.15607/RSS.2025.XXI.017.
- [22] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard, editors, *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 2679–2713. PMLR, 06–09 Nov 2025. URL <https://proceedings.mlr.press/v270/kim25c.html>.
- [23] Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *European Conference on Computer Vision*, pages 104–120. Springer, 2020.
- [24] Haoran Lang, Haotao Lu, Shiyu Sang, Haoyang Luo, Guo Chen, Qun Li, Jingyi Yu, Ye Shi, and Jingya Wang. EmbodiRSI: Recursive self-improvement for data-efficient robot adaptation. *arXiv preprint arXiv:2609.38905*, 2026. URL <https://arxiv.org/abs/2609.38905>.
- [25] Jason Lee, Jiafei Duan, Haoquan Fang, Yuquan Deng, Shuo Liu, Boyang Li, Bohan Fang, Jieyu Zhang, Yi Ru Wang, Sangho Lee, Winson Han, Wilbert Pumacay, Angelica Wu, Rose Hendrix, Karen Farley, Eli VanderBilt, Ali Farhadi, Dieter Fox, and Ranjay Krishna. MolmoAct: Action reasoning models that can reason in space. *arXiv preprint arXiv:2508.07917*, 2025.
- [26] Hao Li, Ziqin Wang, Zi-han Ding, Shuai Yang, Yilun Chen, Yang Tian, Xiaolin Hu, Tai Wang, Dahua Lin, Feng Zhao, Si Liu, and Jiangmiao Pang. Robointer: A holistic intermediate representation suite towards robotic manipulation. *arXiv preprint arXiv:2602.09973*, 2025.
- [27] Yongkang Li, Kaixin Xiong, Xiangyu Guo, Fang Li, Sixu Yan, Gangwei Xu, Lijun Zhou, Long Chen, Haiyang Sun, Bing Wang, et al. Recogdrive: A reinforced cognitive framework for end-to-end autonomous driving. *arXiv preprint arXiv:2506.08052*, 2025.
- [28] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International conference on robotics and automation (ICRA)*, pages 9493–9500. IEEE, 2023.

- [29] Jinghui Lu, Jiayi Guan, Zhijian Huang, Jinlong Li, Guang Li, Lingdong Kong, Yingyan Li, Han Wang, Shaoqing Xu, Yuechen Luo, et al. Xiaomi onevl: One-step latent reasoning and planning with vision-language explanation. *arXiv preprint arXiv:2604.18486*, 2026.
- [30] Yuechen Luo, Fang Li, Shaoqing Xu, Zhiyi Lai, Lei Yang, Qimao Chen, Ziang Luo, Zixun Xie, Shengyin Jiang, Jiaxin Liu, et al. Adathinkdrive: Adaptive thinking via reinforcement learning for autonomous driving. *arXiv preprint arXiv:2509.13769*, 2025.
- [31] Yuechen Luo, He Chong, Fang Li, Shaoqing Xu, Hexin Zhang, Ruiqi Zhang, Ziyang Song, Lei Yang, Zhi-Xin Yang, and Fuxi Wen. Reflect-VLA: Internalizing reflective guidance via dual-path on-policy self-distillation for autonomous driving. *arXiv preprint arXiv:2610.13453*, 2026. URL <https://arxiv.org/abs/2610.13453>.
- [32] Yuechen Luo, Fang Li, Qimao Chen, Shaoqing Xu, Jiaxin Liu, Ziyang Song, Zhi-xin Yang, and Fuxi Wen. Unleashing vla potentials in autonomous driving via explicit learning from failures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24833–24842, 2026.
- [33] Yuechen Luo, Fang Li, Shaoqing Xu, Yang Ji, Zehan Zhang, Bing Wang, Yuannan Shen, Jianwei Cui, Long Chen, Guang Chen, et al. Last-vla: Thinking in latent spatio-temporal space for vision-language-action in autonomous driving. *arXiv preprint arXiv:2603.01928*, 2026.
- [34] Ziang Luo, Kangan Qian, Jiahua Wang, Yuechen Luo, Jinyu Miao, Zheng Fu, Yunlong Wang, Sicong Jiang, Zilin Huang, Yifei Hu, et al. Mtrdrive: Memory-tool synergistic reasoning for robust autonomous driving in corner cases. *arXiv preprint arXiv:2509.20843*, 2025.
- [35] Wenhan Ma, Jianyu Wei, Liang Zhao, Hailin Zhang, Bangjun Xiao, Lei Li, Qibin Yang, Bofei Gao, Yudong Wang, Rang Li, et al. Mopd: Multi-teacher on-policy distillation for capability integration in llm post-training. *arXiv preprint arXiv:2606.30406*, 2026.
- [36] Soroush Nasiriany, Sep Nasiriany, Abhiram Maddukuri, and Yuke Zhu. Robocasa365: A large-scale simulation framework for training and benchmarking generalist robots. In *International Conference on Learning Representations*, volume 2026, pages 98643–98667, 2026.
- [37] NVIDIA, Johan Bjorck, Nikita Cherniadev, Fernando Castañeda, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. GR00T N1: An open foundation model for generalist humanoid robots. In *ArXiv Preprint*, March 2025.
- [38] Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL <https://qwen.ai/blog?id=qwen3.5>.
- [39] Remyx AI. SpaceLLaVA. Hugging Face model card, 2024. URL <https://huggingface.co/remyxai/SpaceLLaVA>.
- [40] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pages 8634–8652, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.
- [41] Xiaoquan Sun, Zetian Xu, Chen Cao, Zonghe Liu, Yihan Sun, Jingrui Pang, Ruijian Zhang, Zhen Yang, Kang Pang, Dingxin He, Mingqi Yuan, and Jiayu Chen. AtomVLA: Scalable post-training for robotic manipulation via predictive latent world models. *arXiv preprint arXiv:2603.08519*, 2026.
- [42] Xiaomi Embodied Intelligence Team, Shaoqing Xu, Fang Li, Guozhi Zhan, Zhixiang Duan, Yuhang Wang, Yuechen Luo, Shengyin Jiang, Hanbing Li, Zhiying Du, et al. One policy, many embodiments: Unified camera-centric action geometry pre-training for heterogeneous embodied manipulation. *arXiv preprint arXiv:2608.26058*, 2026.
- [43] Peidong Wang, Ying Chang, Zhiming Ma, Xiaocui Yang, Shi Feng, Daqi Liu, Yuzhe Yang, Xu Cao, Chuang Wang, Jian Su, and Daling Wang. AdaRSI: On-demand recursive self-improvement for embodied agents. *arXiv preprint arXiv:2610.20474*, 2026. URL <https://arxiv.org/abs/2610.20474>.
- [44] Qiuyue Wang, Mingsheng Li, Jian Guan, Jinhui Ye, Sicheng Xie, Yitao Liu, Junhao Chen, Zhixuan Liang, Jie Zhang, Xintong Hu, Xuhong Huang, Pei Lin, Junyang Lin, Dayiheng Liu, Shuai Bai, Jingren Zhou, Jiazhao Zhang, Haoqi Yuan, Gengze Zhou, Hang Yin, Ye Wang, Yiyang Huang, Zixing Lei, Wujian Peng, Delin Chen, Yingming Zheng, Jingyang Fan, Xianwei Zhuang, Xin Zhou, Haoyang Li, Anzhe Chen, Tong Zhang, Xuejing Liu, Yuchong

