

# ACTIVEARENA: BENCHMARKING AND UNDERSTANDING ACTIVE PERCEPTION IN ROBOTIC MANIPULATION

Yibo Li<sup>1,2\*</sup>, Enshen Zhou<sup>1,2\*</sup>, Rui Chen<sup>1</sup>, Yanjun Ding<sup>1</sup>, Mengzhen Liu<sup>2,3</sup>  
Yi Han<sup>1,2</sup>, Jiabo Zhan<sup>4</sup>, Lipeng Wang<sup>1,2</sup>, Shanghang Zhang<sup>2,3</sup>, Lu Sheng<sup>1,2†</sup>

<sup>1</sup>Beihang University   <sup>2</sup>Beijing Academy of Artificial Intelligence

<sup>3</sup>Peking University   <sup>4</sup>Tsinghua University

\* Equal contribution.   † Corresponding author.

## ABSTRACT

Active perception and manipulation are crucial for robots to interact with complex scenes. Existing benchmarks struggle to evaluate how robots effectively acquire and maintain information in memory in an active manner. To this end, we introduce *ActiveArena-Sim*, an active-perception simulator with controllable viewpoints and large-scale workspaces as the foundation. Built on this, we propose *ActiveArena-Bench*, which comprises 35 tasks across 5 fine-grained categories, covering visual exploration and interactive information acquisition. Each task is difficult to solve from passive observations alone, requiring multi-round evidence acquisition and memory-based reasoning. The benchmark provides rich memory annotations, standardized training data, and ID/OOD protocols featuring disjoint scenes, unseen distractor configurations, and novel backgrounds. Moreover, we present *ActiveArena-VLA*, a modular suite of 13 vision-language-action configurations for controlled studies of memory writing, memory capacity, proprioceptive state, subtask supervision, and high-level planning in active perception. Benchmark results reveal a substantial ID-OOD gap: uniform memory sampling, increased memory capacity under reliable write policies, proprioceptive inputs, and subtask supervision improve OOD generalization, while planner-guided memory management and decision-making achieve performance close to the best-performing configuration using only sparse memory. *ActiveArena* thus provides a unified testbed to develop and diagnose models for active perception and manipulation.

Our code and data are open-source: <https://leeibo.github.io/ActiveArena>.

## 1 INTRODUCTION

Active perception is crucial for embodied AI, as robot agents should actively perceive unstructured scenes and act accordingly, like a human (Bajcsy, 1988). This requires two complementary abilities: (1) maintaining a memory of acquired and missing information, and (2) deciding what information to acquire next and how to obtain it via a perception-action loop (Yang et al., 2025; Wang et al., 2026; Xiong et al., 2025; Liu et al., 2026b;a; Hong et al., 2026; He et al., 2026; Li et al., 2026b). Specifically, consider finding a specific novel hidden in a backpack within a cluttered workspace. The robot alternates between viewpoint adjustment (e.g., resolve occlusions) and exploratory interaction (e.g., open containers, search backpack) in a closed loop. It then assesses whether the evidence in memory is sufficient (e.g., verify retrieved book) and uses its memory to guide subsequent action (e.g., search unexplored regions). Thus, a comprehensive evaluation of active perception should jointly measure *active information acquisition* and *memory-based information maintenance*, as either dimension alone provides only a partial view of such ability.

Recent Vision-Language-Action (VLA) and World-Action Models (WAM) benchmarks have brought memory evaluation in long-horizon embodied tasks to the forefront (Cherepanov et al., 2025; Fang

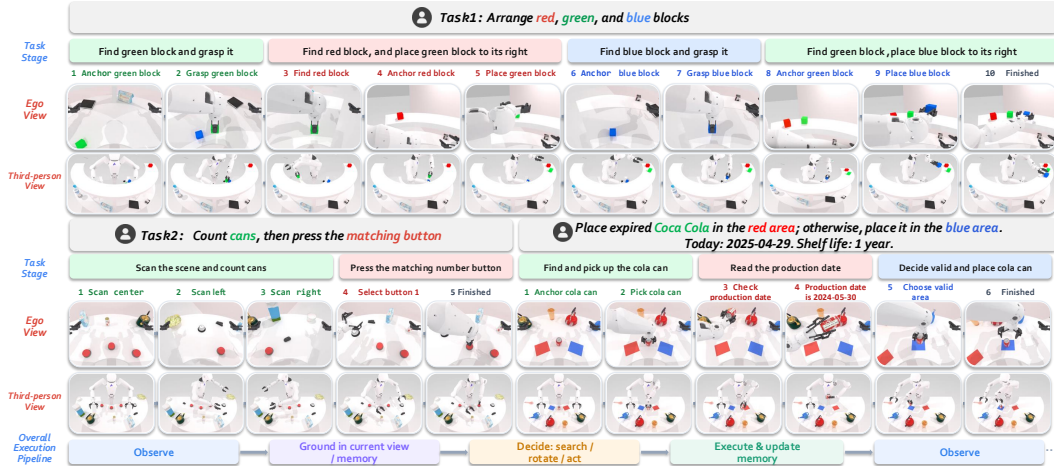


Figure 1: Representative *ActiveArena* rollouts covering 2 families (*i.e.*, visual exploration, interactive acquisition). Each example shows the perception-action loop of evidence grounding, information acquisition, action execution, and memory update by coordinated head/arm actions in a large workspace. The third-person view is included solely for scene visualization.

et al., 2025; Lei et al., 2026; Dai et al., 2026; Chen et al., 2026b; Shah et al., 2026). However, existing benchmarks overlook two key dimensions of active perception: (1) *Active information acquisition*: they assume fixed-view observations are sufficient and do not require robots to seek missing or out-of-view evidence. (2) *Active information maintenance*: they assume complete information in memory, neglecting the selective preservation and updating of information as observations accumulate. These omissions stem from a fundamental limitation of current simulators: fixed viewpoints create an embodiment gap, confined workspaces a scene gap, and limited support for acquiring information an interaction gap. In other words, *despite the importance of active perception, neither a unified simulator addressing all three gaps nor a benchmark systematically evaluating both core dimensions has been explored*.

To this end, we present **ActiveArena**, the first comprehensive and standardized simulator–benchmark–baseline suite to meet above expectation. At its core, **ActiveArena-Sim**, is designed at three levels: (1) *Embodiment*: controllable head and torso joints enable active viewpoint selection, providing 6.3x greater visible-scene coverage than conventional fixed-view simulators. (2) *Scene*: a 180° multi-level workspace 2x the interaction area and distributes task-relevant objects across diverse directions and heights. (3) *Interaction*: robots actively reveal hidden evidence by manipulating objects and inspecting them from informative viewpoints, enabling richer information acquisition (2x) than passive observation. This simulator forms the foundation of the entire suite.

Built on **ActiveArena-Sim**, **ActiveArena-Bench** comprises tasks that are nearly impossible to solve from passive observations alone. It spans two complementary families—visual exploration and interactive acquisition—depending on how task-relevant evidence should be actively revealed. The benchmark further organizes tasks into five categories based on the number of relevant objects and perception–action rounds (up to 9), and jointly assesses active spatial search, cross-view memory, interactive evidence acquisition, and multi-round closed-loop reasoning. It also provides multi-granularity memory annotations to support diverse memory and model design for deep analysis.

As real-world active perception and manipulation demand robust generalization, we establish a rigorous, standardized training and evaluation protocol that strictly separates in-distribution (ID) and out-of-distribution (OOD) settings. The ID setting evaluates the same domain task learning, while the OOD setting introduces unseen distractor layouts and backgrounds to test whether models generalize information-acquisition and memory-maintenance strategies rather than exploit spurious environmental regularities. Our results expose a critical gap: different memory and model designs that achieve strong ID performance do not necessarily generalize OOD robustly in active perception and manipulation.



Figure 2: Overview of *ActiveArena-Bench* and *ActiveArena-Sim*. The third-person view is included solely for scene visualization.

We further introduce **ActiveArena-VLA**, a modular evaluation suite spanning 13 VLA configurations for controlled analysis of memory writing, memory capacity, proprioception, subtask supervision, and high-level planning. Our results reveal that: (1) Uniform temporal sampling generalizes better OOD than event-triggered writing based on subtask transitions or robot motion. (2) Larger memory is not inherently beneficial; its effectiveness depends critically on write-policy stability. (3) Proprioceptive inputs and auxiliary subtask supervision provide additional gains. (4) Explicit planner-mediated memory management and decision-making achieve performance close to our best-performing *ActiveArena-VLA* variant while using only sparse memory. Given that these designs still overcome the ID-OOD gap in active perception, these findings highlight the long-term value of research in this domain.

Our main contributions are threefold: (1) We introduce **ActiveArena-Sim**, a simulator designed for active viewpoint control, large-workspace manipulation, and spatio-temporal memory research. (2) We develop **ActiveArena-Bench**, a unified benchmark of 35 tasks spanning 2 task families and 5 categories, with rich memory annotations, standardized training data, and ID/OOD protocols for evaluating task learning and active-perception generalization. (3) We present **ActiveArena-VLA**, a modular suite of 13 VLA variants, and systematically study how memory writing, memory capacity, proprioception, subtask supervision, and high-level planning shape active-perception performance and generalization.

## 2 ActiveArena

This section presents *ActiveArena*, a simulator-benchmark-baseline suite for active perception and manipulation. We first formulate active-perception manipulation under partial observability. We then elaborate on the design of *ActiveArena-Sim*. Finally, we introduce task-design principles and taxonomy of the *ActiveArena-Bench*. The position of *ActiveArena-Bench* among related benchmarks is summarized in Table 1.

**Problem Formulation.** We consider language-conditioned robotic manipulation under partial observability. At time step  $t$ , the policy receives a visual observation  $o_t$ , an optional proprioceptive state  $x_t$ , and a language instruction  $l$ . The preceding interaction history is

$$h_t = ((o_\tau, x_\tau, \mathbf{a}_\tau))_{\tau=0}^{t-1}, \quad (1)$$

where  $x_\tau$  is omitted when proprioception is unavailable. A method-dependent memory summarizes this history as

$$M_t = \mathcal{M}_\phi(h_t), \quad \mathcal{I}_t = (o_t, x_t, M_t, l). \quad (2)$$

Conceptually, the policy may take an action with one of two functional roles: acquiring task-relevant information or directly advancing task completion. Let  $z_t \in \{\text{info}, \text{task}\}$  denote this role. The next action is then represented as

$$\mathbf{a}_t \sim \begin{cases} \pi_l^{\text{info}}(\cdot | \mathcal{I}_t), & z_t = \text{info}, \\ \pi_l^{\text{task}}(\cdot | \mathcal{I}_t), & z_t = \text{task}. \end{cases} \quad (3)$$

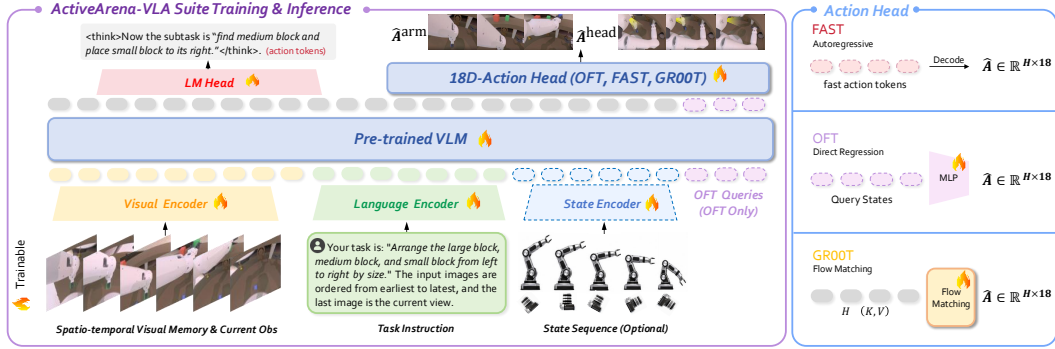


Figure 3: Overview of the *ActiveArena-VLA* framework. Spatio-temporal visual memory, language, and optional proprioceptive state sequences are encoded by a VLM and connected to OFT, FAST, or GR00T action heads for 18-D action prediction.

Here,  $\pi_l^{\text{info}}$  covers actions that expose or localize missing evidence, including viewpoint adjustment, spatial exploration, container opening, and object reorientation. In contrast,  $\pi_l^{\text{task}}$  uses the accumulated evidence to execute the requested manipulation. The policy need not explicitly predict  $z_t$ ; it denotes the functional role of the selected action.

After executing  $\mathbf{a}_t$ , the policy receives  $o_{t+1}$  and updates its memory to  $M_{t+1}$ . Because the same motor primitive may acquire information in one context and advance task completion in another, the two roles are distinguished by their task-level function rather than by disjoint action spaces. Active perception therefore requires the policy to alternate between information acquisition and task execution according to the evidence retained in memory.

Table 1: Comparison of benchmarks across 7 dimensions. **Dynamic Head Viewpoint(DV)**: independently controllable viewpoint changes (excluding motion induced solely by the arm or wrist); **Bimanual(BM)**: bimanual manipulation; **Active Perception(AP)**: task-relevant information not initially visible; **Manipulation Action Output(AO)**: executable manipulation-action output; **Subtask Annotation(SA)**: temporally aligned subtask annotations; **Native Diverse Keyframes(NK)**: native semantic, stage, or event keyframes; **Real-World Counterpart(RW)**: equipped with a corresponding real-robot task, dataset, experiment, or evaluation. ✓/✗ indicate support or not.

| Benchmark                            | DV | BM | AP | AO | SA | NK | RW |
|--------------------------------------|----|----|----|----|----|----|----|
| EmbodiedEval (Cheng et al., 2025)    | ✓  | ✗  | ✓  | ✗  | ✗  | ✗  | ✗  |
| CHAIN (Wu et al., 2026)              | ✗  | ✗  | ✓  | ✗  | ✗  | ✗  | ✗  |
| ESI-Bench (Hong et al., 2026)        | ✓  | ✗  | ✓  | ✗  | ✗  | ✗  | ✗  |
| RoboTwin 2.0 (Chen et al., 2025)     | ✗  | ✓  | ✗  | ✓  | ✗  | ✗  | ✓  |
| RoboDojo (Chen et al., 2026a)        | ✗  | ✓  | ✗  | ✓  | ✗  | ✗  | ✓  |
| LIBERO (Liu et al., 2023)            | ✗  | ✗  | ✗  | ✓  | ✗  | ✗  | ✗  |
| LIBERO-Plus (Fei et al., 2025)       | ✗  | ✗  | ✗  | ✓  | ✗  | ✗  | ✗  |
| MIKASA (Cherepanov et al., 2025)     | ✗  | ✗  | ✗  | ✓  | ✗  | ✗  | ✓  |
| RoboMemArena (Lei et al., 2026)      | ✗  | ✗  | ✗  | ✓  | ✓  | ✓  | ✓  |
| RoboMME (Dai et al., 2026)           | ✗  | ✗  | ✗  | ✓  | ✓  | ✓  | ✓  |
| RMBench (Chen et al., 2026b)         | ✗  | ✓  | ✗  | ✓  | ✓  | ✗  | ✓  |
| MemoryBench (Fang et al., 2025)      | ✗  | ✗  | ✗  | ✓  | ✗  | ✗  | ✓  |
| ReMemBench (Shah et al., 2026)       | ✓  | ✗  | ✗  | ✓  | ✗  | ✗  | ✓  |
| RoboCasa365 (Nasiriany et al., 2026) | ✓  | ✗  | ✗  | ✓  | ✓  | ✗  | ✓  |
| EFM-10 (He et al., 2026)             | ✗  | ✓  | ✓  | ✓  | ✗  | ✗  | ✓  |
| <b>ActiveArena-Bench</b>             | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  |

**Simulation Workspace Design.** Upon RoboTwin 2.0 (Chen et al., 2025), we redesign its workspace to make active information acquisition necessary. In the original setup, manipulation tasks are concentrated on a local tabletop directly in front of the robot, and task-relevant objects are observable



from a fixed camera. Although suitable for evaluating basic manipulation skills, this configuration provides limited support for tasks that genuinely require active perception. We replace the tabletop with a workspace whose horizontal footprint spans an approximately  $180^\circ$  sector centered on the robot, and parameterize object placement using robot-centric cylindrical coordinates. No single fixed view covers the entire scene, requiring the robot to rotate its head or torso to search for and revisit task-relevant objects. Such viewpoint changes frequently move previously observed objects outside the current field of view, making cross-view memory essential. For selected tasks, we further introduce a two-tier layout that distributes objects across different heights and spatial regions that cannot be observed simultaneously. The resulting workspace is shown in Fig. 2.

**Embodiment with Active Viewpoint Control.** The default embodiments in RoboTwin 2.0 use fixed-camera configurations and are therefore unsuitable for systematically evaluating active visual search and cross-view manipulation. We integrate the Atribot S1 (Gao et al., 2025) dual-arm humanoid robot into the simulator and enable control over its head and torso degrees of freedom, allowing the robot to actively adjust its viewpoint. The resulting embodiment is shown in Fig. 2. The robot action space contains controls for both arms, both grippers, torso, and head, yielding an 18-D action vector:

$$\mathbf{a}_t = [\mathbf{a}_t^L, g_t^L, \mathbf{a}_t^R, g_t^R, a_t^{\text{torso}}, a_t^{\text{head}}] \in \mathbb{R}^{18}. \quad (4)$$

**Process-Level Annotations.** We extend trajectory logging with process-level annotations for active perception. Each trajectory is decomposed into semantic subtasks and three functional phases: *search*, *anchor*, and *action*. Search acquires missing evidence, anchor establishes or recovers an evidence-bearing view, and action executes manipulation or reveals additional information. Annotations are aligned frame-wise with observations, robot states, and actions. They include target visibility, discovery history, image-space location, camera orientation, and manipulation targets. Visibility is computed from the visible projected area of each 3D bounding box to account for occlusion and image truncation. For cross-view tasks, informative keyframes and their viewing directions remain annotated after the target leaves the current view. For interaction-dependent tasks, such as inspecting the production date on the back of a can, information-revealing actions and inspection views are linked to the decisions they support. These annotations enable training and evaluation of visual search, evidence localization, keyframe selection, memory retrieval, viewpoint recovery, and action prediction without assuming a specific memory architecture.

**Task-Design Principles.** Every task satisfies 4 requirements: (1) *Initial Information Insufficiency*. The correct manipulation cannot be determined from the initial observation alone. (2) *Information Accessibility*. The missing task-relevant information can be acquired through spatial search or physical interaction. (3) *Historical Dependence*. Evidence acquired from previous observations must remain relevant even after the corresponding objects or regions leave the current field of view. (4) *Closed-Loop Decision Making*. Newly acquired evidence must influence subsequent localization or manipulation decisions. After each action, the robot must reassess whether the accumulated evidence is sufficient and continue acquiring information when necessary.

**Task Taxonomy.** Following these principles, we construct 35 simulated manipulation tasks. At the coarsest level, the tasks are divided into two families: Visual Search and Interactive Information Acquisition. Visual-search tasks are further organized according to the number of task-relevant objects and the number of required perception–action loops, resulting in five categories overall: (1) *Single-Object Search (SS)*. Locate a single target and immediately perform the required manipulation. These tasks require the robot to use its limited memory to identify unexplored regions. (2) *Single-Object Loop (SL)*. Complete cross-region search and the transport of a single object. (3) *Multi-Object Decision (MD)*. Search among multiple candidates and select the correct target or action from the acquired evidence. (4) *Multi-Object Loop (ML)*. Complete multiple perception–action loops across objects and spatial regions. (5) *Interactive Information Acquisition (IA)*. The robot must physically interact with the environment to reveal otherwise hidden task-relevant information before determining the appropriate manipulation. Representative task rollouts are provided in Fig. 1.