- Sun, Ruizhe Chen, Zhaohai Li, Chenxu Lü, Zhibo Yang, Tao Yu, and Xionghui Chen. Qwen-VLA: Unifying vision-language-action modeling across tasks, environments, and robot embodiments. *arXiv preprint arXiv:2605.30280*, 2026.
- [45] Shuo Wang, Yucheng Wang, Guoxin Lian, Yongcai Wang, Maiyue Chen, Kaihui Wang, Bo Zhang, Zhizhong Su, Yutian Zhou, Wanting Li, Deying Li, and Zhaoxin Fan. Progress-Think: Semantic progress reasoning for vision-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4076–4086, 2026.
- [46] Meng Wei, Chenyang Wan, Peng Peng, Xiqian Yu, Yuqiang Yang, Delin Feng, Wenzhe Cai, Chenming Zhu, Tai Wang, Jiangmiao Pang, et al. Ground slow, move fast: A dual-system foundation model for generalizable vision-language navigation. In *International Conference on Learning Representations*, volume 2026, pages 12380–12396, 2026.
- [47] Meng Wei, Chenyang Wan, Xiqian Yu, Tai Wang, Yuqiang Yang, Xiaohan Mao, Chenming Zhu, Wenzhe Cai, Hanqing Wang, Yilun Chen, Xihui Liu, and Jiangmiao Pang. Streamvln: Streaming vision-and-language navigation via slowfast context modeling, 2026. URL <https://arxiv.org/abs/2507.05240>.
- [48] Wei Wu, Fan Lu, Yunnan Wang, Shuai Yang, Shi Liu, Fangjing Wang, Qian Zhu, He Sun, Yong Wang, Shuailei Ma, et al. A pragmatic vla foundation model. *arXiv preprint arXiv:2601.18692*, 2026.
- [49] Wei Wu, Fangjing Wang, Fan Lu, He Sun, Shi Liu, Yunnan Wang, Yibin Yan, Yong Wang, Shuailei Ma, Xinyang Wang, et al. From foundation to application: Improving vla models in practice. *arXiv preprint arXiv:2607.06403*, 2026.
- [50] Shaoqing Xu, Fang Li, Yuechen Luo, Qimao Chen, Yifan Yang, Zhixiang Duan, Long Chen, and Zhi-xin Yang. World-In-Loop: Online correction via event-triggered world models for robust VLA policies. In *Computer Vision – ECCV 2026*, pages 664–681. Springer Nature Switzerland, 2026. doi: 10.1007/978-3-032-37083-9_37.
- [51] Shusheng Yang, Jihan Yang, Pinzhi Huang, Ellis Brown, Zihao Yang, Yue Yu, Shengbang Tong, Zihan Zheng, Yifan Xu, Muhan Wang, et al. Cambrian-s: Towards spatial supersensing in video. In *International Conference on Learning Representations*, volume 2026, pages 78185–78225, 2026.
- [52] Yandan Yang, Shuang Zeng, Tong Lin, Xinyuan Chang, Dekang Qi, Junjin Xiao, Haoyun Liu, Ronghan Chen, Yuzhi Chen, Dongjie Huo, Feng Xiong, Xing Wei, Zhiheng Ma, and Mu Xu. ABot-M0: VLA foundation model for robotic manipulation with action manifold learning. *arXiv preprint arXiv:2602.11236*, 2026.
- [53] Angen Ye, Boyuan Wang, Chaojun Ni, Guan Huang, Guosheng Zhao, Hao Li, Hengtao Li, Jie Li, Jindi Lv, Jingyu Liu, Min Cao, Peng Li, Qiuping Deng, Wenjun Mei, Xiaofeng Wang, Xinze Chen, Xinyu Zhou, Yang Wang, Yifan Chang, Yifan Li, Yukun Zhou, Yun Ye, Zhichao Liu, and Zheng Zhu. GigaWorld-Policy: An efficient action-centered world-action model. *arXiv preprint arXiv:2603.17240*, 2026.
- [54] 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. *arXiv preprint arXiv:2406.10721*, 2024.
- [55] Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. Agenttuning: Enabling generalized agent abilities for llms. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3053–3077, 2024.
- [56] Jiawei Zhang, Xiangrong Zhang, Rui Song, Huanbin Zhou, Chengye Song, and Hongzhou Wang. RE-0: Verified recursive improvement of embodied code-as-policy agents through local on-policy distillation. *arXiv preprint arXiv:2609.32416*, 2026. URL <https://arxiv.org/abs/2609.32416>.
- [57] Jiazhaoh Zhang, Kunyu Wang, Rongtao Xu, Gengze Zhou, Yicong Hong, Xiaomeng Fang, Qi Wu, Zhizheng Zhang, and He Wang. NaVid: Video-based VLM Plans the Next Step for Vision-and-Language Navigation. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi: 10.15607/RSS.2024.XX.079.
- [58] Jiazhaoh Zhang, Kunyu Wang, Shaoan Wang, Minghan Li, Haoran Liu, Songlin Wei, Zhongyuan Wang, Zhizheng Zhang, and He Wang. Uni-NaVid: A Video-based Vision-Language-Action Model for Unifying Embodied Navigation Tasks. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.013.
- [59] Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Y Zou, and Kunle Olukotun. Agentic context