### 3 EXPERIMENTS

**Data and Evaluation Protocol.** Evaluation confined to the training scene distribution can conflate in-distribution imitation with genuine policy generalization. To disentangle imitation learning from generalization, we introduce two evaluation settings: ID and OOD. For benchmark adaptation, every

method is trained on the same set of 100 ID trajectories per task. Across 35 tasks, the training set contains a total of 581.2k frames, corresponding to 10.76 hours of video. Each model is trained for 3.9 epochs while consuming only the annotation fields required by its architecture. During evaluation, all models are tested using the same set of 50 randomly generated seeds for each of the ID and OOD settings. ID scenes contain randomized task-irrelevant distractors sampled from a restricted asset pool that excludes all object categories used as task-relevant objects anywhere in the benchmark. The OOD setting additionally randomizes scene backgrounds and illumination and introduces substantially denser clutter. OOD distractors are sampled from the full asset pool, excluding only objects already instantiated in the current scene and synonymous or near-duplicate assets, so objects that serve as task-relevant targets in other tasks can appear as distractors. Detailed evaluation configurations are provided in the appendix.

**Baselines and *ActiveArena*-VLA Suite.** We evaluate six external baselines spanning three representative policy families. Large-scale pretrained VLA models include  $\pi_{0.5}$  (Black et al., 2025) and SaPaVe (Liu et al., 2026a). Memory-aware policies include HiF-VLA (Lin et al., 2025), MemoryVLA (Shi et al., 2025a), and MemER (Sridhar et al., 2025). We additionally evaluate FastWAM (Yuan et al., 2026b) as a representative world-action model. We construct *ActiveArena*-VLA, a suite of 13 VLA configurations built upon the modular StarVLA (Community, 2026) framework. An overview of the suite is shown in Fig. 3. *ActiveArena*-VLA is designed to isolate the effects of memory write policy, memory capacity, proprioceptive state input, auxiliary VLM supervision, high-level planning, and action decoding. *ActiveArena*-Fixed is a variant of *ActiveArena*-OFT that removes only the active-viewpoint action dimensions. Representative configurations are included in the main leaderboard, and the complete model specifications and evaluation details are provided in the appendix. Table 2 presents the simulation benchmark leaderboard. For each task category, ID and OOD success rates are reported side by side to characterize both in-distribution task learning and robustness to distribution shift. The results show that models with memory generally exhibit stronger generalization on active-perception tasks, and actively controllable viewpoint movement is essential.

Table 2: Simulation benchmark leaderboard in task success rate (%). Each category score is averaged over all tasks within that category, while Avg. is averaged over all 35 tasks. Best and second-best results are shown in bold and underlined, respectively.

| Method                           | Memory | SS           |              | SL           |              | ML           |              | MD           |              | IA           |              | Avg.         |              |
|----------------------------------|--------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
|                                  |        | ID           | OOD          | ID           | OOD          | ID           | OOD          | ID           | OOD          | ID           | OOD          | ID           | OOD          |
| FAST-WAM (Yuan et al., 2026b)    | No     | 59.60        | 4.00         | 40.38        | 1.00         | 25.67        | 0.00         | 12.00        | 3.33         | 22.40        | 5.20         | 35.60        | 2.06         |
| $\pi_{0.5}$ (Black et al., 2025) | No     | 41.20        | 5.60         | 22.38        | 0.63         | 1.67         | 0.00         | 2.67         | 5.33         | 1.60         | 1.20         | 16.86        | 1.71         |
| SaPaVe (Liu et al., 2026a)       | No     | 48.80        | 40.80        | 19.88        | 5.88         | 10.67        | 3.00         | 25.33        | 12.67        | 6.00         | 2.80         | 20.91        | 10.51        |
| HiF-VLA (Lin et al., 2025)       | Yes    | 16.80        | 14.00        | 6.88         | 1.13         | 0.00         | 0.00         | 30.67        | 18.67        | 17.60        | 17.20        | 10.69        | 6.57         |
| MemER (Sridhar et al., 2025)     | Yes    | <b>81.20</b> | 66.00        | 33.88        | 18.63        | 18.33        | 5.67         | 40.00        | <b>42.00</b> | 11.60        | 6.40         | 35.31        | 23.43        |
| MemoryVLA (Shi et al., 2025a)    | Yes    | 46.00        | 34.40        | 20.88        | 6.50         | 14.00        | 2.00         | 28.00        | 27.33        | 9.20         | 4.40         | 22.23        | 11.20        |
| <i>ActiveArena</i> -Fixed        | Yes    | 1.20         | 0.40         | 0.00         | 0.00         | 0.00         | 0.00         | 1.33         | 0.67         | 0.40         | 0.40         | 0.34         | 0.17         |
| <i>ActiveArena</i> -OFT          | Yes    | <u>80.40</u> | <b>70.80</b> | <b>74.50</b> | <b>54.38</b> | <b>53.33</b> | <b>33.33</b> | <b>44.00</b> | <u>35.33</u> | <b>31.60</b> | <b>27.20</b> | <b>62.97</b> | <b>47.60</b> |
| <i>ActiveArena</i> -Plan         | Yes    | 79.20        | <u>69.60</u> | <u>73.38</u> | <u>52.63</u> | <u>49.67</u> | <u>30.00</u> | <u>42.00</u> | 33.33        | <u>28.00</u> | <u>24.80</u> | <u>60.97</u> | <u>45.54</u> |

## 4 ANALYSIS AND ABLATIONS

### 4.1 FAILURE-STATE DECOMPOSITION.

We partition failed episodes into three mutually exclusive states. Discovery Miss (DM) indicates that the policy never observes all task-relevant objects; high DM rate therefore suggests that failures primarily arise from insufficient active perception. Context Omission (CO) indicates that all task-relevant objects were observed previously but are not jointly covered by the final observation; high CO rate suggests that failures mainly stem from the inability to actively retrieve relevant information from memory. Context Retained (CR) indicates that the final observation covers all task-relevant objects, yet the task still fails, suggesting errors in downstream decision-making or control. As shown in Table 3, *ActiveArena*-OFT and *ActiveArena*-Plan substantially reduce both DM and CO,

demonstrating more reliable evidence acquisition and availability at the final decision point. Their remaining failures are predominantly CR, indicating that the primary bottleneck lies in downstream reasoning and execution rather than active perception. The results of *ActiveArena*-Fixed demonstrate that active viewpoint control is essential for completing the task.

Table 3: Failure-state decomposition among unsuccessful episodes (%): discovery miss (DM), context omission (CO), and context retained (CR).

| Method                    | Mem. | DM   |      | CO   |      | CR   |      |
|---------------------------|------|------|------|------|------|------|------|
|                           |      | ID   | OOD  | ID   | OOD  | ID   | OOD  |
| FAST-WAM                  | No   | 13.2 | 20.6 | 34.6 | 47.2 | 52.2 | 32.2 |
| $\pi_{0.5}$               | No   | 20.9 | 25.2 | 40.7 | 43.8 | 38.4 | 31.0 |
| SaPaVe                    | No   | 12.6 | 13.9 | 23.7 | 33.6 | 63.7 | 52.5 |
| HiF-VLA                   | Yes  | 15.8 | 17.0 | 21.4 | 20.9 | 62.8 | 62.1 |
| MemER                     | Yes  | 9.1  | 13.5 | 13.9 | 16.0 | 77.0 | 70.5 |
| MemoryVLA                 | Yes  | 18.0 | 20.2 | 21.3 | 17.3 | 60.7 | 62.5 |
| <i>ActiveArena</i> -Fixed | Yes  | 96.3 | 97.3 | 0.0  | 0.0  | 3.7  | 2.7  |
| <i>ActiveArena</i> -OFT   | Yes  | 2.3  | 3.2  | 6.2  | 7.3  | 91.5 | 89.5 |
| <i>ActiveArena</i> -Plan  | Yes  | 3.0  | 4.7  | 10.9 | 12.9 | 86.1 | 82.4 |

#### 4.2 DO THE TRENDS OBSERVED ON *ActiveArena*-BENCH TRANSFER TO REAL-WORLD ROBOTIC MANIPULATION?

We design eight real-world manipulation tasks and collect a total of 640 trajectories across them to train the evaluated policies. We then evaluate  $\pi_{0.5}$ , SaPaVe, MemER and *ActiveArena*-OFT on these tasks, conducting 20 evaluation trials per task for each policy. The results show that the memory-based model has a clear advantage on tasks requiring active perception, and the relative performance trends among the evaluated policies are consistent with those observed in *ActiveArena*-Sim. These findings indicate that *ActiveArena* captures challenges that are also relevant to physical robotic manipulation. Representative real-world rollouts are shown in Fig. 4. Detailed setup and task definitions are provided in the appendix.

Table 4: Real-world task success rates (%). Instructions are shortened. All tasks require interacting with containers or occluding objects to locate the target object, followed by finding the basket and placing the object into it. The average is computed over all eight tasks.

| Instruction                 | $\pi_{0.5}$ | SaPaVe | MemER | <i>ActiveArena</i> -OFT |
|-----------------------------|-------------|--------|-------|-------------------------|
| Place pen in basket         | 45          | 55     | 65    | <b>75</b>               |
| Place peach in basket       | 30          | 35     | 50    | <b>70</b>               |
| Place screwdriver in basket | 25          | 15     | 30    | <b>55</b>               |
| Place XL T-shirt in basket  | 0           | 5      | 10    | <b>40</b>               |
| ...                         |             |        |       |                         |
| Average                     | 23.8        | 24.4   | 34.4  | <b>55.6</b>             |

#### 4.3 EFFECT OF MEMORY WRITE POLICY AND CAPACITY.

Perceptual memory is jointly governed by its write policy and capacity. We evaluate three memory write policies: *subtask*, which writes a frame when the predicted subtask changes; *motion*, which writes a frame upon gripper transitions or thresholded end-effector, head, or torso motion; and *chunk*, which writes one frame after each action chunk. Combining each policy with memory capacities of 6 and 12 frames yields six *ActiveArena*-OFT variants. We also include a memory-free OFT (Kim et al.,

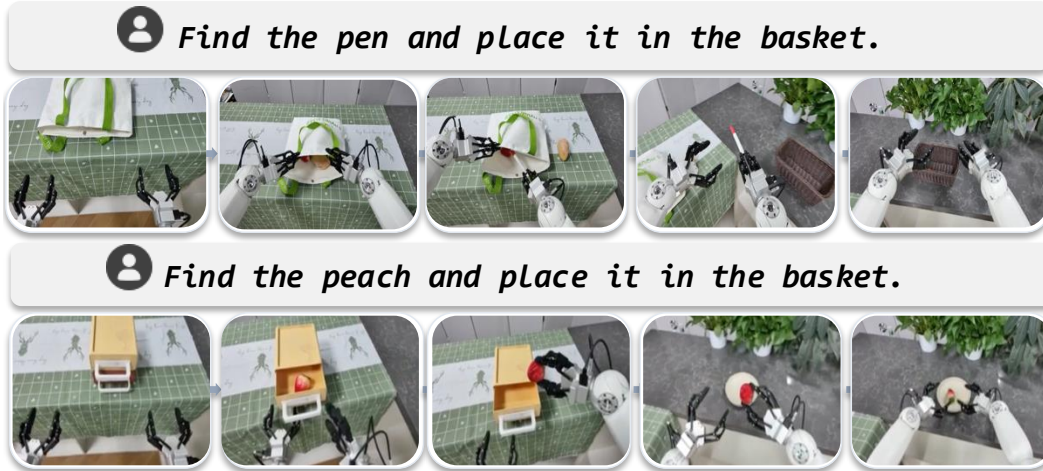


Figure 4: Two representative rollouts from two of the eight real-world manipulation tasks.

2025) baseline. All variants share the same task instruction, 18-D proprioceptive state, OFT action head, and subtask-supervised VLM objective. Results are reported in Table 5.

Table 5: Effect of memory write policy and capacity on task success rate (%). Category scores are averaged over their corresponding tasks, while Avg. is the task-macro average over all 35 tasks (5 SS, 16 SL, 6 ML, 3 MD, and 5 IA). Best and second-best results are shown in bold and underlined, respectively.

| Memory Write Policy | Capacity | SS           |              | SL           |              | ML           |              | MD           |              | IA           |              | Avg.         |              |
|---------------------|----------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
|                     |          | ID           | OOD          | ID           | OOD          | ID           | OOD          | ID           | OOD          | ID           | OOD          | ID           | OOD          |
| None                | 0        | 58.80        | 15.20        | 27.63        | 2.13         | 13.67        | 0.00         | 36.67        | 24.00        | 6.00         | 6.40         | 27.37        | 6.12         |
| Subtask             | 6        | <u>66.40</u> | 60.80        | 29.50        | 19.13        | 0.00         | 0.00         | 16.00        | 13.33        | 0.80         | 0.80         | 24.46        | 18.69        |
|                     | 12       | 62.80        | 54.40        | 37.25        | 15.00        | 0.67         | 0.33         | 21.33        | 15.33        | 1.20         | 0.00         | 28.11        | 16.00        |
| Motion              | 6        | 62.40        | 54.80        | 45.63        | 28.50        | 20.33        | 10.67        | 36.00        | 32.67        | 8.40         | 6.40         | 37.54        | 26.40        |
|                     | 12       | 58.00        | 47.60        | 36.75        | 20.75        | 24.00        | 11.67        | 22.00        | 30.67        | 12.00        | 12.40        | 32.80        | 22.69        |
| Chunk               | 6        | <b>80.40</b> | <u>63.60</u> | <u>65.25</u> | <u>40.88</u> | <u>49.00</u> | <u>21.67</u> | <b>50.67</b> | <b>46.00</b> | <b>32.40</b> | <u>23.20</u> | <u>58.69</u> | <u>38.74</u> |
|                     | 12       | <b>80.40</b> | <b>70.80</b> | <b>74.50</b> | <b>54.38</b> | <b>53.33</b> | <b>33.33</b> | <u>44.00</u> | <u>35.33</u> | <u>31.60</u> | <b>27.20</b> | <b>62.97</b> | <b>47.60</b> |

As shown in Fig. 5, the memory-free policy cannot track explored regions or plan subsequent searches, highlighting the importance of memory for closed-loop active perception. This finding is consistent with the results in Table 2. The two event-triggered write policies show limited robustness. The subtask-based policy degrades substantially from SS to IA and ML. Its sparse, prediction-dependent triggers can mistime memory updates and omit informative views. The motion-based policy is sensitive to control jitter, which introduces redundant or irrelevant frames. Increasing memory capacity retains more of this noise and reduces performance in most settings. Both policies degrade further under OOD evaluation, indicating their sensitivity to prediction errors and control noise. The chunk-based write policy achieves the best performance across all task-evaluation pairs. Its regular update schedule avoids semantic predictions and motion thresholds, providing stable temporal coverage and stronger OOD robustness. Increasing its capacity from 6 to 12 frames improves six of ten settings, with the largest gains on OOD SL and ML. A reliable write policy can thus exploit additional capacity to improve generalization on complex tasks. Overall, the memory write policy is more critical than capacity alone. Additional capacity yields consistent gains only when task-relevant observations are written reliably.

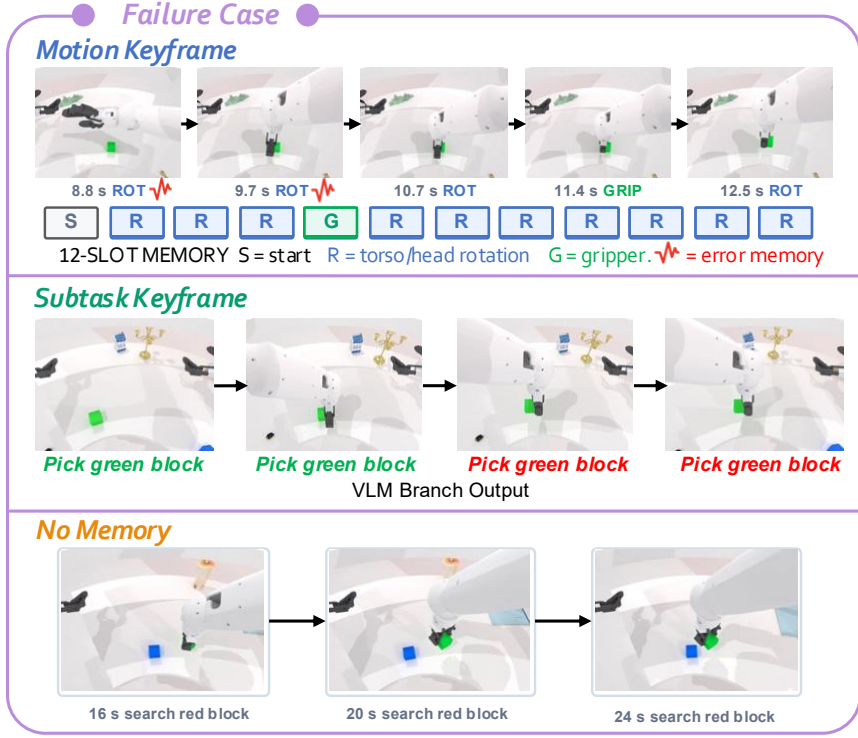


Figure 5: Failure cases of three memory writing policies.

#### 4.4 PLANNER-MEDIATED MEMORY MANAGEMENT.

Recently, many models adopt hierarchical frameworks in which a high-level planner manages memory inputs and decomposes tasks (Shi et al., 2025b; Sridhar et al., 2025; Chen et al., 2026b). To evaluate the effectiveness of planner-based architectures, we introduce *ActiveArena-Plan*. The planner is fine-tuned from Qwen3-VL-2B (Bai et al., 2025) and takes as input the global task instruction, the current observation, and a rolling history of up to 12 frames sampled at action-chunk boundaries. It predicts the current subtask instruction and selects a sparse subset of candidate frames for long-term memory. The downstream VLA then predicts the next action chunk conditioned on the current observation, selected memory frames, predicted subtask instruction, and proprioceptive state. Its architecture is illustrated in Fig. 6. Architectural and training details are provided in the appendix.

The results in Table 6 show that planner-mediated subtask inference and memory writing are more stable than jointly coupling these functions with low-level policy, while retaining near-best performance with sparse memory and reducing peak GPU memory usage during VLA training by 46%.

Table 6: Comparison of planner-mediated memory management and memory write policies on task success rate (%).

| Method                  | Memory Management | ID           | OOD          |
|-------------------------|-------------------|--------------|--------------|
| <i>ActiveArena-OFT</i>  | Chunk Writing     | <b>62.97</b> | <b>47.60</b> |
| <i>ActiveArena-OFT</i>  | Subtask Writing   | 28.11        | 16.00        |
| <i>ActiveArena-Plan</i> | Planner Selection | <u>60.97</u> | <u>45.54</u> |



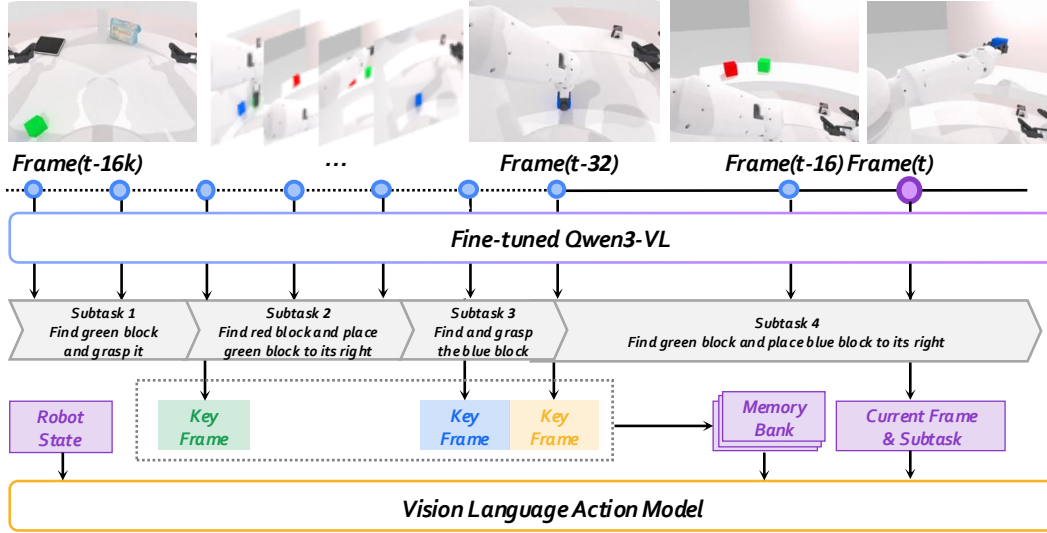


Figure 6: *ActiveArena-Plan*. The Planner identifies historical frames that span subtask transitions and adds them to the memory bank, while inferring the current subtask.