- engineering: Evolving contexts for self-improving language models. In *International Conference on Learning Representations*, volume 2026, pages 86069–86100, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/8a94ff6f922d995d7d3f4ebf4143e442-Abstract-Conference.html.
- [60] Yixian Zhang, Huanming Zhang, Feng Gao, Xiao Li, Zhihao Liu, Chunyang Zhu, Jiaxing Qiu, Yuchen Yan, Jiyuan Liu, Wenhao Tang, et al. Harness vla: Steering frozen vlans into reliable manipulation primitives via memory-guided agents. *arXiv preprint arXiv:2607.08448*, 2026.
 - [61] Zekai Zhang, Weiye Zhu, Hewei Pan, Xiangchen Wang, Rongtao Xu, Xing Sun, and Feng Zheng. ActiveVLN: Towards active exploration via multi-turn RL in vision-and-language navigation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2026. doi: 10.1109/ICRA57385.2026.11696699.
 - [62] Zhonghan Zhao, Ke Ma, Wenhao Chai, Xuan Wang, Kewei Chen, Dongxu Guo, Yanting Zhang, Hongwei Wang, and Gaoang Wang. Do we really need a complex agent system? distill embodied agent into a single model. *arXiv preprint arXiv:2404.04619*, 2024.
 - [63] Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, et al. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. In *International Conference on Learning Representations*, volume 2026, pages 60580–60606, 2026.
 - [64] Linqing Zhong, Yi Liu, Yifei Wei, Ziyu Xiong, Maoqing Yao, Si Liu, and Guanghui Ren. ACoT-VLA: Action chain-of-thought for vision-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.
 - [65] 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. *Advances in Neural Information Processing Systems*, 38:28404–28481, 2026.
 - [66] Gengze Zhou, Yicong Hong, Zun Wang, Xin Eric Wang, and Qi Wu. Navgpt-2: Unleashing navigational reasoning capability for large vision-language models. In *European Conference on Computer Vision*, pages 260–278. Springer, 2024.
 - [67] Xueyang Zhou, Yangming Xu, Guiyao Tie, Yongchao Chen, Guowen Zhang, Duanfeng Chu, Pan Zhou, and Lichao Sun. Libero-pro: Towards robust and fair evaluation of vision-language-action models beyond memorization. *arXiv preprint arXiv:2510.03827*, 2025.
 - [68] Yuhang Zhou, Kai Zheng, Haoling Li, Dengyun Peng, Can Xu, and Jingjing Chen. Turnopd: Making on-policy distillation turn-aware for efficient long-horizon agent training. *arXiv preprint arXiv:2607.05804*, 2026.
 - [69] 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.

Appendix

Overview. This appendix provides system details, evaluation protocols, training configurations, and qualitative execution examples to complement the main paper. **Section A** describes the tool interfaces and Experience Library used for manipulation and navigation. **Section B** details the benchmarks and evaluation protocols. **Section C** reports the data sources, training configurations, and hardware setup. **Section D** analyzes complete manipulation traces and supplementary navigation examples.

A Details of the Agentic System Design

This section provides additional details of the agentic system design, including the tools developed for the VLA and VLN tasks, as well as the construction and utilization of the Experience Library. The tools provide standardized interfaces for interacting with heterogeneous model-based and scripted skills, while the Experience Library organizes reusable knowledge and interaction experiences accumulated from expert models.

Tools. We design a unified set of tools for VLA and VLN tasks, serving as the interface through which the agent interacts with and acts upon the physical environment. The detailed definitions of the tools are as follows.

General Tools
<ul style="list-style-type: none">• observe_state(): Reads the environment snapshot at the current step, including the task description, robot state, execution status, action history, and camera observations.• localize_pixels(view, pixels): Back-projects image pixels to metric coordinates.• segment(prompt, camera): Segments a target object using a text prompt and estimates its metric position.• finish(status): Terminates the agent’s execution loop and records the final execution status.• navigate_base(target_xy, tolerance, budget): Moves the mobile base toward a specified metric target with a scripted controller.• move_base(forward, lateral, turn): Executes short-range local base motions, including forward/backward movement, lateral displacement, and in-place rotation.

Figure A.1 General Tools: Shared tools for environment observation, pixel-to-metric localization, text-guided object segmentation, execution termination, and mobile-base control.

VLA-Specific Tools
<ul style="list-style-type: none">• move_arm(target, arm, gripper, budget): Moves the robot end-effector to a specified world-frame target, optionally controlling its gripper state.• orient_wrist(angle, arm, gripper, budget): Adjusts the end-effector orientation while maintaining its position when supported by the underlying controller.• grasp(arm): Closes the selected gripper.• release(arm): Opens the selected gripper.• vla_skill(prompt, budget): Invokes a learned VLA policy to execute a multi-step manipulation or interaction skill.

Figure A.2 VLA-Specific Tools: Tools for end-effector motion, wrist orientation, gripper control, and multi-step manipulation or interaction through a learned VLA policy.

VLN-Specific Tools

- **look_around():** Rotates the robot 360° in place and returns a labeled panoramic view.
- **follow_instruction(max_latents, s2_instruction, subgoal_id, semantic_subgoal, final_clause):** Invokes the VLN policy to follow a given navigation sub-instruction.
- **vln_ground(s2_instruction, subgoal_id):** Queries the VLN policy once for the next navigation goal and returns it with a grounding_id.
- **nav_act(grounding_id, max_policy_steps):** Executes a previously obtained navigation goal with the VLN policy.
- **follow_to_point(target):** Invokes the VLN policy to move toward a measured target point.
- **backtrack(target_step):** Returns the robot to its pose at a previous step along the recorded trajectory.

Figure A.3 VLN-Specific Tools: Tools for panoramic observation, instruction following, navigation-goal grounding and execution, point-directed navigation, and trajectory backtracking.

Experience Library. To collect diverse and successful expert interaction trajectories, we follow the data collection paradigm of Harness VLA [60], allowing expert models to explore the same task environment across multiple episodes and autonomously reflect on the causes of failures and the factors contributing to successful executions. These cross-episode experiences are then manually reviewed and consolidated into an Experience Library, which provides reusable priors for subsequent model execution. The library is organized into four complementary categories, covering general principles, task execution patterns, failure recovery, and the capability boundaries of VLA/VLN skills.