#### 4.5 EFFECT OF VLM BRANCH SUPERVISION.

To examine whether stronger task understanding in the VLM branch improves policy generalization, we compare three *ActiveArena-VLA* variants: no auxiliary VLM supervision, task-level instruction supervision, and subtask-level supervision. All other components are held fixed, including the OFT architecture, 12-frame Chunk memory, and explicit proprioceptive state conditioning. The results in Table 7 show that fine-grained subtask supervision improves generalization under changes in scene appearance and distractor composition.

Table 7: Comparison of different auxiliary VLM supervision on task success rate (%). All variants’ VLM branch uses next-token-prediction loss while keeping the same action loss.

| Method                 | VLM Supervision | ID           | OOD          |
|------------------------|-----------------|--------------|--------------|
| <i>ActiveArena-OFT</i> | None            | 58.57        | 41.60        |
|                        | Task-level      | <u>60.06</u> | 39.54        |
|                        | Subtask-level   | <b>62.97</b> | <b>47.60</b> |

#### 4.6 EFFECT OF PROPRIOCEPTIVE STATE CONDITIONING.

For each visual frame, we encode the corresponding 18-D proprioceptive state with an MLP and fuse the resulting state embedding with the visual and language tokens (Yuan et al., 2026a). We ablate proprioceptive state conditioning while holding the remaining *ActiveArena-OFT* configuration fixed. Table 8 reports the task success rates, and Fig. 7 analyzes its effect on closed-loop action continuity.

Table 8: Comparison of explicit proprioceptive state conditioning on task success rate (%).

| Method                 | State Input | ID           | OOD          |
|------------------------|-------------|--------------|--------------|
| <i>ActiveArena-OFT</i> | No          | 50.86        | 16.86        |
|                        | Yes         | <b>62.97</b> | <b>47.60</b> |

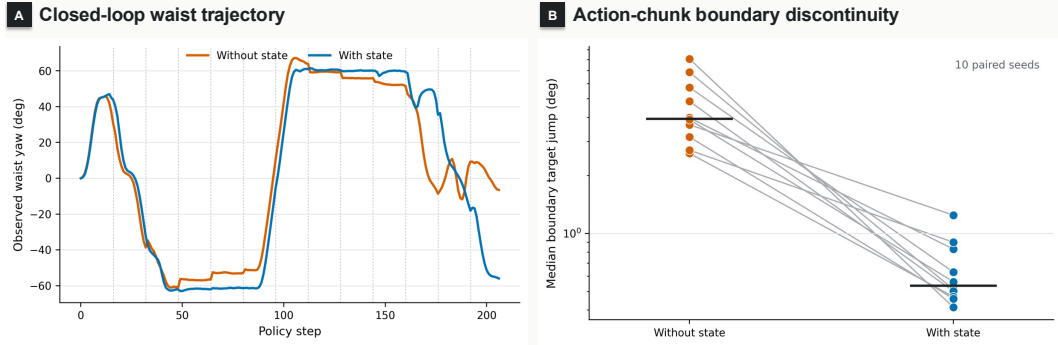


Figure 7: Effect of proprioceptive state conditioning on action continuity. Left: torso target trajectory, with dashed lines indicating 16-step action-chunk boundaries. Right: episode-level median target discontinuity at chunk boundaries across 10 paired seeds; gray lines connect identical seeds.

To examine this effect at the action level, we evaluate both variants on the *place-object-stand-rotate-view* task using the same 10 random seeds. Fig. 7 shows that proprioceptive state conditioning substantially reduces discontinuities at action-chunk boundaries, lowering the mean episode-level median target jump from  $4.55^\circ$  to  $0.65^\circ$ . Explicit state information anchors each predicted chunk to the current robot configuration, reducing the ambiguity of inferring posture from visual observations and producing smoother closed-loop control.

#### 4.7 EFFECT OF ACTION HEAD.

To isolate the effect of action generation, we compare three representative action heads while holding the VLM backbone, memory configuration, training data, and input modalities fixed. The controls for both arms, both grippers, the torso, and the head are jointly represented as an 18-D action vector. OFT (Kim et al., 2025) uses an MLP to regress continuous action chunks in parallel from action query token representations. FAST (Pertsch et al., 2025) discretizes action chunks and predicts the resulting action tokens autoregressively. GR00T (Nvidia et al., 2025) uses a DiT-based flow-matching expert to generate continuous action chunks. Table 9 reports their ID and OOD performance. GR00T’s failures mainly stem from action noise, which typically requires large-scale pretraining to mitigate, whereas FAST’s failures are primarily caused by decoding instability.

Table 9: Comparison of different action-head architectures on task success rate (%).

| Method                    | ID           | OOD          |
|---------------------------|--------------|--------------|
| <i>ActiveArena</i> -OFT   | <b>62.97</b> | <b>47.60</b> |
| <i>ActiveArena</i> -GR00T | <u>40.00</u> | <u>13.43</u> |
| <i>ActiveArena</i> -FAST  | 22.34        | 12.00        |

## 5 CONCLUSION

In this work, we present *ActiveArena*, a comprehensive simulator–benchmark–baseline suite comprising *ActiveArena*-Sim, *ActiveArena*-Bench, and *ActiveArena*-VLA to study active perception and manipulation. Our systematic analysis reveals that principled memory and model design—including memory capacity, maintenance strategies, multimodal integration, supervision, and hierarchical architectures—is fundamental to acquiring and retaining task-relevant information over long horizons. Despite these advances, a substantial ID–OOD gap persists, highlighting the inherent challenges of active perception and manipulation and motivating future research. Future work may explore more effective multimodal memory mechanisms and more scalable system architectures, as well as pre-training paradigms that endow embodied agents with active perception and manipulation capabilities from large-scale egocentric data.

---

## REFERENCES

- Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.
- Ruzena Bajcsy. Active perception. *Proceedings of the IEEE*, 76(8):966–1005, 1988.
- Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al.  $\pi_0$ : A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.
- Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y Galliker, et al.  $\pi_{0.5}$ : A vision-language-action model with open-world generalization. In *9th Annual Conference on Robot Learning*, 2025.
- Tianxing Chen, Zanxin 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.
- Tianxing Chen, Yue Chen, Zixuan Li, Junyuan Tang, Kailun Su, Weijie Wan, Baijun Chen, Haoran Lu, Haowen Yan, Honghao Su, et al. Robodojo: A unified sim-and-real benchmark for comprehensive evaluation of generalist robot manipulation policies. *arXiv preprint arXiv:2607.04434*, 2026a.
- Tianxing Chen, Yuran Wang, Mingleyang Li, Yan Qin, Hao Shi, Zixuan Li, Yifan Hu, Yingsheng Zhang, Kaixuan Wang, Yue Chen, et al. Rmbench: Memory-dependent robotic manipulation benchmark with insights into policy design. *arXiv preprint arXiv:2603.01229*, 2026b.
- Zhili Cheng, Yuge Tu, Ran Li, Shiqi Dai, Jinyi Hu, Shengding Hu, Jiahao Li, Yang Shi, Tianyu Yu, Weize Chen, et al. Embodiedeval: Evaluate multimodal llms as embodied agents. *arXiv preprint arXiv:2501.11858*, 2025.
- Egor Cherepanov, Nikita Kachaev, Alexey Kovalev, and Aleksandr Panov. Memory, benchmark & robots: A benchmark for solving complex tasks with reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2025.
- StarVLA Community. Starvla: A lego-like codebase for vision-language-action model developing. *arXiv preprint arXiv:2604.05014*, 2026.
- Yinpei Dai, Hongze Fu, Jayjun Lee, Yuejiang Liu, Haoran Zhang, Jianing Yang, Chelsea Finn, Nima Fazeli, and Joyce Chai. Robomme: Benchmarking and understanding memory for robotic generalist policies. *arXiv preprint arXiv:2603.04639*, 2026.
- Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–10, 2018.
- Haoquan Fang, Markus Grotz, Wilbert Pumacay, Yi Ru Wang, Dieter Fox, Ranjay Krishna, and Jiafei Duan. Sam2act: Integrating visual foundation model with a memory architecture for robotic manipulation. *arXiv preprint arXiv:2501.18564*, 2025.
- Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, et al. Libero-plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.
- Guang Gao, Jianan Wang, Jinbo Zuo, Junnan Jiang, Jingfan Zhang, Xianwen Zeng, Yuejiang Zhu, Lianyang Ma, Ke Chen, Minhua Sheng, et al. Towards human-level intelligence via human-like whole-body manipulation. *arXiv preprint arXiv:2507.17141*, 2025.
- Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, et al. Maniskill2: A unified benchmark for generalizable manipulation skills. *arXiv preprint arXiv:2302.04659*, 2023.

- 
- Yi Han, Enshen Zhou, Shanyu Rong, Jingkun An, Pengwei Wang, Zhongyuan Wang, Cheng Chi, Lu Sheng, and Shanghang Zhang. Tiger: Tool-integrated geometric reasoning in vision-language models for robotics. *arXiv preprint arXiv:2510.07181*, 2025.
- Yuxin He, Ruihao Zhang, Tianao Shen, Cheng Liu, and Qiang Nie. Towards exploratory and focused manipulation with bimanual active perception: A new problem, benchmark and strategy. *arXiv preprint arXiv:2602.01939*, 2026.
- Yining Hong, Jiageng Liu, Han Yin, Manling Li, Leonidas Guibas, Li Fei-Fei, Jiajun Wu, and Yejin Choi. Esi-bench: Towards embodied spatial intelligence that closes the perception-action loop. *arXiv preprint arXiv:2605.18746*, 2026.
- Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.
- Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. Vima: General robot manipulation with multimodal prompts, 2023. URL <https://arxiv.org/abs/2210.03094>.
- Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.
- Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.
- Huashuo Lei, Wenxuan Song, Huarui Zhang, Jieyuan Pei, Jiayi Chen, Haodong Yan, Han Zhao, Pengxiang Ding, Zhipeng Zhang, Lida Huang, et al. Robomemarena: A comprehensive and challenging robotic memory benchmark. *arXiv preprint arXiv:2605.10921*, 2026.
- Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabriel Levine, Wensi Ai, Benjamin Martinez, et al. Behavior-1k: A human-centered, embodied ai benchmark with 1,000 everyday activities and realistic simulation. *arXiv preprint arXiv:2403.09227*, 2024.
- Hang Li, Fengyi Shen, Dong Chen, Liudi Yang, Xudong Wang, Jinkui Shi, Zhenshan Bing, Ziyuan Liu, and Alois Knoll. Remem-vla: Empowering vision-language-action model with memory via dual-level recurrent queries. *arXiv preprint arXiv:2603.12942*, 2026a.
- Jialiang Li, Yi Qiao, Yunhan Guo, Changwen Chen, and Wenzhao Lian. Act, sense, act: Learning non-markovian active perception strategies from large-scale egocentric human data. *arXiv preprint arXiv:2602.04600*, 2026b.
- Minghui Lin, Pengxiang Ding, Shu Wang, Zifeng Zhuang, Yang Liu, Xinyang Tong, Wenxuan Song, Shangke Lyu, Siteng Huang, and Donglin Wang. Hif-vla: Hindsight, insight and foresight through motion representation for vision-language-action models. *arXiv preprint arXiv:2512.09928*, 2025.
- Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.
- Mengzhen Liu, Enshen Zhou, Cheng Chi, Yi Han, Shanyu Rong, Liming Chen, Pengwei Wang, Zhongyuan Wang, and Shanghang Zhang. Sapave: Towards active perception and manipulation in vision-language-action models for robotics. *arXiv preprint arXiv:2603.12193*, 2026a.
- Zhenyang Liu, Yongchong Gu, Yikai Wang, Xiangyang Xue, and Yanwei Fu. Activevla: Injecting active perception into vision-language-action models for precise 3d robotic manipulation. *arXiv preprint arXiv:2601.08325*, 2026b.

- 
- Arjun Majumdar, Anurag Ajay, Xiaohan Zhang, Pranav Putta, Sriram Yenamandra, Mikael Henaff, Sneha Silwal, Paul Mcvay, Oleksandr Maksymets, Sergio Arnaud, Karmesh Yadav, Qiyang Li, Ben Newman, Mohit Sharma, Vincent Berges, Shiqi Zhang, Pulkit Agrawal, Yonatan Bisk, Dhruv Batra, Mrinal Kalakrishnan, Franziska Meier, Chris Paxton, Sasha Sax, and Aravind Rajeswaran. Openeqa: Embodied question answering in the era of foundation models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334, 2022.
- Soroush Nasiriany, Sepehr Nasiriany, Abhiram Maddukuri, and Yuke Zhu. Robocasa365: A large-scale simulation framework for training and benchmarking generalist robots. *arXiv preprint arXiv:2603.04356*, 2026.
- J Bjorck Nvidia, Fernando Castaneda, N Cherniadev, X Da, R Ding, L Fan, Y Fang, D Fox, F Hu, S Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.
- Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.
- Rutav Shah, Rajat Kumar Jenamani, Xiaohan Zhang, Lingfeng Sun, Roberto Martín-Martín, Yuke Zhu, Deva Ramanan, and Karl Schmeckpeper. Scaling short-term memory of visuomotor policies for long-horizon tasks. *arXiv preprint arXiv:2606.16178*, 2026.
- Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. Memoryvla: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2508.19236*, 2025a.
- Lucy Xiaoyang Shi, Brian Ichter, Michael Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, et al. Hi robot: Open-ended instruction following with hierarchical vision-language-action models. *arXiv preprint arXiv:2502.19417*, 2025b.
- Ajay Sridhar, Jennifer Pan, Satvik Sharma, and Chelsea Finn. Memer: Scaling up memory for robot control via experience retrieval. *arXiv preprint arXiv:2510.20328*, 2025.
- Huajie Tan, Enshen Zhou, Zhiyu Li, Yijie Xu, Yuheng Ji, Xiansheng Chen, Cheng Chi, Pengwei Wang, Huizhu Jia, Yulong Ao, et al. Robobrain 2.5: Depth in sight, time in mind. *arXiv preprint arXiv:2601.14352*, 2026.
- Xi Tang, Jihao Qiu, Lingxi Xie, Yunjie Tian, Jianbin Jiao, and Qixiang Ye. Adaptive keyframe sampling for long video understanding, 2025. URL <https://arxiv.org/abs/2502.21271>.
- Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.
- Qineng Wang, Baiqiao Yin, Pingyue Zhang, Jianshu Zhang, Kangrui Wang, Zihan Wang, Jieyu Zhang, Keshigeyan Chandrasegaran, Han Liu, Ranjay Krishna, et al. Mindcube: Spatial mental modeling from limited views. *arXiv preprint arXiv:2506.21458*, 2026.
- Yuhao Wu, Maojia Song, Yihuai Lan, Lei Wang, Zhiqiang Hu, Yao Xiao, Heng Zhou, Weihua Zheng, Dylan Raharja, Soujanya Poria, et al. From perception to action: An interactive benchmark for vision reasoning. *arXiv preprint arXiv:2602.21015*, 2026.
- Haoyu Xiong, Xiaomeng Xu, Jimmy Wu, Yifan Hou, Jeannette Bohg, and Shuran Song. Vision in action: Learning active perception from human demonstrations. *arXiv preprint arXiv:2506.15666*, 2025.



- 
- Jihan Yang, Shusheng Yang, Anjali W Gupta, Rilyn Han, Li Fei-Fei, and Saining Xie. Thinking in space: How multimodal large language models see, remember, and recall spaces. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 10632–10643, 2025.
- Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pp. 1094–1100. PMLR, 2020.
- Haoqi Yuan, Zhixuan Liang, Anzhe Chen, Ye Wang, Haoyang Li, Pei Lin, Yiyang Huang, Zixing Lei, Tong Zhang, Jiazhao Zhang, et al. Qwen-robotmanip technical report: Alignment unlocks scale for robotic manipulation foundation models. *arXiv preprint arXiv:2606.17846*, 2026a.
- Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-wam: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026b.
- Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé III, Andrey Kolobov, Furong Huang, and Jianwei Yang. Tracevla: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. *arXiv preprint arXiv:2412.10345*, 2024.
- Enshen Zhou, Qi Su, Cheng Chi, Zhizheng Zhang, Zhongyuan Wang, Tiejun Huang, Lu Sheng, and He Wang. Code-as-monitor: Constraint-aware visual programming for reactive and proactive robotic failure detection. *arXiv preprint arXiv:2412.04455*, 2024.
- 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, 2026a.
- Enshen Zhou, Yibo Li, Jingkun An, Jiayuan Zhang, Shanyu Rong, Mengzhen Liu, Yi Han, Yuheng Ji, Huajie Tan, Jiawei He, et al. Towards spatial trace with reasoning in vision-language models for robotics. In *European Conference on Computer Vision*, pp. 380–400. Springer, 2026b.
- Zirui Zhu, Hailun Xu, Yang Luo, Yong Liu, Kanchan Sarkar, Zhenheng Yang, and Yang You. Focus: Efficient keyframe selection for long video understanding, 2025. URL <https://arxiv.org/abs/2510.27280>.
- Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pp. 2165–2183. PMLR, 2023.

---

## APPENDIX

### A RELATED WORK

**Robot Manipulation, Active Perception, and Memory Benchmarks.** Robot manipulation benchmarks have evolved from standardized multi-task control toward language-conditioned, long-horizon, household-scale, and distribution-aware evaluation. Meta-World, RLBench, and ManiSkill established general-purpose testbeds for manipulation learning and transfer (Yu et al., 2020; James et al., 2020; Gu et al., 2023); CALVIN, VIMA, and LIBERO extended this scope to language grounding and compositional task learning (Mees et al., 2022; Jiang et al., 2023; Liu et al., 2023); and recent platforms such as BEHAVIOR-1K, RoboCasa365, RoboTwin 2.0, RoboDojo, and LIBERO-Plus further emphasize household diversity, bimanual control, sim-to-real deployment, and robustness to distribution shifts (Li et al., 2024; Nasiriany et al., 2026; Chen et al., 2025; 2026a; Fei et al., 2025). Despite this progress, most manipulation benchmarks assume that task-relevant evidence is already available in the observation stream and therefore evaluate how a policy acts on perceived information rather than how it acquires missing evidence. Active-perception benchmarks instead treat sensing as part of decision making: EmbodiedQA, OpenEQA, EmbodiedEval, CHAIN, and ESI-Bench require agents to explore or interact with 3D environments to resolve uncertainty (Das et al., 2018; Majumdar et al., 2024; Cheng et al., 2025; Wu et al., 2026; Hong et al., 2026). However, these benchmarks primarily operate through high-level semantic actions and largely abstract away continuous robot control. In parallel, memory-centric manipulation suites evaluate delayed evidence, state aliasing, and history-dependent decisions (Fang et al., 2025; Cherepanov et al., 2025; Lei et al., 2026; Dai et al., 2026; Chen et al., 2026b; Shah et al., 2026). These directions remain only partially unified: memory benchmarks emphasize retaining acquired evidence, whereas active perception additionally requires deciding what evidence to acquire and how to obtain it through closed-loop action. *ActiveArena* integrates incomplete initial evidence, active viewpoint control, physical information acquisition, spatiotemporal memory, and low-level dual-arm manipulation within a standardized ID/OOD evaluation framework.

**Vision–Language–Action Models and Memory.** Vision–language–action (VLA) models have recently enabled generalist robot policies by aligning visual, linguistic, and action representations. Representative systems, including RT-2, Octo, OpenVLA,  $\pi_0$ ,  $\pi_{0.5}$ , and GR00T, explore large-scale pretraining, heterogeneous robot data, and expressive action generation (Zitkovich et al., 2023; Team et al., 2024; Kim et al., 2024; Black et al., 2024; 2025; Nvidia et al., 2025). Related VLM-based approaches address spatial referring, spatial tracing, and geometric reasoning for robotic manipulation (Zhou et al., 2026a;b; Han et al., 2025). Beyond action generation, recent efforts increasingly investigate how VLAs utilize temporal information. Different approaches construct memory through recent visual traces, explicit keyframe banks, semantic retrieval, or recurrent latent states, including TraceVLA, HiF-VLA, MemoryVLA, MemER, and ReMemVLA (Zheng et al., 2024; Lin et al., 2025; Shi et al., 2025a; Sridhar et al., 2025; Li et al., 2026a). These methods demonstrate that retaining historical evidence improves long-horizon manipulation, but they mainly focus on memory utilization after observations are acquired. A complementary direction introduces active sensing policies that explicitly control viewpoints or exploration behaviors, such as SaPaVe and CoMe-VLA (Liu et al., 2026a; Li et al., 2026b). Visual feedback also supports execution progress estimation (Tan et al., 2026) and constraint-aware failure monitoring (Zhou et al., 2024) in robotic manipulation. Nevertheless, existing VLA evaluations lack a unified framework for studying how a policy decides what information to acquire, how to encode acquired evidence, and how memory interacts with low-level manipulation. *ActiveArena* provides such a testbed by combining active information acquisition, diverse memory annotations, and continuous robot control within a unified benchmark.

### B DETAILS OF *ActiveArena*-SIM

This section summarizes the engineering modifications used to transform RoboTwin 2.0 into *ActiveArena*-Sim. We focus on extensions to the robot embodiment, workspace, task assets, action interface, and annotation pipeline; standard RoboTwin data-generation, domain-randomization, and manipulation APIs are omitted.

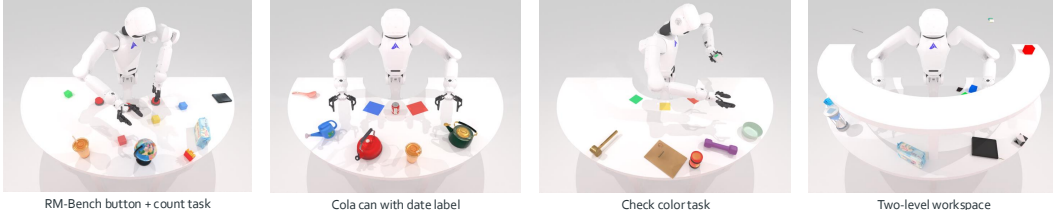


Figure 8: Representative simulator frames. The panels show the RM-Bench button integrated into a counting task, a block with hidden color patch, a can with a runtime-generated date label, and the two-level fan workspace.

### B.1 ASTRIBOT EMBODIMENT ADAPTATION

We add an Astribot S1 whole-body URDF, a manually annotated CuRobo self-collision model, and independent CuRobo planning models for the two arms. The robot is loaded as a single fixed-base articulation at

$$[x, y, z, q_w, q_x, q_y, q_z] = [0, -0.45, -0.15, 0.707, 0, 0, 0.707].$$

This transform aligns the native robot frame with RoboTwin’s world frame and places the base at the center of the fan-shaped workspace. The model comprises two seven-DoF arms, a single-DoF head that rotates vertically in pitch, an actuated torso-yaw joint, and two parallel grippers. We replace the original multi-link gripper actuation with a single-command mimic-joint model: one active joint drives five mimic joints, enabling scalar open/close control.

Table 10: Astribot control parameters introduced in *ActiveArena-Sim*. Angles are in radians. Gripper values follow RoboTwin’s normalized convention, where 0 and 1 denote closed and open, respectively.

| Component       | Parameter                          | Value / implementation                               |
|-----------------|------------------------------------|------------------------------------------------------|
| Left arm        | Home state                         | $[-0.4, -0.7, -1.0, 1.4, -0.9, 0, 0]$                |
| Right arm       | Home state                         | $[0.4, -0.7, 1.0, 1.4, 0.9, 0, 0]$                   |
| Head            | Joint limits                       | pitch $[-1.22, 1.22]$                                |
| Head            | Home; velocity / acceleration      | 1.0; 10 rad/s and 25 rad/s <sup>2</sup>              |
| Torso           | Joint and limits                   | <code>astribot_torso_joint_4</code> , $[-1.2, 1.2]$  |
| Torso           | Home; velocity / acceleration      | 0; 2.5 rad/s and 10 rad/s <sup>2</sup>               |
| Arm drives      | Stiffness / damping                | 2000/200                                             |
| Torso drive     | Stiffness / damping                | 3000/800                                             |
| Gripper drive   | Stiffness / damping / force        | 4000/200/120                                         |
| Gripper mapping | Normalized command to active joint | <code>gripper_scale</code> = $[-1, 0]$ ; bias 0.21 m |

The motion planner uses `astribot_arm_left_link_7` and `astribot_arm_right_link_7` as the left and right move groups. We calibrate RoboTwin’s tool-center-point convention against these end links using

$$R_{\Delta} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \quad R_G = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix}. \quad (5)$$

Before planning, the TCP is shifted by  $0.12 - 0.21 = -0.09$  m along the local tool  $x$  axis. Calibration is estimated from paired TCP and end-link poses and verified by world-frame forward kinematics. We additionally implement joint-specific drive overrides, joint-limit clipping, synchronized initialization of articulation states and drive targets, and collision filtering between the head and both arms.

### B.2 MOVING CAMERAS AND CALIBRATION

We add one massless head-camera link and two massless wrist-camera links to the URDF. The head camera is attached to `astribot_head_link_2` with translation  $[0.066, -0.210, 0]$  m

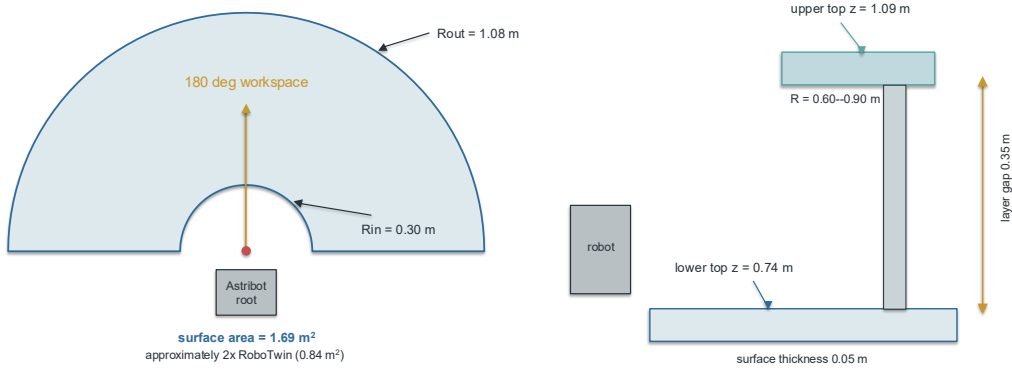


Figure 9: Fan-Shaped and Two-Level Workspaces.

and roll-pitch-yaw  $[\pi/2, 0, 0]$ . Each wrist camera is attached to its gripper base with translation  $[0, -0.05661, 0.01979]$  m and roll-pitch-yaw  $[0, -1.37081, 1.5708]$ . Although these parent-to-camera transforms are fixed, the world extrinsics depend on the robot configuration:

$${}^W T_C(t) = {}^W T_P(q_t) {}^P T_C. \quad (6)$$

Camera poses are therefore updated through forward kinematics and recorded per frame. The head camera moves only with the single pitch joint, while horizontal viewpoint changes are produced by torso yaw.

The policy head camera uses a new preset:  $512 \times 384$  pixels, a  $60^\circ$  vertical field of view, and 0.1–100 m near/far planes. Under an ideal pinhole model, this gives  $f_x = f_y = 332.55$  px and  $(c_x, c_y) = (256, 192)$  px. Optional wrist cameras use  $320 \times 240$  pixels and a  $37^\circ$  vertical field of view ( $f_x = f_y \simeq 358.64$  px). The benchmark disables wrist streams and exposes only the moving head camera to the policy.

### B.3 FAN-SHAPED AND TWO-LEVEL WORKSPACES

A rectangular table keeps most objects within a single frontal view and is therefore poorly suited to active visual search. We instead construct an annular fan-shaped table centered at the robot base. Its visual and PhysX collision surfaces are generated from the same thin box patches, with 14 radial segments, at least 24 angular segments, and a target density of 18 segments/m. The benchmark uses inner and outer radii of 0.30 and 1.08 m, a  $180^\circ$  span centered at  $90^\circ$  in world coordinates, a  $20^\circ$  placement margin at each boundary, a table thickness of 0.05 m, and a lower tabletop height of 0.74 m.

The two-level variant retains the lower fan and adds an upper annular band with inner/outer radii of 0.60/0.90 m. A 0.35 m inter-level gap places the upper surface at 1.09 m. Its angular interval may be shifted by  $20^\circ$  to induce asymmetric occlusion, and a support column is placed at the center of the upper span. Clutter placement is correspondingly changed from rectangular  $x$ - $y$  sampling to annular sampling.

### B.4 ADDITIONAL ASSETS AND TASKS

**Button asset.** The three count-and-press tasks use the articulated `005_button` asset from RM-Bench. A button is considered pressed when its joint position falls below  $-0.005$  m and reset when it rises above  $-0.001$  m.

**Task-specific assets.** For check-color tasks, we add a thin colored patch to the rear face of an otherwise gray dynamic block. The block half-size is 0.024 m; the patch half-size is  $(0.0015, 0.012, 0.012)$  m and protrudes by only 0.00005 m.

For can-inspection tasks, we retain the collision and primary visual geometry of RoboTwin’s `071_can/base3` but generate the information-bearing surface ourselves. The date task creates and caches an OBJ/MTL/PNG curved label at runtime: a  $45^\circ$  cylindrical arc of radius 0.029 m centered on

---

the rear face. Production dates range from 2023-01-01 to 2026-12-31; valid and expired samples are balanced under a 365-day shelf life. The color-check can task uses the same mechanism to generate a colored rear label.

## B.5 NEW ACTION PRIMITIVES

We retain RoboTwin’s Cartesian arm-motion and gripper primitives and add the controls required for embodiment-level motion and active perception:

- `move_joint(target_joint_pos)` commands one arm in joint space. Missing target entries are filled from the current state, excess entries are truncated, and all joints are clipped to URDF limits. TOPP time parameterization is used when available; otherwise, the fallback is 1 rad/s linear interpolation over at least 20 simulation steps.
- `move_head(delta)` and `move_head_to(target)` provide relative and absolute control of the single head-pitch joint. Likewise, `move_torso(delta)` and `move_torso_to(target)` control torso yaw. Both use joint-limit-clipped triangular or trapezoidal velocity profiles at the simulator timestep.
- `look_at_world_point_with_head` and `look_at_object` convert a world point, object center, contact point, or functional point into a viewing target. Horizontal orientation is assigned to `astribot_torso_joint_4`, while head pitch controls elevation.
- `face_world_point_with_torso`, `face_object_with_torso`, and `search_and_focus_rotate_subtask` implement discrete sector search, target centering, and visibility/memory updates.

Head or torso commands cannot be combined with arm/gripper commands within the same primitive step. The action vector is 18-dimensional, comprising 14 arm joints, two grippers, torso yaw, and head pitch.

## B.6 PER-FRAME DATA AND ONLINE ANNOTATIONS

At each simulator step, the benchmark records core observations, actions, and task metadata in HDF5. Table 11 summarizes the training-facing schema, where  $N$  is the number of registered task objects.

Each episode also includes a human-readable JSON sidecar containing the task instruction, object key/index/name mappings, subtask definitions, transition log, and final discovery state. All annotations are generated online from simulator and task state rather than labeled after collection.

Table 11: Per-frame trajectory fields recorded by the benchmark configuration.

| Group                   | Field                   | Shape / type         | Meaning                                            |
|-------------------------|-------------------------|----------------------|----------------------------------------------------|
| observation/camera_head | rgb                     | encoded RGB          | moving head-camera image                           |
|                         | intrinsic_cv            | $3 \times 3$ , float | per-frame camera intrinsics                        |
|                         | extrinsic_cv            | $4 \times 4$ , float | world-to-camera transform, CV convention           |
|                         | cam2world_gl            | $4 \times 4$ , float | camera-to-world transform, OpenGL convention       |
| endpose                 | left/right_endpose      | 7, float             | TCP position and quaternion                        |
| joint_action            | left/right_gripper      | scalar, float        | normalized gripper state                           |
|                         | left/right_arm          | 7, float             | arm joint state                                    |
|                         | left/right_gripper      | scalar, float        | normalized gripper state                           |
|                         | head/torso              | 1 / 1, float         | head pitch and torso-yaw state                     |
| Top level               | vector                  | 18, float            | concatenated simulator state                       |
|                         | subtask                 | int32                | current semantic subtask ID                        |
|                         | stage                   | int8                 | 1: search; 2: anchor; 3: action                    |
|                         | subtask_instruction_idx | int32                | instruction lookup ID                              |
|                         | focus_object_idx        | int16                | focused object; $-1$ if absent                     |
|                         | focus_object_visible    | int8                 | whether the focused object is currently visible    |
|                         | info_complete           | int8                 | whether sufficient information has been acquired   |
|                         | camera_mode             | int8                 | active-camera state-machine mode                   |
|                         | camera_target_theta     | float32              | planned viewing angle; NaN if inactive             |
|                         | waist_heading_deg       | float32              | current horizontal viewing direction               |
|                         | visible_object_mask     | $N$ , int8           | objects visible in the current frame               |
|                         | discovered_object_mask  | $N$ , int8           | objects observed at least once                     |
|                         | search_target_mask      | $N$ , int8           | objects required for information acquisition       |
|                         | action_target_mask      | $N$ , int8           | objects involved in the physical action            |
|                         | carried_object_mask     | $N$ , int8           | objects currently carried                          |
|                         | target_uv_norm          | 2, float32           | normalized target coordinate; $(-1, -1)$ if absent |
| Post-process            | subtask_keyframe        | int8                 | semantic-boundary indicator                        |
|                         | motion_keyframe         | int8                 | causal motion-boundary indicator                   |

## C ActiveArena-BENCH DESIGN AND DATA PROTOCOL

This section describes the design principles, evaluation domains, data protocol, and history-keyframe definitions of our information-gathering manipulation benchmark. Unlike conventional tasks that can be solved from the current observation alone, these tasks require the robot to actively acquire missing evidence, retain it after it leaves the field of view, and use it to guide subsequent decisions.

### C.1 TASK DESIGN PRINCIPLES AND OBJECT DISTRIBUTION

All tasks satisfy four design principles. **(1) Initial Information Insufficiency.** The correct manipulation cannot be determined from the initial observation alone. **(2) Information Accessibility.** The missing task-relevant information can be acquired through spatial search or physical interaction. **(3) Historical Dependence.** Evidence acquired from previous observations must remain relevant even after the corresponding objects or regions leave the current field of view. **(4) Closed-Loop Decision Making.** Newly acquired evidence must influence subsequent localization or manipulation decisions. After each action, the robot must reassess whether the accumulated evidence is sufficient and continue acquiring information when necessary.

These principles jointly constrain scene generation and object placement. Figure 9 illustrates the shared distribution template. Single-level tasks use a  $180^\circ$  annular sector centered at the robot, with inner and outer radii of 0.3 m and 1.08 m, respectively. Task objects and information-bearing regions are placed at least  $20^\circ$  from either angular boundary. In two-level tasks, the upper sector may be offset by  $20^\circ$  relative to the lower sector, forcing the robot to actively switch viewing regions. An object is considered visible when at least 40% of its projected bounding-box area lies within the head-camera image and remains unoccluded by other scene elements, as determined through depth-aware visibility checking. Each scene additionally contains a randomized number of static distractors on the lower level.



## C.2 BENCHMARK TASK INVENTORY AND SUBTASK SEMANTICS

Table 12 specifies the benchmark at two complementary semantic levels.