1. **General Principles.** Reusable perception, coordinate, gripper, controller, navigation, and safety knowledge that applies across tasks. Examples include image-first decision making, object-to-end-effector offsets, appropriate camera/view selection, spatial landmark awareness, and short, capped contact or navigation motions.
2. **Task Execution Patterns.** Reusable recipes for pick-transport-place, containment, articulated fixtures, stacking, handover, ranking, multi-object ordering, and navigation. These experiences describe task phases, skill composition, and verification gates rather than fixed scene coordinates, and cover both VLA-based manipulation and VLN-based navigation behaviors.
3. **Failure Cases and Recovery Strategies.** Negative evidence and recovery strategies accumulated from unsuccessful executions, including common failure modes, their observable symptoms, recovery actions, and appropriate execution budgets. For VLA tasks, these may involve grasp failures, collisions, or inaccurate object localization; for VLN tasks, they include navigation deviations, incorrect waypoint selection, ambiguous landmark grounding, and ineffective recovery behaviors.
4. **VLA/VLN Capability Boundaries.** Knowledge about the respective capabilities and limitations of VLA and VLN skills, including the conditions under which to invoke learned VLA/VLN policies, switch between them, or fall back to scripted tools. These experiences capture when learned skills are more reliable and when explicit geometric or rule-based actions are preferable.

B Benchmarks and Evaluation Protocol

LIBERO-Pro. LIBERO-Pro [67] is an extension of the LIBERO benchmark designed to evaluate the generalization capabilities of vision-language-action (VLA) models. It introduces controlled perturbations to the original LIBERO tasks, including changes to manipulated objects, object positions, task instructions, task configurations, and environments.

Following Harness VLA [60], we evaluate eight LIBERO-Pro cells: SPATIAL-T, SPATIAL-S, OBJECT-T, OBJECT-S, GOAL-T, GOAL-S, L10-T, and L10-S, where T denotes task or instruction redirection, and S denotes spatial swapping of object configurations. Each cell contains 10 tasks, with evaluation performed on seeds s_1 – s_{10} following the LIBERO-Pro protocol. This yields 100 rollouts per cell and 800 rollouts across the eight cells in total.

LIBERO-Plus. LIBERO-Plus [13] extends LIBERO with 10,030 task variants spanning camera viewpoints, robot initial states, language instructions, lighting, background textures, sensor noise, and object layout. These variations assess robustness to visual, linguistic, and physical changes in manipulation settings.

We report task success rates separately for each perturbation dimension and the overall success rate in Table 2. Each reported zero-shot model result is computed on a subset of 210 episodes, with 30 episodes per dimension, rather than all 10,030 task variants. Each episode is scored using the environment’s task-success condition; the overall score aggregates episode outcomes. Evaluation uses the released perturbation configurations.

RoboTwin 2.0. RoboTwin 2.0 [8] is a simulation benchmark for evaluating generalist policies in bimanual robotic manipulation, featuring 50 diverse manipulation tasks. It provides two evaluation settings: a **Clean** setting (Easy), which largely preserves the standard environment, and a **Randomized** setting (Hard), which introduces substantial environmental variations. This design enables evaluation of both task execution capability and robustness to distribution shifts.

Following Harness VLA [60], we evaluate all 50 tasks under both the **Clean** and **Randomized** settings, with 5 seeds evaluated for each setting, resulting in $2 \times 50 \times 5 = 500$ reported rollouts.

RoboCasa365. RoboCasa365 [36] is a large-scale simulation benchmark for evaluating generalist robot policies on everyday household tasks. The benchmark includes both atomic tasks targeting individual manipulation skills and composite tasks that require sequencing multiple skills for longer-horizon activities, enabling evaluation of general task-solving capabilities and generalization across diverse environments and task compositions.

We evaluate on the RoboCasa365 Target-50 split, comprising 18 ATOMIC-SEEN, 16 COMPOSITE-SEEN, and 16 COMPOSITE-UNSEEN tasks. The SEEN groups correspond to task templates present in pretraining, while UNSEEN contains templates held out from pretraining. We use held-out seeds s_1 – s_{10} for ATOMIC-SEEN and s_1 – s_5 for both composite groups, yielding $18 \times 10 = 180$, $16 \times 5 = 80$, and $16 \times 5 = 80$ reported rollouts, respectively.

R2R-CE. R2R-CE [23] is the continuous-environment adaptation of the Room-to-Room (R2R) vision-and-language navigation benchmark. It ports language-annotated navigation episodes to the Habitat simulator, where an agent follows a natural-language route instruction through reconstructed indoor environments. Unlike the original navigation-graph setting, the agent moves through continuous space rather than jumping between predefined viewpoints. We evaluate on **rand100**, a subset containing 100 episodes from the Val-Unseen split.

We report Navigation Error (NE), the final distance to the goal; Success Rate (SR), the percentage of episodes ending within 3m of the goal; Oracle Success Rate (OSR), the percentage whose trajectory reaches within 3m of the goal at any point; and Success weighted by Path Length (SPL), which accounts for both success and path efficiency.

EmbSpatial-Bench. EmbSpatial-Bench [12] contains 3,640 multiple-choice visual questions derived from MP3D, AI2-THOR, and ScanNet scenes. It tests six egocentric relations: above, below, left, right, close, and far. The three sources contribute 1,201, 1,239, and 1,200 questions, respectively.

Each input consists of an image, a spatial question, and candidate answers. Accuracy is the percentage of questions answered correctly. Table 6 reports accuracy for each source environment and across all questions; Overall therefore weights each environment by its question count.

RefSpatial-Bench. RefSpatial-Bench [65] evaluates spatial referring from an image and a spatially constrained instruction. Its Location and Placement sets contain 100 examples each, requiring target-object localization and free-space selection, respectively. Unseen comprises 77 examples drawn from these sets with spatial-relation combinations absent from the benchmark authors’ RefSpatial training data.

A prediction is correct when the output point falls within the annotated target mask. Table 7 reports the percentage of correct predictions for Location, Placement, and Unseen. Overall weights the three scores by their category sizes (100, 100, and 77); because Unseen overlaps the two task sets, it is a summary across evaluation categories rather than accuracy on a disjoint set of 277 examples.

Real-world manipulation. The real-world manipulation suite comprises three language-conditioned tasks: *block stacking*, where the robot places the red cube on the blue cube; *apple placement*, where the robot picks up the apple and places it in the blue bowl; and *drawer opening*, where the robot grasps the handle and pulls the drawer open. These tasks cover precise object placement, pick-and-place manipulation, and contact-rich articulated-object manipulation.

Each method is evaluated over 20 independent trials per task. A trial is successful if the robot completes the task within the allotted execution budget: placing the red cube stably on the blue cube, placing and releasing the apple inside the blue bowl, or opening the drawer to the specified target position. We report per-task Success Rate (SR), defined as the percentage of successful trials, and the macro-average SR across the three tasks.

C Datasets and Experimental Details

VQA Data. We construct our embodied pretraining corpus from ShareRobot [18] and RoboInter-VQA [26]. ShareRobot provides heterogeneous supervision for task planning, object affordance, and end-effector trajectories, spanning 23 source datasets, 12 embodiments, and 107 atomic task types, with over 1.0M planning-oriented VQA pairs. RoboInter-VQA complements this data with approximately 2.3M embodied VQA samples covering spatial and temporal reasoning, intermediate-representation understanding, and task planning. We further select data that focuses on spatial grounding, affordance understanding, and task planning, and incorporate agentic data at a predefined ratio, ultimately obtaining approximately 500k embodied pretraining samples.

Agentic Data. To obtain agentic data, we first use multiple expert models, including GPT-5.6-Sol and GLM-5.3-Flash, to perform closed-loop execution on manipulation and navigation tasks from LIBERO/LIBERO-Pro, RoboTwin, RoboCasa365, and R2R-CE, collecting raw agentic interaction trajectories. We then apply our designed data pipeline to filter and process these raw interaction logs, ultimately obtaining agentic data for supervised fine-tuning. From the interaction executions of LIBERO/LIBERO-Pro, RoboTwin, RoboCasa365, and R2R-CE, we obtain 1,902, 427, 614, and 903 training samples, respectively. Each sample consists of task instructions, structured decision reasoning, visual observations, robot states, tool calls, and environment feedback.

Hardware Configuration and Training Parameters. We train Qwen3.5-9B on 16 NVIDIA H20 GPUs using bfloat16 precision. Embodied Pretraining runs for 2 epochs with a batch size of 1, a learning rate of 5×10^{-6} , 4 gradient accumulation steps, and utilizes the Adam optimizer with a weight decay of 0.05 and a warmup ratio of 0.1. We then perform Domain-Specialized Distillation for 3 epochs using a batch size of 2, a learning rate of 1×10^{-5} , 2 gradient accumulation steps, and the same Adam optimizer setup (weight decay 0.05, warmup ratio 0.1). Finally, MOPD is applied for 2 epochs with a batch size of 2, a learning rate of 2×10^{-6} , and 2 gradient accumulation steps. Multiple frozen teacher models are independently deployed on dedicated GPUs. MOPD-specific configurations include a rollout temperature of 1.0, a distillation support top- k of 64, a Reverse KL distillation objective, and $\rho_{VLA} = \rho_{VLN} = 0.5$.

D More Qualitative Examples

This section complements the benchmark-level visualizations in Section 6.4 with complete manipulation execution traces and detailed navigation comparisons. Figures D.1–D.3 show the observation–reasoning–tool interaction sequences for three manipulation tasks. Figure D.4 presents selected states from a navigation recovery case, and Figure D.5 provides additional comparisons with InternVLA-N1. The navigation keyframes illustrate route decisions and outcomes rather than every intermediate action.

Task execution process of OmniBrain-0. Figures D.1, D.2, and D.3 present complete task execution processes across LIBERO-Pro, RoboTwin 2.0, and RoboCasa365; Figure D.4 extends the analysis to R2R-CE navigation with recovery. At each manipulation turn, the agent produces structured reasoning with four fields, *Phase*, *State*, *Evidence*, and *Decision*, and then issues a tool call. These fields connect the current task stage and interaction state to the evidence supporting the next action. The traces show object and destination grounding, specialist-skill invocation, intermediate state tracking, and verified termination. The navigation comparison likewise exposes the link between observations, route progress, recovery decisions, and the final stop.

LIBERO-Pro: grounded pick-and-place. In Figure D.1, the agent localizes the salad dressing and the basket separately, then approaches the target with the gripper open. A VLA skill establishes the grasp, after which explicit arm motion transports the held bottle above the basket. The state changes from holding the dressing to an open, empty gripper after release. The final decision uses the placement outcome and environment success signal to terminate, illustrating how localization, grasping, transport, and verification form a closed-loop sequence.

RoboTwin 2.0: tracking two held objects. Figure D.2 requires placing two specified bread loaves in the beige breadbasket. The agent tracks the left and right grippers separately. After the initial VLA execution, both grippers hold bread; releasing the left loaf completes only part of the task. The next decision retains the right-hand grasp, invokes the manipulation skill to continue placement, and then releases the right loaf. Termination follows visual confirmation that both loaves are in the basket and both grippers are empty.

RoboCasa365: preserving multi-stage task progress. In Figure D.3, the agent must place a croissant on the cutting board and then retrieve a jam jar from the cabinet to place beside it. It first grounds the objects and adjusts the mobile base for manipulation. Subsequent decisions distinguish the completed croissant placement from the remaining jam-jar sub-goal, and track the jar while it is held and transported. The final check considers both objects together on the board, illustrating progress tracking across navigation, manipulation, and successive placement requirements.

R2R-CE: landmark-guided navigation and recovery. Figure D.4 compares OmniBrain-0 with InternVLA-N1 on R2R-CE Val-Unseen. The instruction is *Start in the middle of the large room, head towards the door that leads to the outside, turn left and head up the stairs, take a hard left, stop when you are inside the workout room*. OmniBrain-0 grounds the outside door as the first landmark, identifies the left turn toward the stairs, and continues the ascent to the upper landing. It then follows the hard-left turn toward the workout room. When direct approaches are blocked by equipment or doorway geometry, the agent selects an alternative corridor around the obstruction instead of repeating the same forward move. After confirming that it is inside the workout room, it issues STOP at environment step 64. This example shows route progress and local execution evidence jointly guiding recovery and endpoint verification.