Table 12: Benchmark tasks, instructions, and subtask definitions.

| Task                             | Instruction                                                             | Subtasks                                                                                                                                                                                                                                                                                                                                                      |
|----------------------------------|-------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| beat_block_hammer_rotate_view    | Grab hammer using the left arm and hit the block.                       | <b>T1</b> pick_hammer: locate and grasp the hammer.<br><b>T2</b> hammer_block: locate the block and strike it with the carried hammer.                                                                                                                                                                                                                        |
| blocks_ranking_rgb_fan_double    | Arrange the red, green, and blue blocks from left to right.             | <b>T1</b> pick_green_block: locate and grasp the green block.<br><b>T2</b> place_green_block_right_of_red: place the green block to the right of the red block.<br><b>T3</b> pick_blue_block: locate and grasp the blue block.<br><b>T4</b> place_blue_block_right_of_green: place the blue block to the right of the green block.                            |
| blocks_ranking_rgb_rotate_view   | Arrange the red, green, and blue blocks from left to right.             | <b>T1</b> pick_green_block: locate and grasp the green block.<br><b>T2</b> place_green_block_right_of_red: place the green block to the right of the red block.<br><b>T3</b> pick_blue_block: locate and grasp the blue block.<br><b>T4</b> place_blue_block_right_of_green: place the blue block to the right of the green block.                            |
| blocks_ranking_size_fan_double   | Arrange the large, medium, and small blocks from left to right.         | <b>T1</b> pick_medium_block: locate and grasp the medium block.<br><b>T2</b> place_medium_block_right_of_large: place the medium block to the right of the large block.<br><b>T3</b> pick_small_block: locate and grasp the small block.<br><b>T4</b> place_small_block_right_of_medium: place the small block to the right of the medium block.              |
| blocks_ranking_size_rotate_view  | Arrange the large, medium, and small blocks from left to right.         | <b>T1</b> pick_medium_block: locate and grasp the medium block.<br><b>T2</b> place_medium_block_right_of_large: place the medium block to the right of the large block.<br><b>T3</b> pick_small_block: locate and grasp the small block.<br><b>T4</b> place_small_block_right_of_medium: relocate the medium block and place the small block to its right.    |
| check_block_color                | Inspect the gray block’s hidden color and place it on the matching pad. | <b>T1</b> pick_target_block: locate and pick up the gray target block.<br><b>T2</b> inspect_target_block_backside: reveal and identify the block’s backside color.<br><b>T3</b> restore_target_block_pose: restore the block to a placement-ready pose.<br><b>T4</b> place_target_block_on_matching_pad: localize the matching pad and place the block on it. |
| check_cola_color                 | Inspect the can’s hidden color and place it in the matching area.       | <b>T1</b> pick_cola_can: pick up the can.<br><b>T2</b> inspect_cola_backside_color: expose and read the backside color label.<br><b>T3</b> restore_cola_after_inspection: restore the can to a placement-ready pose.<br><b>T4</b> place_can_on_matching_color_area: select the matching area and place the can.                                               |
| check_cola_date                  | Inspect the production date and sort the can by expiry status.          | <b>T1</b> pick_cola_can: pick up the can.<br><b>T2</b> inspect_cola_backside_date: expose and read the production-date label.<br><b>T3</b> restore_cola_after_inspection: restore the can to a placement-ready pose.<br><b>T4</b> place_cola_by_expiry: infer expiry status, localize the corresponding area, and place the can.                              |
| click_bell_rotate_view           | Press the bell’s top center using the left arm.                         | <b>T1</b> click_bell: locate the bell and press its top center.                                                                                                                                                                                                                                                                                               |
| count_color_kinds_press_button   | Count the distinct block colors and press the matching button.          | <b>T1</b> count_unique_block_colors: scan all blocks and infer the number of distinct colors.<br><b>T2</b> press_matching_number_button: localize and press the button corresponding to the inferred count.                                                                                                                                                   |
| count_random_Object_press_button | Count the instructed objects and press the matching button.             | <b>T1</b> count_random_target_objects: scan the scene and count instances of the instructed category.<br><b>T2</b> press_matching_number_button: localize and press the button corresponding to the inferred count.                                                                                                                                           |
| count_target_press_button        | Count the target blocks and press the matching button.                  | <b>T1</b> count_green_target_blocks: scan targets and distractors and infer the green-block count.<br><b>T2</b> press_matching_number_button: localize and press the button corresponding to the inferred count.                                                                                                                                              |

Continued on next page

Table 12 (continued)

| Task                                  | Instruction                                                         | Subtasks                                                                                                                                                                                                                                                                                                                                  |
|---------------------------------------|---------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| match_backside_two_blocks             | Inspect two gray blocks and place each on its matching pad.         | <b>T1/T4</b> pick_a_grey_block: locate and pick the next gray block.<br><b>T2/T5</b> inspect_a_grey_block_backside_color: expose and infer the carried block's backside color.<br><b>T3/T6</b> place_a_grey_block_on_matching_pad: localize the matching pad and place the block.                                                         |
| move_pillbottle_pad_rotate_view       | Move the pill bottle onto the pad.                                  | <b>T1</b> pick_pillbottle: locate and pick up the pill bottle.<br><b>T2</b> place_pillbottle_on_pad: localize the pad and place the carried bottle on it.                                                                                                                                                                                 |
| move_stapler_pad_rotate_view          | Move the stapler onto the black mat.                                | <b>T1</b> pick_stapler: locate and pick up the stapler.<br><b>T2</b> place_stapler_on_pad: localize the instructed mat and place the stapler on it.                                                                                                                                                                                       |
| place_a2b_left_rotate_view            | Place the Rubik's cube to the left of the wooden block.             | <b>T1</b> pick_object_A: locate and pick up source object A.<br><b>T2</b> place_A_left_of_B: localize reference object B and place A to its left.                                                                                                                                                                                         |
| place_a2b_right_rotate_view           | Place the bell to the right of the coffee box.                      | <b>T1</b> pick_object_A: locate and pick up source object A.<br><b>T2</b> place_A_right_of_B: localize reference object B and place A to its right.                                                                                                                                                                                       |
| place_cans_plasticbox_rotate_view     | Place both cans into the plastic box.                               | <b>T1</b> pick_first_can: locate and pick up the first can.<br><b>T2</b> place_first_can_in_box: localize the box and place the first can inside.<br><b>T3</b> pick_second_can: locate and pick up the second can.<br><b>T4</b> place_second_can_in_box: relocate the box and place the second can inside.                                |
| place_container_plate_rotate_view     | Place the container on the plate.                                   | <b>T1</b> pick_container: locate and grasp the container.<br><b>T2</b> place_container_on_plate: localize the plate and place the carried container on it.                                                                                                                                                                                |
| place_empty_cup_rotate_view           | Place the empty cup on the coaster.                                 | <b>T1</b> pick_cup: locate and grasp the cup.<br><b>T2</b> place_cup_on_coaster: localize the coaster and place the carried cup on it.                                                                                                                                                                                                    |
| place_fan_rotate_view                 | Place the fan on the silver mat facing the robot.                   | <b>T1</b> pick_fan: locate and grasp the fan.<br><b>T2</b> place_fan_on_pad: localize the instructed mat and place the fan with the required orientation.                                                                                                                                                                                 |
| place_mouse_pad_rotate_view           | Place the mouse on the cyan mat.                                    | <b>T1</b> pick_mouse: locate and grasp the mouse.<br><b>T2</b> place_mouse_on_pad: localize the instructed pad and place the mouse on it.                                                                                                                                                                                                 |
| place_object_basket_fan_double        | Place the target object into the bread basket.                      | <b>T1</b> pick_object: locate and grasp the episode-specific target object.<br><b>T2</b> place_object_into_basket: localize the bread basket and place the carried object inside.                                                                                                                                                         |
| place_object_scale_rotate_view        | Place the target object on the electronic scale.                    | <b>T1</b> pick_object: locate and grasp the episode-specific target object.<br><b>T2</b> place_object_on_scale: localize the scale and place the carried object on it.                                                                                                                                                                    |
| place_object_stand_rotate_view        | Place the Rubik's cube on the display stand.                        | <b>T1</b> pick_object: locate and grasp the target object.<br><b>T2</b> place_object_on_stand: localize the display stand and place the carried object on it.                                                                                                                                                                             |
| place_shoe_rotate_view                | Place the shoe on the mat.                                          | <b>T1</b> pick_shoe: locate and grasp the shoe.<br><b>T2</b> place_shoe_on_mat: localize the mat and place the carried shoe on it.                                                                                                                                                                                                        |
| press_stapler_rotate_view             | Firmly press the stapler.                                           | <b>T1</b> _press_stapler: localize and press the stapler.                                                                                                                                                                                                                                                                                 |
| put_block_on_upper_easy               | Place the block into the upper-level plate.                         | <b>T1</b> pick_a_block_0: locate and pick up the block.<br><b>T2</b> place_a_block_0_into_plate: localize the upper-level plate and place the block inside.                                                                                                                                                                               |
| put_block_on_upper_hard               | Place all blocks into the plate.                                    | <b>T1</b> pick_a_block_0: select and pick up the first remaining block.<br><b>T2</b> place_a_block_0_into_plate: localize the plate and place the first block inside.<br><b>T3</b> pick_a_block_1: locate and pick up the remaining block.<br><b>T4</b> place_a_block_1_into_plate: relocate the plate and place the second block inside. |
| rank_backside_rgb_blocks              | Inspect three gray blocks and arrange them in red-green-blue order. | <b>T1/T4/T7</b> pick_a_grey_block: locate and pick the next gray block.<br><b>T2/T5/T8</b> inspect_a_grey_block_backside_color: expose and infer its hidden backside color.<br><b>T3/T6/T9</b> place_a_grey_block_on_rgb_rank_pad: localize the color-implied rank pad and place the block.                                               |
| shake_bottle_horizontally_rotate_view | Pick up the bottle and shake it horizontally.                       | <b>T1</b> _shake_bottle_horizontally: locate and grasp the bottle, then execute horizontal shaking.                                                                                                                                                                                                                                       |
| shake_bottle_rotate_view              | Pick up the bottle and shake it.                                    | <b>T1</b> _shake_bottle: locate and grasp the bottle, then execute the shaking motion.                                                                                                                                                                                                                                                    |

Continued on next page

Table 12 (continued)

| Task                             | Instruction                                  | Subtasks                                                                                                                                                       |
|----------------------------------|----------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| stack_blocks_two_<br>rotate_view | Stack the green block on the red block.      | <b>T1</b> pick_green_block: locate and grasp the green block.<br><b>T2</b> stack_green_on_red: localize the red block and stack the carried green block on it. |
| stamp_seal_rotate_<br>view       | Move the seal to the beige pad and stamp it. | <b>T1</b> pick_seal: locate and grasp the seal.<br><b>T2</b> stamp_target_pad: localize the instructed pad and execute the stamping action.                    |
| turn_switch_rotate_<br>view      | Locate and activate the switch.              | <b>T1</b> _turn_switch: localize and trigger the switch.                                                                                                       |

### C.3 ID/OOD EVALUATION PROTOCOL

Both the in-distribution (ID) and out-of-distribution (OOD) settings randomize task-object poses, object attributes, and clutter layouts while sharing the same workspace geometry, camera configuration, and success criteria.

The OOD setting introduces stronger visual distribution shifts through randomized backgrounds, tabletop textures, illumination, and object distributions, with clutter potentially including target objects from other tasks. Experimental results reveal a clear ID/OOD performance gap, indicating that this configuration provides a sufficiently challenging test of model generalization.

### C.4 HISTORY-KEYFRAME DEFINITIONS

Because there is no standard representation for robot visual history, we compare three complementary keyframe definitions: semantic Subtask Keyframes, motion-driven Motion Keyframes, and action-aligned Chunk Keyframes. For a trajectory of length  $T$ ,

$$\tau = \{(o_t, a_t, s_t, x_t)\}_{t=0}^{T-1}, \quad (7)$$

where  $o_t$  is the visual observation,  $a_t$  is the action,  $s_t$  is the discrete subtask label, and  $x_t$  is the robot state. For each keyframe type  $z \in \text{sub, mot, chunk}$ , let  $\mathcal{K}^z$  denote the retained frame set and  $k_t^z = \mathbb{1}[t \in \mathcal{K}^z]$  its binary indicator.

**Subtask Keyframes.** The first trajectory frame and the first frame following each change in the discrete subtask label are retained:

$$\mathcal{K}^{\text{sub}} = \{0\} \cup \{t \mid s_t \neq s_{t-1}, 1 \leq t < T\}. \quad (8)$$

For example, when a task transitions from “search for and grasp the hammer” to “locate and strike the target block,” the first frame of the latter subtask is retained. This definition captures high-level task progress and semantic phase transitions.

**Motion Keyframes.** Motion Keyframes retain meaningful motion-state boundaries rather than uniformly sampled frames. Let  $\mathcal{E}(x_{\leq t})$  denote the set of events causally confirmed by time  $t$ . The default events include the gripper entering or leaving a stable open or closed state, as well as the onset or termination of head-pitch or torso-yaw rotation. The retained set is

$$\mathcal{K}^{\text{mot}} = \text{Dedup}_\delta(\{0\} \cup \{t \mid \mathcal{E}(x_{\leq t}) \neq \emptyset\}), \quad \delta = 5. \quad (9)$$

A gripper is considered closed at values no greater than 0.2 and open at values no less than 0.8; the state must persist for two frames before confirmation. Head pitch or torso yaw is considered rotating when its absolute frame-to-frame change exceeds 0.005 rad. Rotation must persist for four frames, and interruptions of at most five frames are merged.

We also provide end-effector-motion event detector. For arm  $a \in \{\text{L}, \text{R}\}$ , let  $\mathbf{p}_t^a \in \mathbb{R}^3$  denote the end-effector position. Its Cartesian velocity and speed are defined as

$$\mathbf{v}_t^a = \frac{\mathbf{p}_t^a - \mathbf{p}_{t-1}^a}{\Delta t}, \quad s_t^a = \|\mathbf{v}_t^a\|_2. \quad (10)$$

We detect two types of end-effector events. A turning event is triggered when the angle between two consecutive velocity vectors exceeds  $40^\circ$ :

$$\alpha_t^a = \arccos\left(\frac{\langle \mathbf{v}_{t-1}^a, \mathbf{v}_t^a \rangle}{s_{t-1}^a s_t^a}\right) > 40^\circ, \quad (11)$$

where both  $s_{t-1}^a$  and  $s_t^a$  must be greater than  $\epsilon_{\text{stop}} = 0.001$ .

A stopping event is triggered when the end effector changes from moving to nearly stationary:

$$s_{t-1}^a > \epsilon_{\text{stop}}, \quad s_t^a \leq \epsilon_{\text{stop}}, \quad \epsilon_{\text{stop}} = 0.001. \quad (12)$$

The resulting eef event set is

$$\mathcal{E}^{\text{eef}}(x_{\leq t}) = \{(a, \text{turn}, t) \mid \alpha_t^a > 40^\circ\} \cup \{(a, \text{stop}, t) \mid s_{t-1}^a > 0.001, s_t^a \leq 0.001\}. \quad (13)$$

**Chunk Keyframes.** Chunk Keyframes provide a uniform temporal baseline aligned with policy action chunks rather than motion changes. With a fixed chunk length of  $C = 16$ ,

$$\mathcal{K}^{\text{chunk}} = nC \mid n \in \mathbb{N}_0, nC < T, \quad k_t^{\text{chunk}} = \mathbb{I}[t \bmod 16 = 0]. \quad (14)$$

Each Chunk Keyframe corresponds to a timestep at which the policy receives a new observation, predicts the next action chunk, and replans.

**Unified Input Construction.** For keyframe type  $z$ , the visual memory at time  $t$  consists of the most recent  $H$  retained observations strictly preceding the current frame:

$$M_t^z = \text{Recent}_H(\{o_i \mid i < t, i \in \mathcal{K}^z\}). \quad a_t \sim \pi_\theta(a_t \mid M_t^z, o_t, g), \quad (15)$$

where  $g$  denotes the task instruction.

$$X_t^z = \text{Recent}_H(\{x_i \mid i < t, i \in \mathcal{K}^z\}) \parallel x_t, \quad (16)$$

The state-conditioned policy is

$$a_t \sim \pi_\theta(a_t \mid M_t^z, o_t, g, X_t^z). \quad (17)$$

We use  $H = 12$ , corresponding to at most 12 historical keyframes and one current frame.

## D MODEL TRAINING AND EVALUATION DETAILS

### D.1 EVALUATION RESOURCES

**Deterministic seed generation and storage.** The evaluation environment is configured as follows:

- Ubuntu 22.04;
- Python 3.10;
- CUDA 12.1;
- PyTorch 2.4.1.

Before evaluating the models, we perform a simulator-only run without loading any policy model to collect the random seeds for all evaluation tasks. The resulting seed manifest is shared across all models, ensuring that every model is evaluated using the same set of random seeds.

**RTX-4090 reproduction.** Table 13 reports the reference configuration for evaluating an OFT policy with a 12-frame memory and 16-step action chunks on a single NVIDIA RTX 4090. For  $N$  NVIDIA RTX 4090 GPUs, the same seed manifest can be evaluated using  $N$  independent evaluation workers in parallel.

Table 13: Reference evaluation profile on a single NVIDIA RTX 4090.

| Configuration           | Value                  |
|-------------------------|------------------------|
| GPU                     | NVIDIA RTX 4090, 24 GB |
| Peak GPU memory         | 21.8 GB                |
| Peak host memory        | 14.6 GB                |
| Mean throughput         | 93.3 episodes/hour     |
| ID episodes             | 1750                   |
| OOD episodes            | 1750                   |
| Complete ID+OOD runtime | 37.50 hours            |

## D.2 EXTERNAL BASELINE TRAINING SETTINGS

$\pi_{0.5}$ . We fine-tune  $\pi_{0.5}$  using the current head-camera image and its aligned 18-dimensional proprioceptive state. Wrist-camera input slots are masked throughout training, while all other optimization settings follow the default OPENPI configuration. At evaluation time, each action chunk is generated using five flow-matching sampling steps.

**HiF-VLA.** Following the original configuration, we fine-tune OpenVLA-7B using LoRA with rank 32 and zero dropout. Each training sample contains the current head-camera image, its aligned 18-dimensional proprioceptive state, and eight motion-vector entries spanning timesteps  $t - 7$  through  $t$ . Images are resized to  $224 \times 224$  and normalized for the fused DINOv2/SigLIP visual backbone. Under quantile normalization, the model directly regresses an eight-step action chunk, with each step represented as an 18-dimensional absolute control vector. At inference time, the corresponding inverse normalization is applied, and the two gripper channels are clipped to  $[0, 1]$ .

**MemER.** The MemER high-level policy is fine-tuned from Qwen3-VL-4B-Instruct, while  $\pi_{0.5}$  serves as the low-level controller. We otherwise retain the original training pipeline. The high-level policy receives up to eight selected memory frames, followed by up to eight recent head-camera frames sampled at a stride of five. It jointly predicts the current subtask and the location of the next keyframe.

**Fast-WAM.** We adapt Fast-WAM to a single head-camera stream and initialize it from the Wan2.2-5B video DiT, T5 text encoder, and video VAE. Each training sample contains the current head-camera image, its aligned 18-dimensional proprioceptive state, future head-camera frames, and a 32-step action chunk of 18-dimensional absolute controls. The current frame serves as a clean visual anchor, whereas future frames are used only for the auxiliary video flow-matching objective; the action branch is causally prevented from accessing future visual tokens. Training follows the original joint action–video flow-matching objective. At inference time, the future-video branch is removed, and the action chunk is generated from the current observation using ten flow-matching steps.

**MemoryVLA.** We retain the original Prismatic-7B vision-language backbone, perceptual–cognitive memory bank, and diffusion action expert. The head-camera image is encoded by the fused DINOv2/SigLIP visual backbone, while the aligned 18-dimensional proprioceptive state is linearly projected as an additional condition for the action expert. The memory bank stores at most 16 perceptual–cognitive entries and is updated online through historical retrieval, gated fusion, and adjacent-entry merging. We replace the original action head with an 18-dimensional absolute-control head that predicts a 16-step action chunk. All other training settings remain unchanged. Inference uses ten DDIM sampling steps with a classifier-free guidance weight of 1.5.

**SaPaVe.** We retain the Eagle-2 camera adapter, MapAnything spatial encoder, and decoupled diffusion action heads. The original camera-action branch is adapted to a two-dimensional active-view controller that predicts torso yaw and the single-DoF head pitch, while the manipulation branch predicts 14 arm-joint commands and two gripper commands. Training follows the original two-stage procedure. In the first stage, only the LoRA camera adapter and active-view decoder are optimized on search-and-focus segments, with the spatial encoder and manipulation decoder frozen. In the second stage, the camera adapter is frozen and both action branches are jointly trained on complete demonstrations. Their outputs are concatenated along the action dimension to form an 18-dimensional action chunk.

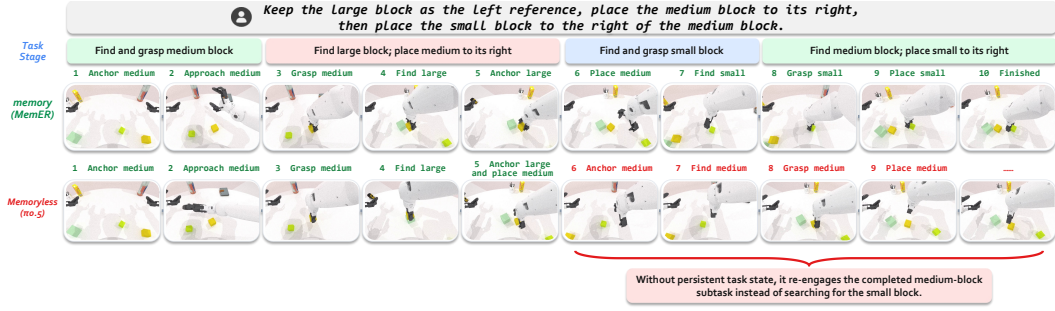


Figure 10: Paired rollouts of the memory-aware policy MemER (top) and the memoryless policy  $\pi_{0.5}$  (bottom) on the block-arrangement task.



Figure 11: Paired rollouts of the memory-aware policy MemER (top) and the memoryless policy  $\pi_{0.5}$  (bottom) on the hidden-color sorting task.