InternVLA-N1, in contrast, repeatedly searches for the initial landmark, remains on a partial route, and eventually encounters `specialist_no_actions` while still outside the target room. The comparison illustrates the distinction between advancing along part of an instruction and satisfying its ordered landmarks and final stopping condition. The displayed keyframes are selected independently for the two runs; their environment-step values do not imply synchronized execution and should not be read as paired progress measurements. Additional front-view and trajectory comparisons are provided in Figure D.5.

User Query:

Run this single-attempt perception-isolated LIBERO task. Task instruction: Pick the salad dressing and place it in the basket. Begin by inspecting environment step 0.

Tool_Call: `Observe_state()`



Step 0

Reasoning: Phase: LOCALIZE_OBJECT.

State: held_object=None; gripper=open; grasp=no; destination=basket; task_complete=no.

Evidence: The gripper fingers are visibly separated, with no clear grasp around the bottle; Robot state and camera observations were returned at step=0; The task requests moving salad dressing to basket.

Decision: next_tool=localize_pixels; Use localize_pixels at agentview pixel (row 455, col 430) for step 0 to ground the requested salad dressing target.

Tool_Call: `localize_pixels(agentview, [455,430])`

Reasoning: Phase: LOCALIZE_DESTINATION.

State: held_object=None; gripper=open; grasp=no; destination=basket; task_complete=no.

Evidence: The basket is visible and its interior appears empty; Robot state and camera observations were returned at step=0; The task requests moving cream cheese to basket.

Decision: next_tool=localize_pixels; Use localize_pixels at image coordinates row=500, col=575 for step=0 to ground the visible basket in world coordinates.

Tool_Call: `localize_pixels(agentview, [500,575])`

Reasoning: Phase: APPROACH_OBJECT.

State: held_object=None; gripper=open; grasp=no; destination=basket; task_complete=no.

Evidence: Localize_pixels grounded target at world_xyz=[-0.1805, -0.0914, 0.0574]; The gripper fingers are visibly separated, with no clear grasp around the bottle.

Decision: next_tool=move_arm; Move toward the grounded target with the gripper open.

Tool_Call: `move_arm([-0.18, -0.085, 0.16], left, -1, 80)`



Step 1

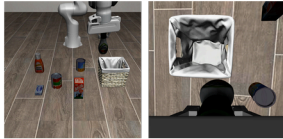
Reasoning: Phase: PICK.

State: held_object=None; gripper=open; grasp=no; destination=basket; task_complete=no.

Evidence: The salad dressing bottle is not in or on the basket; The basket is visible to the right and appears empty.

Decision: next_tool=vla_skill; Run pi0 with the prompt "pick up the salad dressing", allowing 10 chunks.

Tool_Call: `vla_skill("pick up the salad dressing", 10)`



Step 2

Reasoning: Phase: TRANSPORT.

State: held_object=salad dressing; gripper=closed; grasp=yes; destination=basket; task_complete=no.

Evidence: The dressing bottle is no longer at its earlier floor location; The basket interior is visibly empty.

Decision: next_tool=move_to; Transport the held salad dressing toward xyz [0.067, 0.249, 0.3] while keeping the gripper closed.

Tool_Call: `move_arm([0.067, 0.249, 0.3], left, 1, 80)`



Step 3

Reasoning: Phase: RELEASE.

State: held_object=salad dressing; gripper=closed; grasp=yes; destination=basket; task_complete=no.

Evidence: The gripper fingers are wrapped around the bottle neck; The original location of the salad dressing bottle is empty.

Decision: next_tool=release; Open the gripper with release because the salad dressing is held above its destination.

Tool_Call: `release(left)`



Step 4

Reasoning: Phase: VERIFY_SUCCESS.

State: held_object=None; gripper=open; grasp=no; destination=basket; task_complete=yes.

Evidence: The gripper fingers are visibly separated above the basket and are not holding the bottle; The release completed and opened the gripper.

Decision: next_tool=finish; Finish with success status and the supplied placement summary because release opened the gripper and the environment terminated successfully.

Tool_Call: `finish("success")`

Figure D.1 LIBERO-Pro task execution. Complete execution process for the task *Pick the salad dressing and place it in the basket.*

User Query:

RoboTwin task: Take the golden bread loaf and the bread with rounded corners then place in the beige breadbasket. Begin by inspecting environment step 0.

Tool_Call: `Observe_state()`



Step 0

Reasoning: Phase:PICK.

State: left_gripper=open; right_gripper=open; left_held_object=None; right_held_object=None.

Evidence: The synchronized head and wrist views are current for step 0; Breads are visibly positioned on either side of the beige breadbasket.

Decision: next_tool=vla_skill.

Tool_Call: `vla_skill("Take the golden bread loaf and the bread with rounded corners then place in the beige breadbasket", 2)`



Step 1

Reasoning: Phase:RELEASE.

State: left_gripper=closed; right_gripper=closed; left_held_object=bread; right_held_object=bread.

Evidence: The synchronized head and wrist views are current for step 1. One golden bread loaf is being held by the gripper above the basket.

Decision: next_tool=release.

Tool_Call: `release(left)`



Step 2

Reasoning: Phase:TRANSPORT.

State: left_gripper=open; right_gripper=closed; left_held_object=None; right_held_object=bread.

Evidence: The synchronized head and wrist views are current for step 2; One golden bread loaf is visibly inside the beige breadbasket.

Decision: next_tool=vla_skill.

Tool_Call: `vla_skill("Take the golden bread loaf and the bread with rounded corners then place in the beige breadbasket", 1)`



Step 3

Reasoning: Phase:RELEASE.

State: left_gripper=open; right_gripper=closed; left_held_object=None; right_held_object=bread.

Evidence: The synchronized head and wrist views are current for step 3; Bread loaves are visible in the breadbasket.

Decision: next_tool=release;

Tool_Call: `release(right)`



Step 4

Reasoning: Phase:VERIFY_SUCCESS.

State: left_gripper=open; right_gripper=open; left_held_object=None; right_held_object=None.

Evidence: The synchronized head and wrist views are current for step 4; Bread loaves are visible in the breadbasket.

Decision: next_tool=finish.

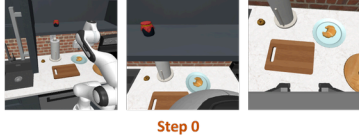
Tool_Call: `finish("success")`

Figure D.2 RoboTwin 2.0 task execution. Complete execution process for placing the two specified bread loaves in the beige breadbasket.