### D.3 VISUALIZATION OF MEMORYLESS-MODEL FAILURES

We use  $\pi_{0.5}$  as a representative memoryless policy and compare it with MemER to analyze failures caused by the lack of persistent historical information. Figures 10 and 11 present paired rollouts: both policies receive the same instruction and start from scene configurations generated with the same random seed. In these tasks, the next action depends not only on the current visual observation, but also on completed subgoals and evidence acquired from previous viewpoints. MemER is able to preserve this execution context across manipulation steps and viewpoint changes.

In the block-arrangement task, MemER remembers that the medium block has already been placed relative to the large block and therefore shifts its search toward the small block. In contrast,  $\pi_{0.5}$  re-enters the visually plausible medium-block subtask and repeatedly manipulates the same object. In the hidden-color sorting task, MemER retains the backside color observed during inspection after the object is reoriented and uses this information to select the matching target area. By contrast,  $\pi_{0.5}$  fails to retain the inspection result as evidence for subsequent decisions, repeatedly returning to the inspection stage without completing the placement. These failures show that losing execution history and previously acquired evidence can reduce long-horizon manipulation to locally repetitive behavior.

### D.4 INTERNAL BASELINE DETAILS

This section provides the implementation and optimization details for the 13 internal baseline policies. All variants use the same training set and the same head-camera observation stream. The comparison changes only the action-generation mechanism, language-level auxiliary supervision, temporal keyframe selection, memory length, or proprioceptive-state conditioning. Table 15 gives the complete run manifest, while Tables 14 and 16 report the shared and head-specific hyperparameters, respectively.



#### D.4.1 SHARED TRAINING PROTOCOL

A run took approximately 50 hours, corresponding to approximately 400 A800 GPU-hours per model. Runs with auxiliary vision–language co-training draw one action batch and one language-supervision batch at each step.

All parameters are optimized and we use AdamW with  $\beta_1 = 0.9$ ,  $\beta_2 = 0.95$ ,  $\epsilon = 10^{-8}$ , and weight decay  $10^{-8}$ . The Qwen backbone and its interface use a peak learning rate of  $10^{-5}$ , whereas the state encoder and continuous action heads use  $10^{-4}$ .

Table 14: Shared optimization and data hyperparameters for all inner baselines.

| Setting                          | Value                                               |
|----------------------------------|-----------------------------------------------------|
| Hardware                         | 8× NVIDIA A800, single node                         |
| Training budget                  | 150,000 optimizer steps; approximately 50 h per run |
| Precision / distribution         | bfloat16; DeepSpeed ZeRO-2; no CPU offload          |
| Batch size                       | 7 per GPU                                           |
| Gradient accumulation            | 1                                                   |
| Optimizer                        | AdamW, $\beta = (0.9, 0.95)$ , $\epsilon = 10^{-8}$ |
| Weight decay / gradient clipping | $10^{-8}$ / global norm 1.0                         |
| Backbone learning rate           | $10^{-5}$                                           |
| State/action-head learning rate  | $10^{-4}$                                           |
| Schedule                         | 5,000-step linear warmup, then cosine decay         |
| Minimum learning rate            | $5 \times 10^{-7}$                                  |
| Random seed                      | 42                                                  |
| Action representation            | 18-D absolute control; 16-step action chunk         |

**Inputs and targets.** Each sample contains the current head-camera image and, when enabled, up to  $K \in \{6, 12\}$  historical images. Consequently, a 12-frame memory variant receives at most 13 images (12 history frames plus the current frame). For *action* history, frames are sampled backward at a stride of 16 environment steps. The no-history variant uses only the current image.

The action target is a 16-step chunk of 18-D absolute controls. State-enabled models receive an aligned 18-D proprioceptive vector for every input image. A two-layer MLP with hidden width 1,024 projects each vector into one soft state token; learned frame embeddings preserve temporal identity.

**Language supervision.** Every action branch is prompted with the full task instruction. The second field in an OFT run name controls only the auxiliary language branch: `no` disables this branch, `instruction` supervises the `<think>` target with the full task instruction, and `subtask` supervises it with the annotated current subtask. For co-trained OFT and GR00T runs, the total loss is

$$\mathcal{L} = \mathcal{L}_{\text{action}} + 0.1 \mathcal{L}_{\text{VLM}}. \quad (18)$$

The FAST policy places the same subtask-level reasoning target before the discrete action tokens in a single autoregressive response and is therefore trained with one token-level cross-entropy objective rather than the separate loss in Eq. 18.

#### D.4.2 ActiveArena-VLA MODEL VARIANTS

Run names follow `architecture_think_history_memory_state`. Here `ws` and `wos` denote with and without proprioceptive state. The memory field is the maximum number of *historical* frames and excludes the current frame. The 13 configurations in Table 15 form controlled comparisons along these axes.

#### D.4.3 ActiveArena-FAST

`fast_subtask_action_12_ws` uses the same 12-frame action history and state conditioning, but discretizes the  $16 \times 18$  action chunk with the FAST tokenizer and predicts it autoregressively. We initialize from Qwen3-VL-2B-Instruct-Action, which extends Qwen3-VL-2B-Instruct with 2,048

Table 15: Complete internal-baseline manifest. “Think” is the auxiliary language target,  $K$  is the maximum number of history frames, and “State” indicates per-image proprioceptive soft-token conditioning.

| Run                          | Head  | Think       | History | $K$ | State | Action objective    |
|------------------------------|-------|-------------|---------|-----|-------|---------------------|
| oft_instruction_action_12_ws | OFT   | instruction | action  | 12  | yes   | continuous L1       |
| oft_no_action_12_ws          | OFT   | none        | action  | 12  | yes   | continuous L1       |
| oft_subtask_no_0_ws          | OFT   | subtask     | none    | 0   | yes   | continuous L1       |
| oft_subtask_action_12_wos    | OFT   | subtask     | action  | 12  | no    | continuous L1       |
| oft_subtask_action_12_ws     | OFT   | subtask     | action  | 12  | yes   | continuous L1       |
| oft_subtask_action_6_ws      | OFT   | subtask     | action  | 6   | yes   | continuous L1       |
| oft_subtask_motion_12_ws     | OFT   | subtask     | motion  | 12  | yes   | continuous L1       |
| oft_subtask_motion_6_ws      | OFT   | subtask     | motion  | 6   | yes   | continuous L1       |
| oft_subtask_subtask_12_ws    | OFT   | subtask     | subtask | 12  | yes   | continuous L1       |
| oft_subtask_subtask_6_ws     | OFT   | subtask     | subtask | 6   | yes   | continuous L1       |
| gr00t_subtask_action_12_ws   | GR00T | subtask     | action  | 12  | yes   | flow matching       |
| fast_subtask_action_12_ws    | FAST  | subtask     | action  | 12  | yes   | token cross-entropy |
| oft_subtask_planned_12_ws    | OFT   | subtask     | planned | 12  | yes   | continuous L1       |

pre-initialized robot-action tokens. The target sequence concatenates a subtask-supervised `<think>` span and the FAST codes inside an `<action>` span, and the complete response is optimized by token cross-entropy.

Table 16: Action-head-specific settings. Shared entries such as the 18-D action space, 16-step horizon, optimizer, and training budget are reported in Table 14.

| Family | Initialization              | Head-specific configuration                                                                          |
|--------|-----------------------------|------------------------------------------------------------------------------------------------------|
| OFT    | Qwen3-VL-2B-Instruct        | 16 action-query positions; 2,048-D input, 4,096-D hidden MLPResNet; 2 residual blocks; L1 regression |
| GR00T  | Qwen3-VL-2B-Instruct        | DiT-B, width 1,024, 16 layers; 32 vision tokens; 8 training noise repeats; 4 inference steps         |
| FAST   | Qwen3-VL-2B-Instruct-Action | FAST tokenizer; 2,048 action tokens; autoregressive cross-entropy                                    |

#### D.4.4 ActiveArena-PLAN ARCHITECTURE

We design *ActiveArena-Plan*: A planner observes the global task instruction, a bounded history, and the current image. It autoregressively predicts the current subtask and indices of frames worth retaining. A separate VLA executor is conditioned on that subtask, the retrieved memory frames, the current image, and frame-aligned robot state, and predicts only the continuous action chunk. Thus, the planner decides *what stage the task is in* and *what to remember*, while the VLA decides *how to execute the current stage*.

**Training Details.** For a training sample at current trajectory frame  $x$ , let the stride be  $s = 16$ . We first construct the full stride-aligned sequence

$$\mathcal{S}_\infty(x) = \text{sort}\{x - js \mid j \in \{0, 1, 2, \dots\}, x - js \geq 0\}. \quad (19)$$

The planner keeps only the most recent  $K + 1$  elements, where  $K = 12$  is the maximum number of *history* frames and the final element is always  $x$ . It therefore sees at most 13 images. Near the beginning of a trajectory the sequence is naturally shorter; sampling stops before an index would become negative. Images are ordered from earliest to latest and are followed by the global task prompt. Under the Qwen chat template, this is serialized conceptually as  $N$  consecutive image items followed by the prompt text; the `<image>` markers are not inserted as ordinary hand-written text tokens:

```
Your task is: "{instruction}" The input images are ordered
from earliest to latest, and the last image is the current
view. Please think about the current subtask and which
```

---

```

frame(s) among the input images are worth remembering. Your
response should be in the format of: <think>...</think>
<subtask>...</subtask><retrieval>...
</retrieval>.

```

The planner receives images and language but no robot state. Its target has exactly one `<subtask>` span. We use FlashAttention-2, a maximum language-model sequence length of 4,096 tokens, and dynamic image resizing between 784 and 50,176 pixels:

```

<think>Frames: {num_frames} total ({num_history} history +
current). Now the
subtask is "{subtask instruction}"</think><subtask>{subtask
instruction}</subtask>
<retrieval>[keyframe indices]</retrieval>

```

Only the current subtask instruction is passed as language conditioning to the executor; the `<think>` span provides structured autoregressive supervision, and `<retrieval>` controls visual memory.

**Retrieval labels.** Let  $q$  be a previously observed subtask keyframe and let  $f_i$  denote the frame at zero-based position  $i$  in the planner input. Position  $i$  is a positive retrieval target exactly when

$$q \leq f_i < q + s. \quad (20)$$

Equivalently,  $f_i$  must lie in the half-open window  $[q, q + 16)$ . We emit the unique matching input positions in chronological order. Frames outside all such windows are not retrieval targets.

For example, when  $x = 89$ , Eq. 19 gives the input frame indices

$$[9, 25, 41, 57, 73, 89]. \quad (21)$$

Suppose the observed subtask keyframes are  $q = 0$  and  $q = 45$ . Frame 9 lies in  $[0, 16)$  and frame 57 lies in  $[45, 61)$ , so their positions in the six-image input are 0 and 3. The complete target is

```

<think>Frames: 6 total (5 history + current). Now the
subtask is
"{subtask instruction}"</think>
<subtask>{subtask instruction}</subtask><retrieval>[0,
3]</retrieval>.

```

The textual placeholders are filled with the subtask annotation for frame 89.

**Executor Memory and Action Target.** The executor uses the same stride-16 sequence but does *not* apply the planner’s  $K = 12$  history cap. During supervised VLA data construction, we apply Eq. 20 to all subtask keyframes and all frames in  $\mathcal{S}_\infty(x)$ . The selected memory frames are arranged chronologically and followed by the current frame  $x$ ; if  $x$  is already the latest selected memory frame, it is not duplicated. Each image is paired with its aligned 18-D state vector.

The executor is prompted with the predicted current subtask rather than the global instruction:

```

Your task is: "{subtask instruction}" The input images are
ordered from earliest to latest, and the last image is the
current view. Please think about the next
action.

```

It uses the same OFT configuration described.

**Resource Efficiency.** Training `oft_subtask_planned_12_ws` reduces peak GPU memory usage by 46% compared with the best dense-history model, `oft_subtask_action_12_ws`, decreasing from 78.9 GB to 42.6 GB.

## E DETAILS OF REAL-WORLD EXPERIMENTS

### E.1 HARDWARE AND SOFTWARE SETUP

**Robot platform and observations.** All real-world experiments are conducted on an Astribot S1. The platform is equipped with two 7-DoF arms, two grippers, an actuated torso, and an actuated head.

The policy receives egocentric RGB observations from the head-mounted camera, together with the language instruction and robot proprioceptive state.

Consistent with the simulation setup, the policy predicts an 18-D action:

$$\mathbf{a}_t = [\mathbf{a}_t^L, g_t^L, \mathbf{a}_t^R, g_t^R, a_t^{\text{torso}}, a_t^{\text{head}}], \quad (22)$$

which jointly controls the two arms, two grippers, torso, and head.

**Calibration and execution.** Before each evaluation session, we verify the camera intrinsics, camera-to-head extrinsics, robot kinematics, and the zero configurations of the head and torso. Policy inference is performed on a dedicated GPU workstation connected to the robot controller through a wired local network. For every episode, we record RGB observations, proprioceptive states, predicted actions, controller feedback, and timestamps.

**Capability taxonomy.** We use four operational labels to characterize the active perception capabilities required by the real-world tasks.

Table 17: Operational definitions of the active perception capabilities used in the real-world evaluation.

| Symbol | Capability | Operational meaning                                                                                                                                                          |
|--------|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| S      | Search     | The robot actively changes its head or torso viewpoint and examines different workspace regions to locate task-relevant objects, containers, or destination baskets.         |
| T      | Track      | The robot maintains visual alignment with a moving or task-relevant target during execution, such as tracking a moving recipient in a handover task.                         |
| I      | Interact   | The robot physically manipulates containers, occluders, or candidate objects to reveal information unavailable through passive observation or to complete physical subtasks. |
| F      | Focus      | The robot moves its camera, head, or a manipulated object to obtain a clearer view of fine-grained visual attributes, such as the size label of a T-shirt.                   |

**Real-world task suite.** We construct eight real-world manipulation tasks and collect 640 successful robot trajectories for policy fine-tuning. Table 18 summarizes the task instructions, required capabilities, and primary challenges.

Table 18: Real-world task suite and required active perception capabilities.

| ID | Task                                                | Capability | Main challenge                                                                                                 |
|----|-----------------------------------------------------|------------|----------------------------------------------------------------------------------------------------------------|
| T1 | Find the pen and place it in the basket             | S, I       | Search through a cloth bag until the pen is found, retrieve it, locate the basket, and complete the placement. |
| T2 | Find the peach and place it in the basket           | S, I       | Identify which drawer of a two-drawer cabinet contains the peach, retrieve it, and place it in the basket.     |
| T3 | Find the screwdriver and place it in the basket     | S, I       | Search through cluttered objects, identify the screwdriver, and transfer it to the basket.                     |
| T4 | Find the XL-size T-shirt and place it in the basket | S, I, F    | Inspect two visually similar T-shirts, identify the one with the XL-size label, and place it in the basket.    |
| T5 | Find the grape and place it in the basket           | S, I       | Determine which drawer of a three-drawer cabinet contains the grape, retrieve it, and complete the placement.  |
| T6 | Find the glue stick and place it in the basket      | S, I       | Search a cloth bag or drawer for the glue stick, retrieve it, and transfer it to the basket.                   |

Continued on next page

Table 18 (continued)

| ID | Task                                              | Capability | Main challenge                                                                                                                                                        |
|----|---------------------------------------------------|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| T7 | Find the banana and place it in the basket        | S, I       | Search a three-drawer cabinet containing distractor fruits, identify the banana, and place it in the basket.                                                          |
| T8 | Dynamic multi-person tracking and object handover | S, I, T    | Continuously distinguish and track members of the red team while both teams are moving, receive a bottle from red-team player 1, and deliver it to red-team player 3. |

**Reset and evaluation protocol.** Each method is evaluated for 20 trials on each of the eight tasks. After every trial, the robot returns to a fixed home configuration. An operator then reconstructs the predefined scene, including the target, source container, distractors, occluders, basket, background, illumination, and initial robot pose.

## E.2 RANDOMIZED PERTURBATIONS

**Per-episode randomization.** We introduce task-preserving perturbations to prevent the policies from exploiting fixed object coordinates, camera poses, or memorized action sequences. For each task, we generate 20 configurations in advance, and replay every configuration for all evaluated methods.

Table 19: Per-episode randomization used in the real-world evaluation. All pose offsets are defined relative to a task-specific nominal configuration.

| Variable              | Sampling range                                                                     | Validity constraint                                                                                   |
|-----------------------|------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------|
| Target position       | $\Delta x, \Delta y \sim \mathcal{U}(-5, 5)$ cm                                    | The target remains reachable but cannot be grasped directly from the initial configuration.           |
| Target orientation    | $\Delta \psi \sim \mathcal{U}(-20^\circ, 20^\circ)$                                | The range is extended to $\pm 45^\circ$ for elongated tools and flexible objects.                     |
| Source-region pose    | Position within $\pm 5$ cm; yaw within $\pm 15^\circ$                              | The container or occluder arrangement remains physically operable.                                    |
| Basket pose           | Position within $\pm 10$ cm; yaw within $\pm 20^\circ$                             | The basket remains spatially separated from the source region.                                        |
| Distractors           | $N_{\text{dist}} \sim \text{Unif}\{3, 4, 5, 6\}$                                   | Distractor identities are sampled from a task-irrelevant object pool.                                 |
| Occluders             | $N_{\text{occ}} \sim \text{Unif}\{1, 2, 3\}$                                       | The target is initially occluded but can be revealed through feasible interaction.                    |
| Head initialization   | Pitch within $\pm 10^\circ$                                                        | The sampled pose is clipped to the safe joint range.                                                  |
| Torso initialization  | Yaw within $\pm 10^\circ$                                                          | Both arms begin from the same safe home configuration.                                                |
| Illumination          | Intensity scale $\sim \mathcal{U}(0.7, 1.3)$ ; light azimuth within $\pm 25^\circ$ | The resulting image must not be severely underexposed or saturated.                                   |
| Background            | Uniform sampling from a predefined set                                             | The same physical background set is used for every method.                                            |
| T-shirt configuration | Randomized candidate order, overlap, orientation, and label pose                   | The size label is not reliably readable initially but becomes observable after feasible manipulation. |

**Experimental results.** Table 20 reports the success rates of the evaluated methods across the eight real-world tasks.

Table 20: Success rates (%) on the real-world task suite.

| Task                                                   | $\pi_{0.5}$  | SaPaVe       | MemER        | ActiveArena-OFT |
|--------------------------------------------------------|--------------|--------------|--------------|-----------------|
| Locate the pen and place it in the basket.             | 45           | 55           | 65           | 75              |
| Retrieve the peach and transfer it to the basket.      | 30           | 35           | 50           | 70              |
| Find the screwdriver and deposit it in the basket.     | 25           | 15           | 30           | 55              |
| Select the XL-size T-shirt and place it in the basket. | 0            | 5            | 10           | 40              |
| Retrieve the grape and put it in the basket.           | 35           | 40           | 45           | 70              |
| Locate the glue stick and move it to the basket.       | 25           | 25           | 40           | 65              |
| Find the banana and complete the placement.            | 25           | 15           | 25           | 50              |
| Track the relay game with moving red and blue teams.   | 5            | 5            | 10           | 20              |
| <b>Average</b>                                         | <b>23.75</b> | <b>24.38</b> | <b>34.38</b> | <b>55.63</b>    |

## F FURTHER DISCUSSION

**Why do we introduce an OOD setting, and is the current OOD protocol sufficient?** Recent studies have increasingly recognized that many existing robotic manipulation benchmarks essentially evaluate data fitting within a single distribution (Fei et al., 2025; Chen et al., 2026a; Yuan et al., 2026a). For example, benchmarks such as LIBERO often mix clean and randomized scenes within the same overall data distribution. While this setting is suitable for evaluating a model’s ability to learn tasks within a given distribution, it is inadequate for assessing generalization, because it cannot reveal how the model performs when the scene distribution changes.

We therefore provide two complementary evaluation protocols. The ID setting measures task learning within the training distribution, whereas the OOD setting evaluates generalization to changed scenes. Our OOD scenes include unseen backgrounds, illumination changes, a broader and more challenging object asset pool, and target categories from other tasks appearing as distractors in the current task. These changes are sufficient to test whether a model relies on visual patterns and distractor configurations specific to the training scenes. The substantial ID–OOD performance gaps observed across most models further indicate that the protocol is sufficiently discriminative, at least to a meaningful extent, for comparing their generalization ability.

**Are external baselines compared under fair input conditions?** Existing models require different forms of input, and this difference is even more pronounced for memory-based methods. Since research on robotic memory has not yet converged on a unified formulation, different methods rely on different memory representations and input interfaces.

As a benchmark, we therefore aim to provide training data that can support the input requirements of diverse methods. *ActiveArena-Sim* records rich annotations so that each model can construct the representations required by its original design. We do not modify the core architecture of an external baseline merely to enforce an artificial common input format. This evaluation practice is consistent with existing memory-based manipulation benchmarks, including RoboMME, RoboMemArena, and RMBench (Dai et al., 2026; Lei et al., 2026; Chen et al., 2026b).

To complement this external comparison, we further introduce *ActiveArena-VLA*, which performs controlled studies within a shared framework. It isolates the effects of memory, state input, auxiliary supervision, high-level planning, and action-head design. This internal suite partly addresses the difficulty of directly comparing heterogeneous external methods and provides useful insights into individual design choices. This is also one of the main motivations for developing *ActiveArena*.

**Is uniform sampling the optimal memory strategy?** Not necessarily. As discussed in the main paper, for perceptual memory, the key issue is not simply memory sparsity or the writing interval, but the stability of the writing trigger. If an event-driven trigger occurs too early or too late, it may write irrelevant observations or miss informative ones. Such noisy memory can further interfere with subsequent reasoning and lead to additional instability.



---

The chunk-based writing strategy avoids this problem by using a stable writing schedule. However, memory management is inherently more complex and also involves forgetting strategies and trade-offs between memory capacity and computational cost. We do not claim that the strategies explored in this work are optimal. Rather, we provide a framework for controlled comparison, and the evaluated strategies are sufficient to support the conclusions drawn in this paper.

Related studies in video understanding, such as FOCUS and AKS (Zhu et al., 2025; Tang et al., 2025), have also shown that uniform sampling often incurs only a limited performance drop compared with frame-selection methods based on learned visual scores, particularly for short videos. In embodied tasks, a video considered short in conventional video understanding may already correspond to a medium-horizon trajectory. This observation provides additional motivation for using regularly spaced sampling in manipulation tasks, where actions and scene contents often change gradually.

**Can task success rate reflect active-perception capability?** Yes. In *ActiveArena*, active perception is necessary for task completion: the robot must acquire missing information, retain it across views, and use it to guide subsequent actions. Therefore, task success rate provides a direct end-to-end measure of active-perception capability. Our failure decomposition further complements this metric by indicating whether failures mainly arise from missing relevant evidence, losing or failing to use context, or downstream reasoning and execution errors.

## G LIMITATIONS AND FUTURE WORK

The current version of *ActiveArena* primarily supports the Atribot S1. Future releases will extend *ActiveArena* to additional embodiments, such as the Unitree G1 and AgiBot G2. We will also expand the benchmark with more diverse and challenging tasks involving richer interactions, longer perception-action horizons, and more complex requirements for information acquisition, memory, and closed-loop reasoning.

We plan to open-source the simulator, benchmark, datasets, annotations, and evaluation tools, and continuously improve the platform based on feedback from the research community. In addition, we will provide more complete and standardized support for existing and emerging robotic foundation models, evaluate a broader range of VLA, memory-based, and world-action models, and maintain a reproducible leaderboard. Our long-term goal is to develop *ActiveArena* into an extensible and influential benchmark for active perception and robotic manipulation.

---

## H ADDITIONAL DEMONSTRATIONS

### H.1 SIMULATION TASK DEMONSTRATIONS

See Fig. 12, 13, 14, 15, 16, 17, 18, 19, 20.

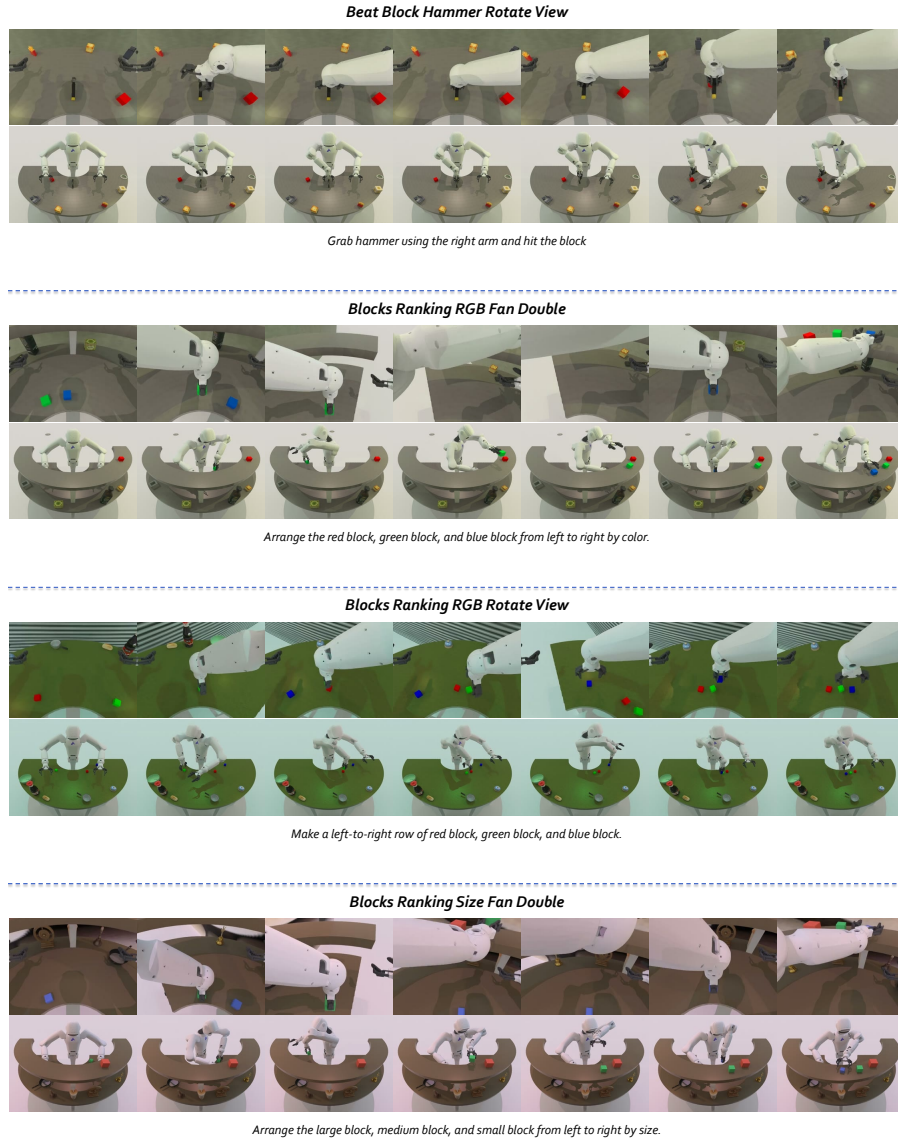


Figure 12: Visualization of our custom simulator with all task.

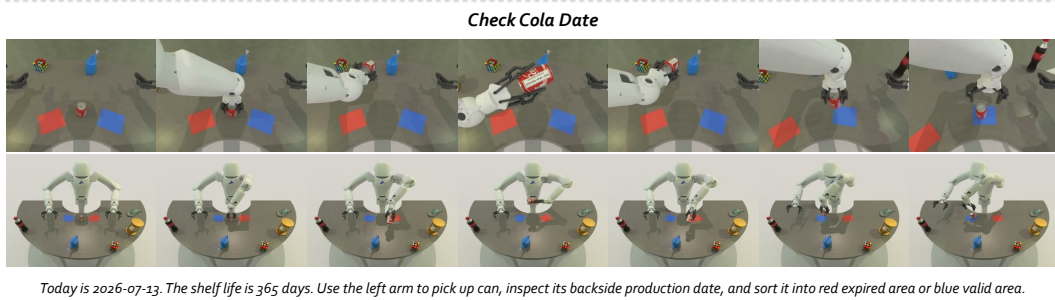
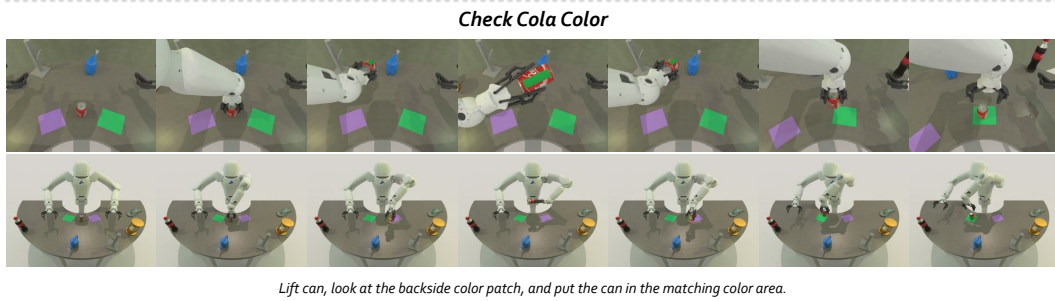
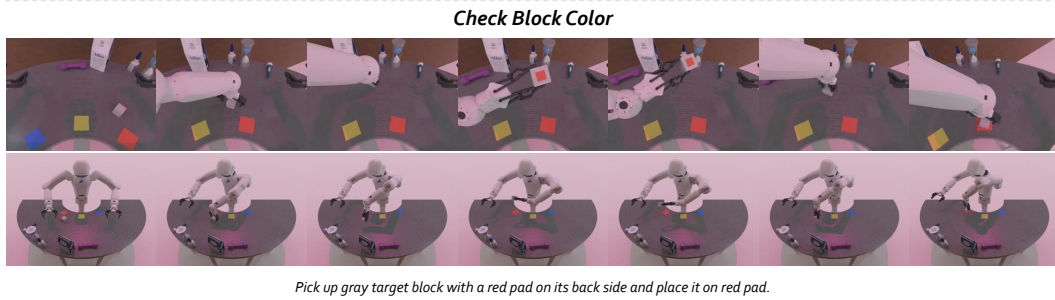
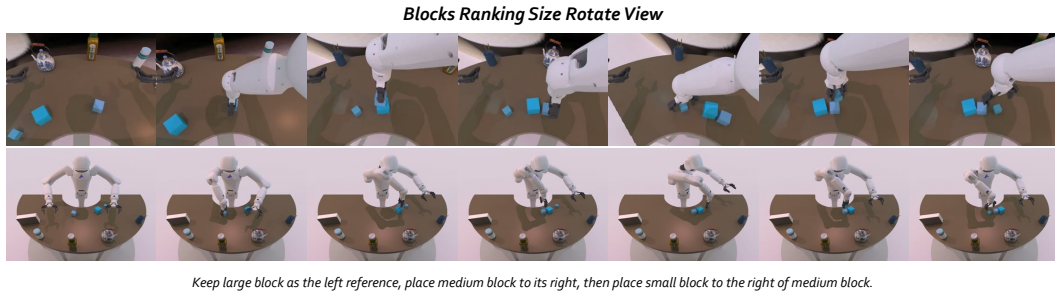


Figure 13: Visualization of our custom simulator with all task.

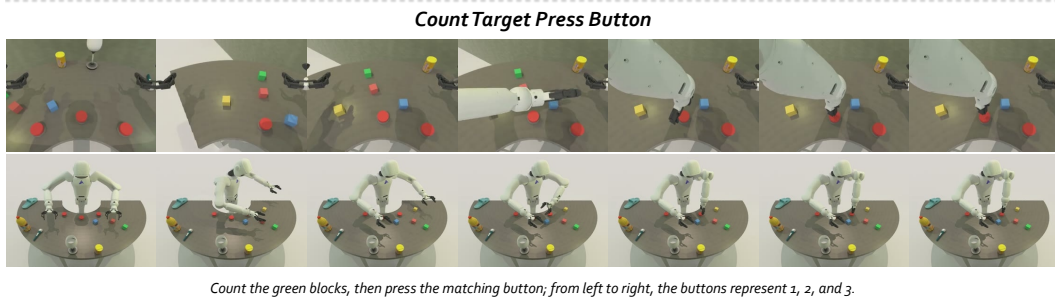
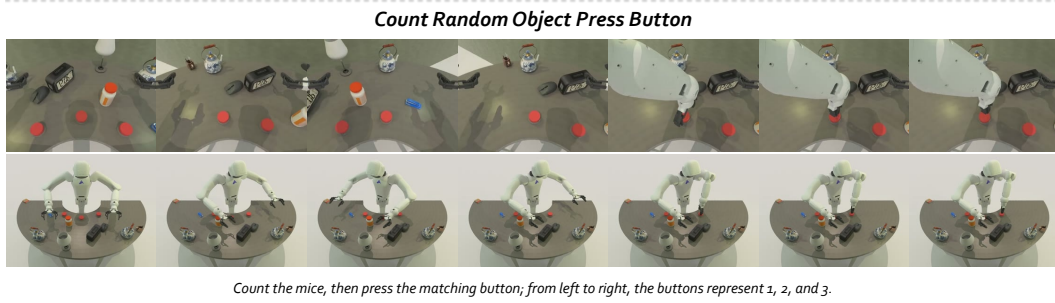
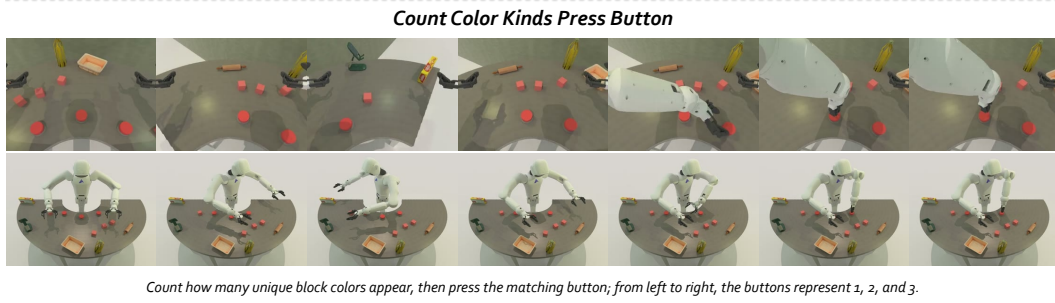
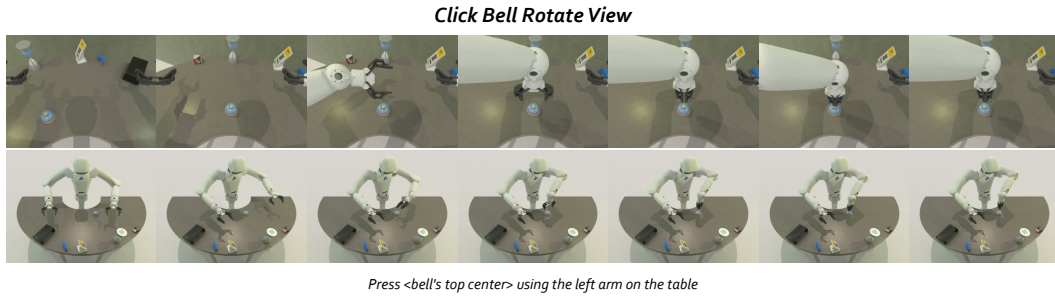
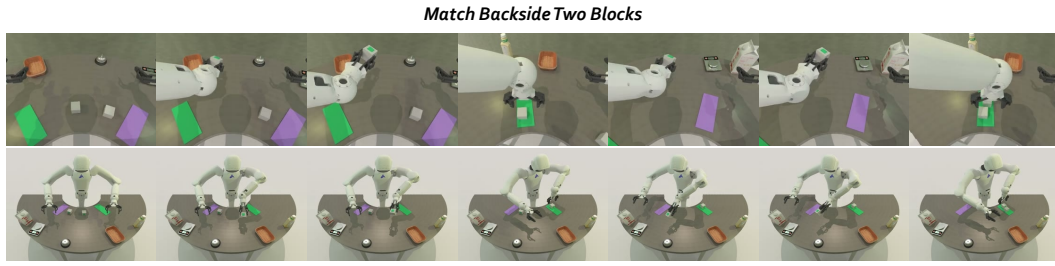
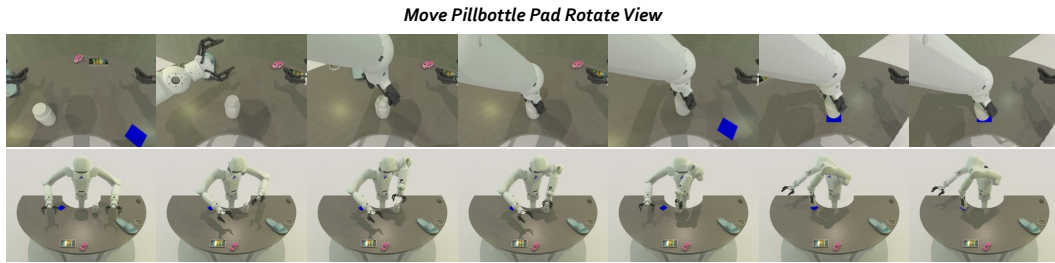


Figure 14: Visualization of our custom simulator with all task.

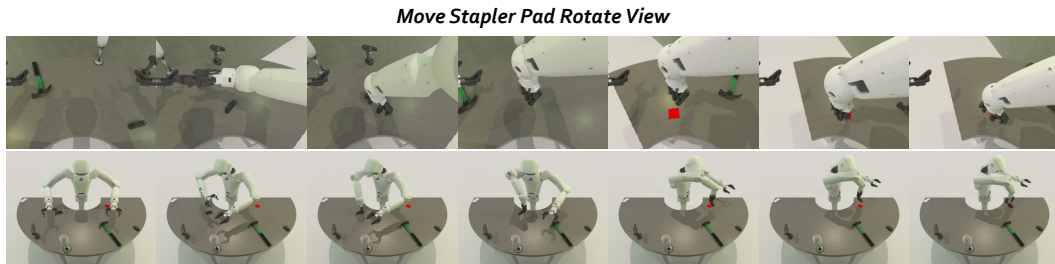




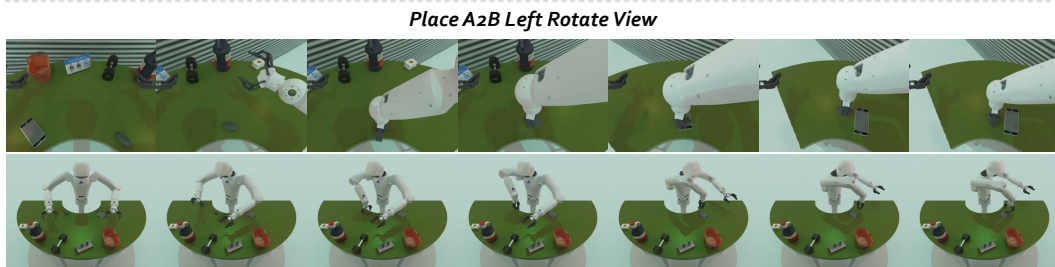
*Inspect the backside colors of two gray blocks and place each block on the matching colored pad.*



*Move pill bottle and drop it on the pad*



*Pick stapler using the right arm and move it to Red mat.*



*Place mouse carefully to phone's left*

Figure 15: Visualization of our custom simulator with all task.