User Query:

RoboCasa task: From the different types of pastries on the counter, select a croissant and place it on the cutting board. Then retrieve a jar of jam from the cabinet and place it alongside the croissant on the cutting board. Begin by inspecting environment step 0.

Tool_Call: `Observe_state()`



Step 0

Reasoning: Phase: LOCALIZE.

State: gripper=open; held_object=None.

Evidence: The synchronized agentview, navview, and wrist views are current for step 0; A croissant is on the blue plate beside the empty cutting board, and a jam jar is visible in the open upper cabinet; neither requested placement has been completed.

Decision: next_tool=localize_pixels.

Tool_Call: `localize_pixels(agentview, [[138,166],[139,169],[141,170],[136,171],[142,167]])`

Reasoning: Phase: LOCALIZE.

State: gripper=open; held_object=None.

Evidence: The synchronized agentview, navview, and wrist views are current for step 0; A croissant is on the blue plate beside the empty cutting board, and a jam jar is visible in the open upper cabinet; neither requested placement has been completed.

Decision: next_tool=localize_pixels.

Tool_Call: `localize_pixels(agentview, [[54,114],[56,116],[58,116],[60,116],[57,118],[52,116]])`

Reasoning: Phase: NAVIGATE.

State: gripper=open; held_object=None.

Evidence: The synchronized agentview, navview, and wrist views are current for step 0; Croissant and jam jar are visible and have been localized; Move to a suitable position for manipulation.

Decision: next_tool=navigate_base;

Tool_Call: `navigate_base([2.335,-0.224], 0.35, 300)`



Step 1

Reasoning: Phase: NAVIGATE.

State: gripper=open; held_object=None.

Evidence: The synchronized agentview, navview, and wrist views are current for step 1; The croissant and cutting board are visible, but the croissant has not been placed on the cutting board; Fine-tune the position to prepare for grasping.

Decision: next_tool=move_base.

Tool_Call: `move_base([0,-0.5], 0.35, 0)`



Step 2

Reasoning: Phase: MANIPULATE.

State: gripper=open; held_object=None.

Evidence: The synchronized agentview, navview, and wrist views are current for step 2; The croissant and cutting board are visible, but the croissant has not been placed on the cutting board.

Decision: next_tool=rldx_skill.

Tool_Call: `vla_skill("From the different types of pastries on the counter, select a croissant and place it on the cutting board. Then retrieve a jar of jam from the cabinet and place it alongside the croissant on the cutting board.", 40)`



Step 3

Reasoning: Phase: MANIPULATE.

State: gripper=open; held_object=None.

Evidence: The synchronized agentview, navview, and wrist views are current for step 3; The croissant is on the cutting board. The jam jar is visible in the open upper cabinet, not yet alongside the croissant.

Decision: next_tool=vla_skill.

Tool_Call: `vla_skill("From the different types of pastries on the counter, select a croissant and place it on the cutting board. Then retrieve a jar of jam from the cabinet and place it alongside the croissant on the cutting board.", 40)`



Step 4

Reasoning: Phase: MANIPULATE.

State: gripper=open; held_object=jam jar.

Evidence: The synchronized agentview, navview, and wrist views are current for step 4; Croissant placement on the cutting board is visible; The jam jar is being retrieved and has not visibly been placed alongside it yet.

Decision: next_tool=vla_skill.

Tool_Call: `vla_skill("From the different types of pastries on the counter, select a croissant and place it on the cutting board. Then retrieve a jar of jam from the cabinet and place it alongside the croissant on the cutting board.", 40)`



Step 5

Reasoning: Phase: VERIFY_TASK.

State: gripper=open; held_object=None.

Evidence: The synchronized agentview, navview, and wrist views are current for step 5; The croissant and jam jar have been placed together on the cutting board.

Decision: next_tool=finish.

Tool_Call: `finish("success")`

Figure D.3 RoboCasa365 task execution. Complete execution process for placing a croissant and a jam jar together on the cutting board.

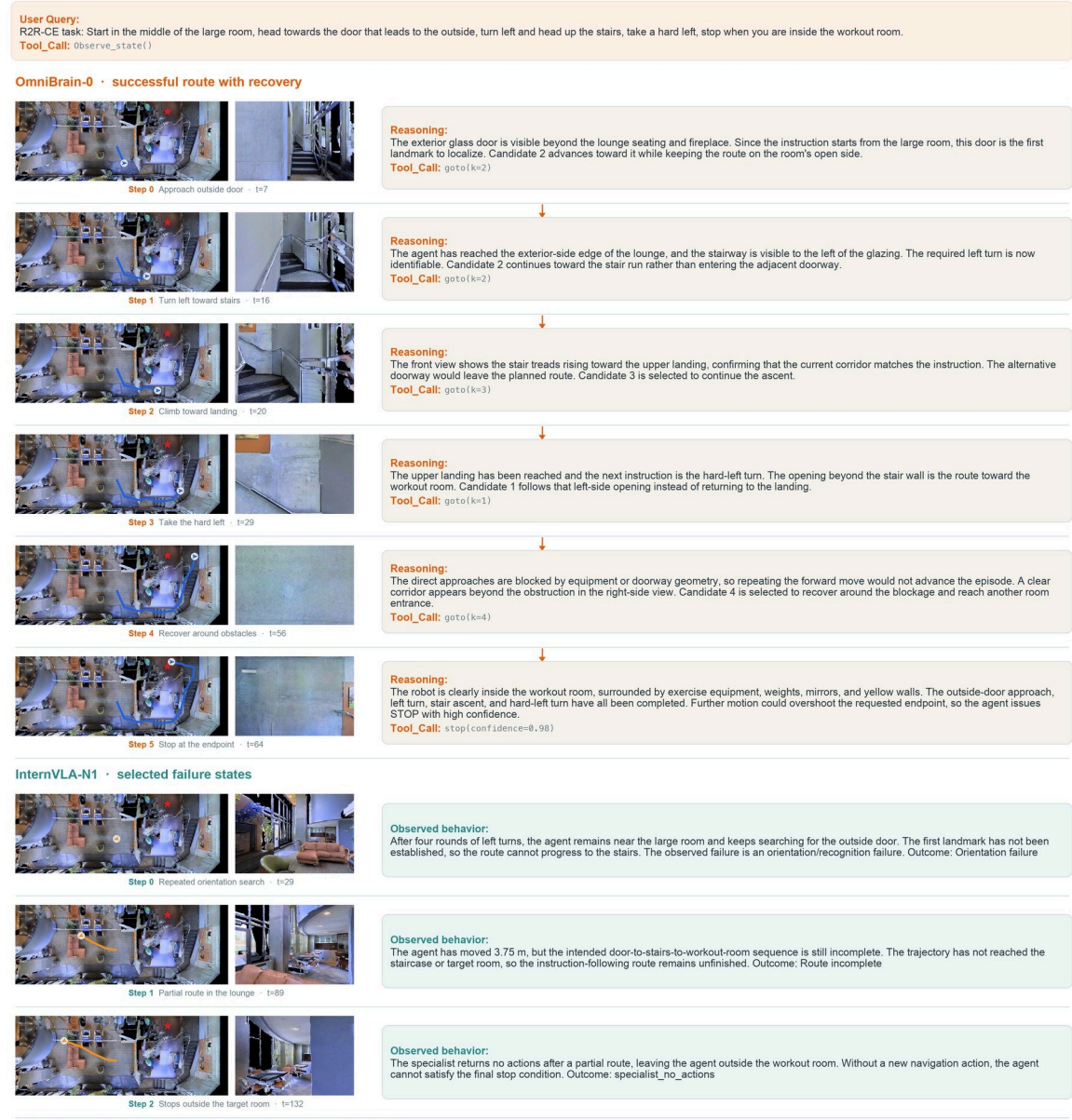


Figure D.4 R2R-CE navigation with recovery. Selected states from a representative navigation task in the Val-Unseen split. OmniBrain-0 follows the instructed landmarks, adapts its approach after blocked entries, and issues STOP at environment step 64, while InternVLA-N1 remains on a partial route and encounters **specialist_no_actions**. Keyframes are selected independently for the two runs; environment steps do not imply synchronized execution.

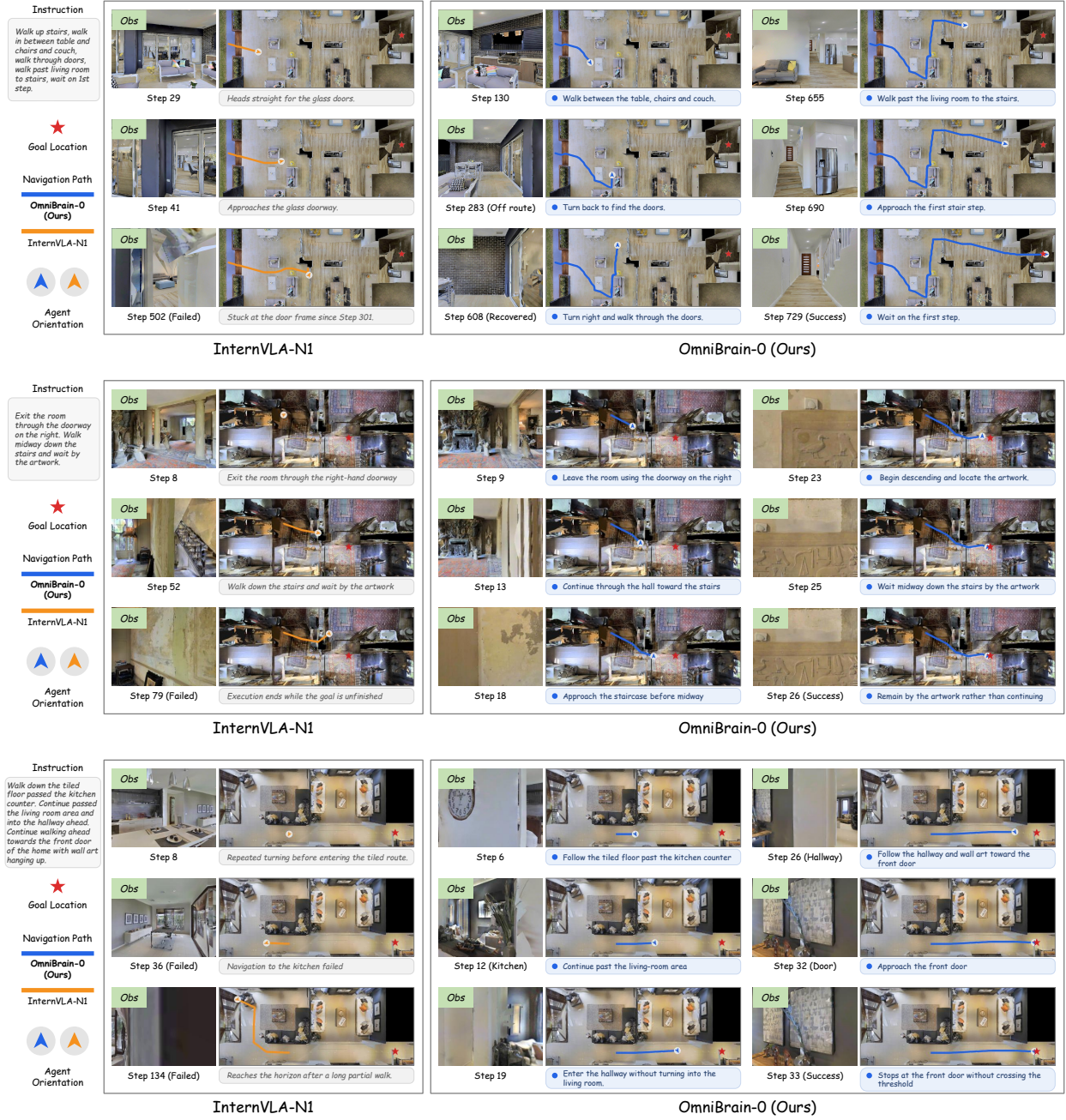


Figure D.5 Additional R2R-CE navigation comparisons. Each sequence compares InternVLA-N1 (left) and OmniBrain-0 (right) on the same Val-Unseen task. Cells pair front-view observations with top-down trajectories at the indicated environment steps. Blue annotations show OmniBrain-0's decisions, and gray annotations describe InternVLA-N1's behavior.