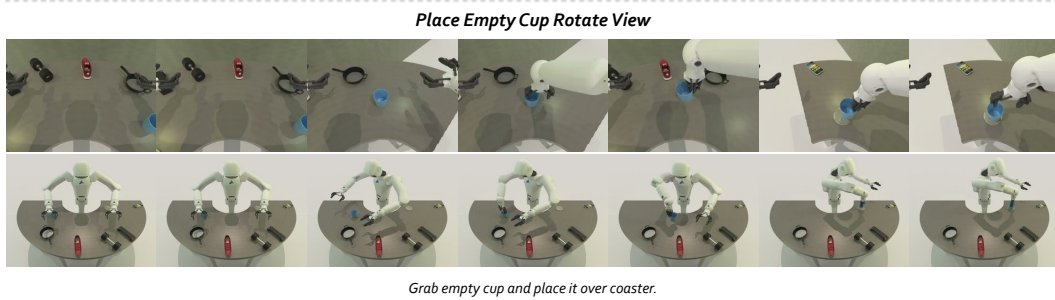
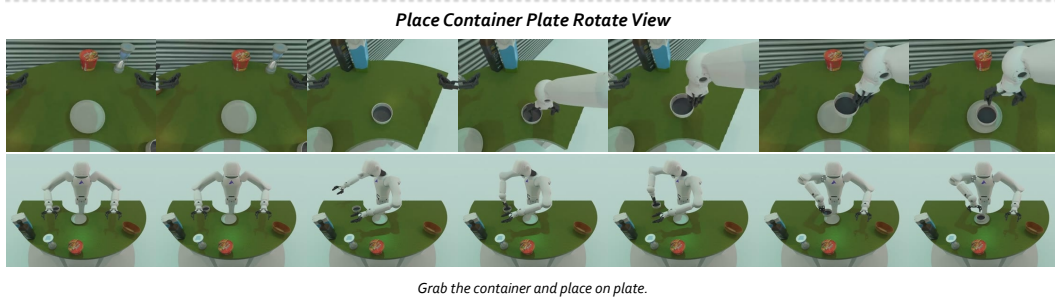
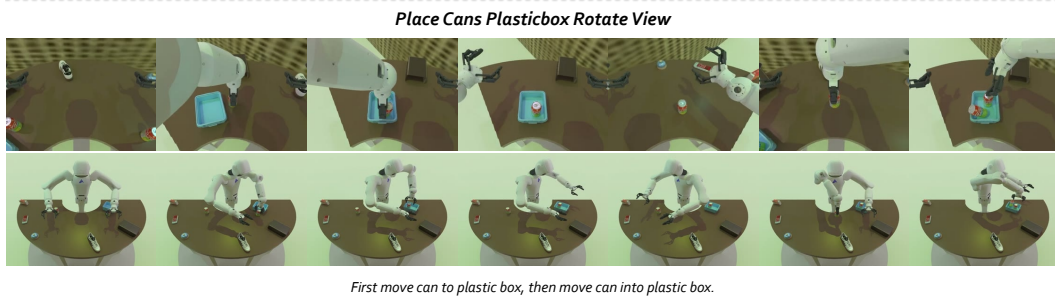
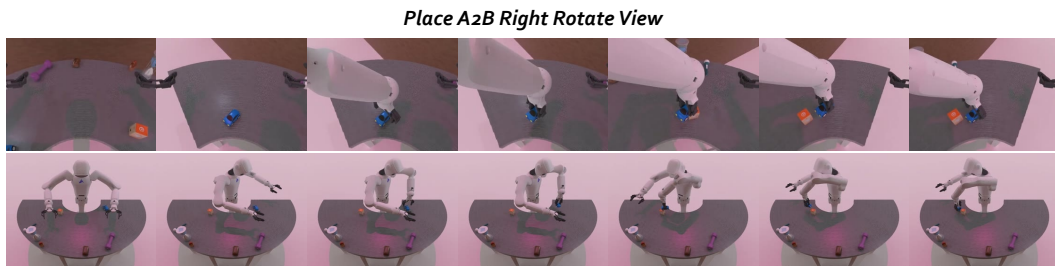


Figure 16: Visualization of our custom simulator with all task.

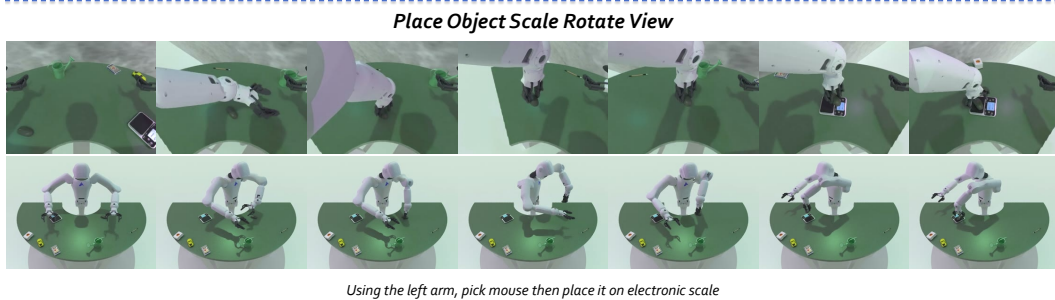
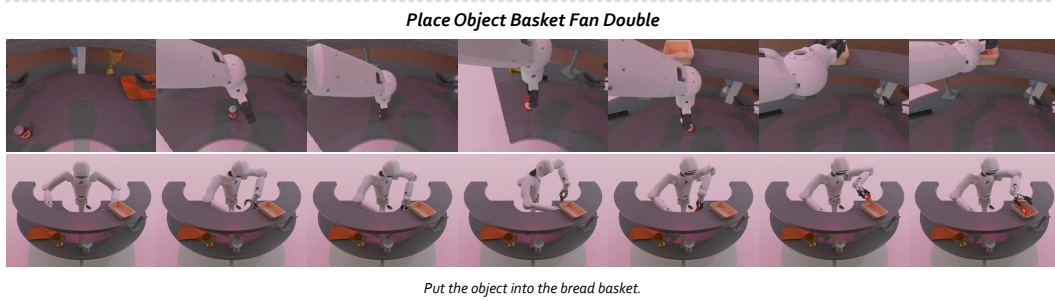
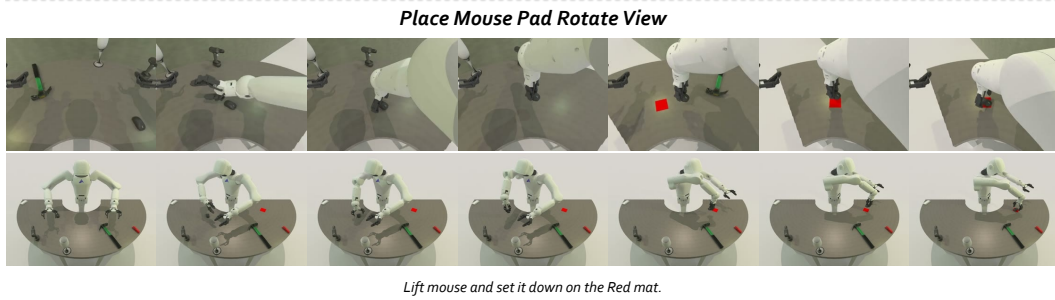
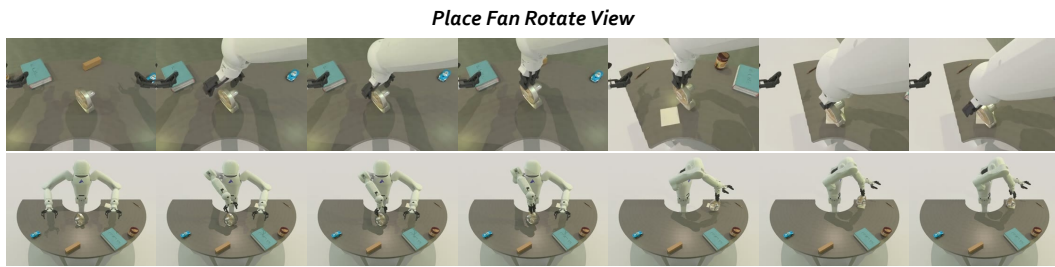


Figure 17: Visualization of our custom simulator with all task.



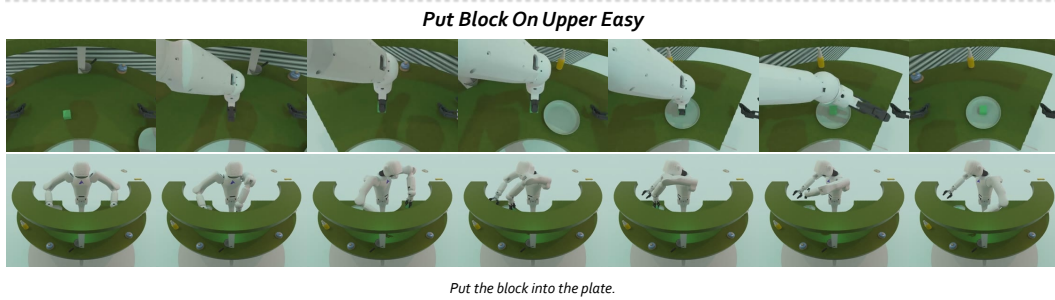
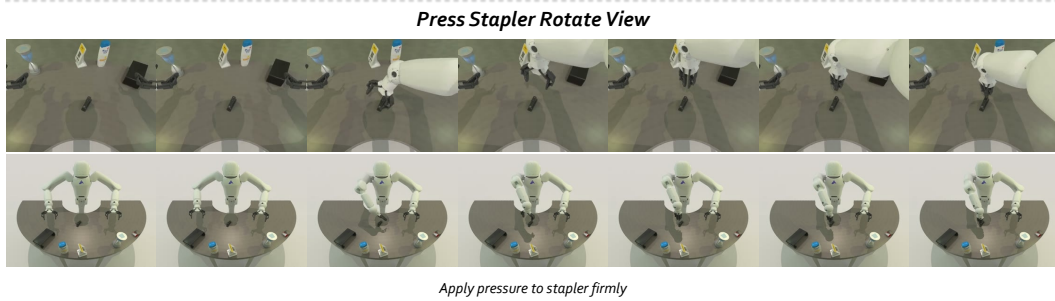
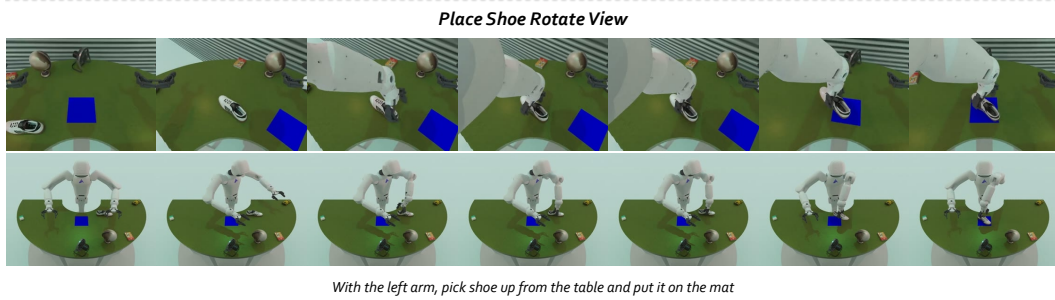
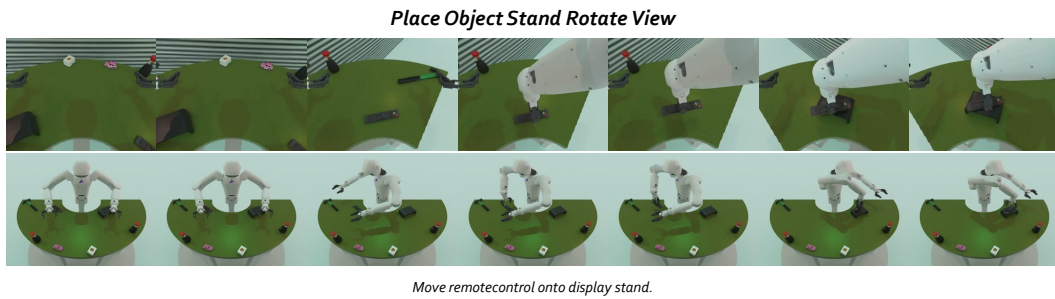


Figure 18: Visualization of our custom simulator with all task.

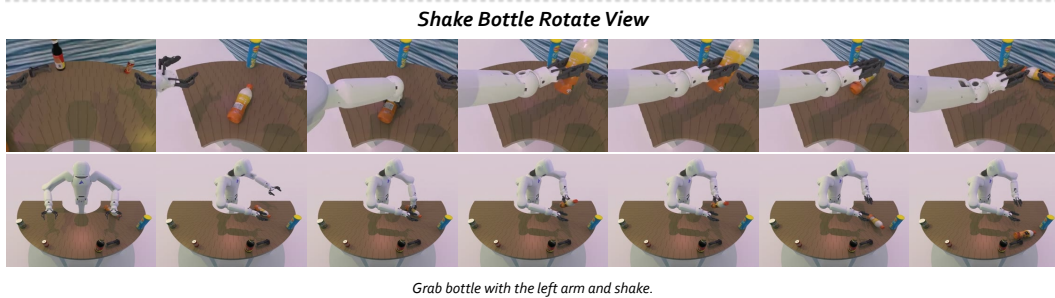
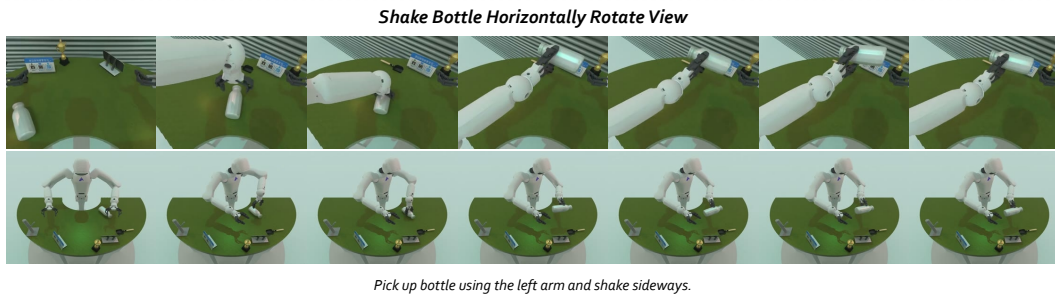
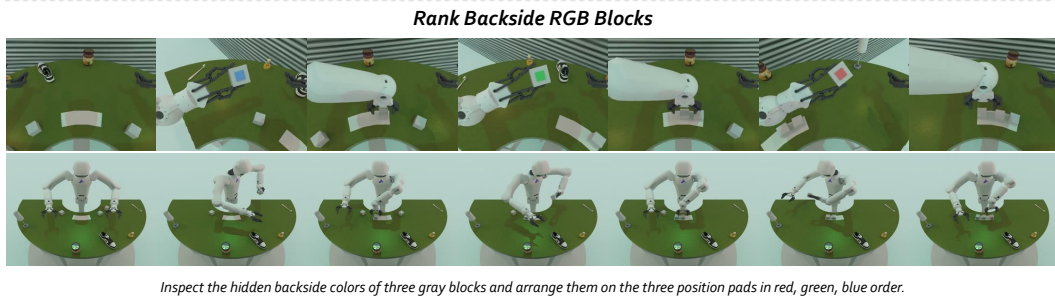
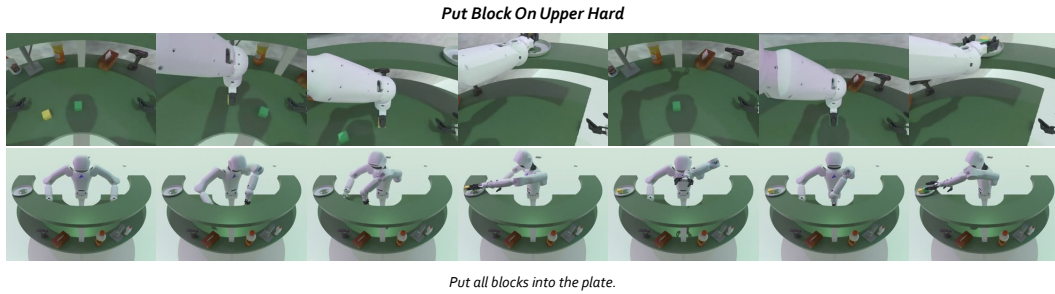


Figure 19: Visualization of our custom simulator with all task.

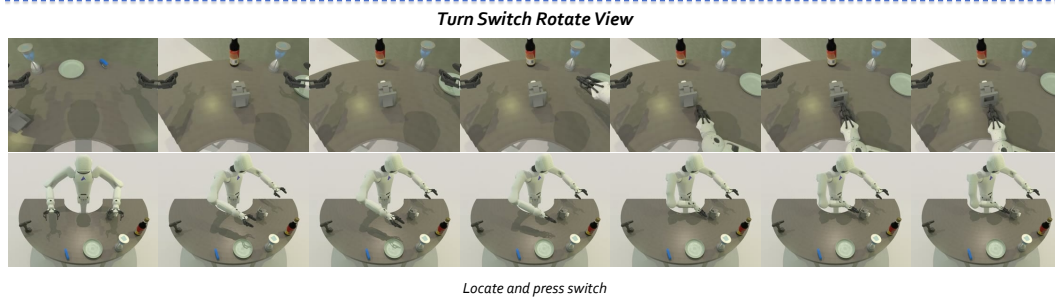
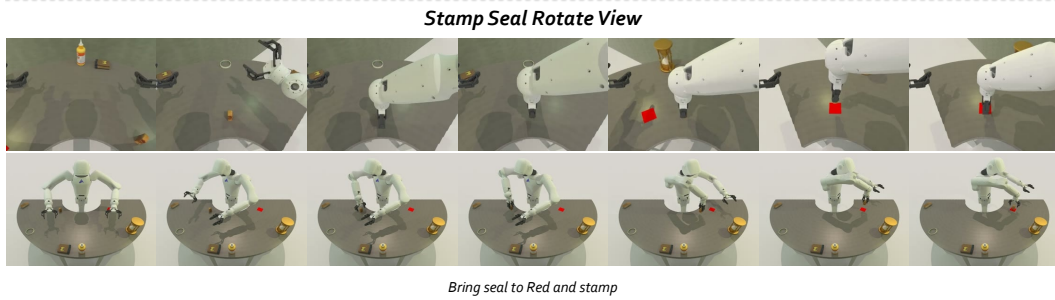
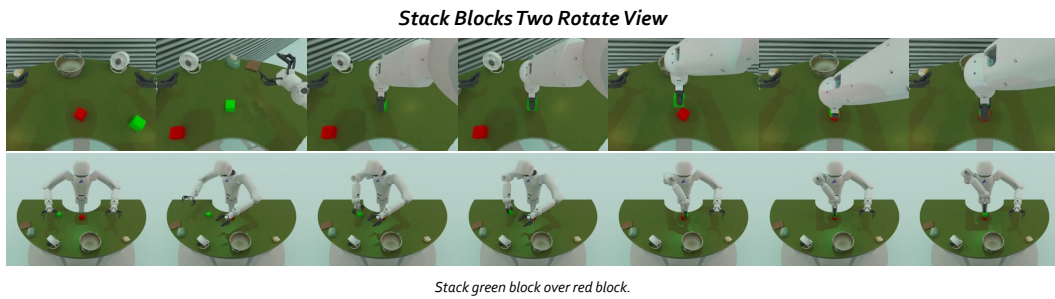


Figure 20: Visualization of our custom simulator with all task.



## H.2 REAL-WORLD TASK DEMONSTRATIONS

See Fig. 21.



Figure 21: Visualization of the eight real-world tasks.