ROBOMEMARENA: A CHALLENGING ROBOTIC MEMORY BENCHMARK

Anonymous authors
Paper under double-blind review

ABSTRACT

Memory is a critical component of robotic intelligence, as robots must rely on past observations and actions to accomplish long-horizon tasks in partially observable environments. However, existing robotic memory benchmarks still lack multimodal annotations for memory formation, provide limited task coverage and structural complexity, and remain restricted to simulation without real-world evaluation. We address this gap with *RoboMemArena*, a large-scale benchmark of 26 tasks, with average trajectory lengths exceeding 1,000 steps per task and 68.9% of subtasks being memory-dependent. The generation pipeline leverages a vision-language model (VLM) to design and compose subtasks, generates full trajectories through atomic functions, and provides memory-related annotations, including subtask instructions and native keyframe annotations, while paired real-world memory tasks support physical evaluation. We further design *PrediMem*, a dual-system VLA in which a high-level VLM planner manages a memory bank with recent and keyframe buffers and uses a predictive coding head to improve sensitivity to task dynamics. Extensive experiments on RoboMemArena show that PrediMem outperforms all baselines and provides insights into memory management, model architecture, and scaling laws for complex memory systems.

1 INTRODUCTION

Memory is a critical component of robotic intelligence, as it determines whether a robot can accomplish long-horizon and complex tasks in partially observable environments. With the advancement of robot foundation policies (Kim et al., 2025; Black et al., 2025; Intelligence et al., 2025), recent research (Shi et al., 2025; Sridhar et al., 2026; Lin et al., 2025; Torne et al., 2026; Fang et al., 2025) has begun to endow these foundation models with effective memory mechanisms, enabling them to handle longer-horizon and more complex tasks.

This trend drives the development of corresponding benchmarks (Fang et al., 2025; Cherepanov et al., 2026; Chen et al., 2026a). However, existing robotic memory benchmarks suffer from several limitations. (1) Their datasets lack the multimodal annotations necessary for memory formation. Recent works (Torne et al., 2026; Intelligence et al., 2026) have highlighted the inherently multimodal nature of robotic memory. Similar to human memory, challenging memory representations may include multiple modalities, such as visual information (*e.g.*, keyframe images) and language (*e.g.*, subtask instructions). Existing benchmarks, however, do not provide such annotations. (2) Their task coverage remains limited: they primarily focus on short-term memory, exhibit relatively low structural complexity, offer limited task diversity, and, in many cases, include tasks that do not genuinely require memory. (3) These benchmarks are restricted to simulation and lack corresponding real-world robotic evaluations. As a result, there remains a significant gap between memory effectiveness in simulated planning and execution in the physical world.

We address these gaps with *RoboMemArena*, a large-scale benchmark built from the ground up for evaluating embodied memory. In *RoboMemArena*, we design and compose multiple subtasks using a vision-language model (VLM), generate the full trajectory through atomic functions, and subsequently provide memory-related annotations (*i.e.*, subtask instructions and keyframe annotations). This automated pipeline (Figure 1) is well-suited for large-scale data generation. The simulated benchmark contains 26 tasks across 4 memory-dependent categories (transferring, occlusion, counting, sequential execution), with an average trajectory length of 1,076 steps per task and 68.9% history-dependent

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

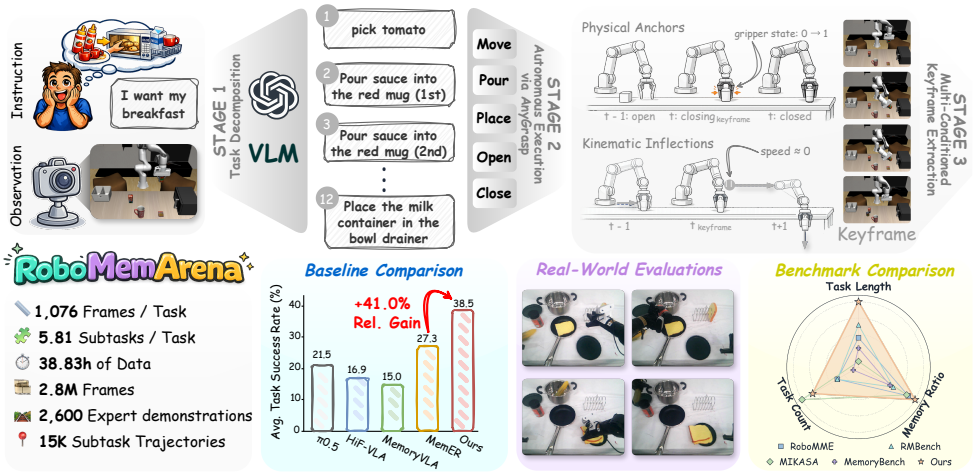


Figure 1: **Overview of RoboMemArena.** Our scalable annotation pipeline converts natural-language instructions into keyframe-annotated trajectories via VLM-based task decomposition, autonomous execution, closed-loop verification, and targeted human refinement of unsuitable annotations.

subtasks, which is the highest ratio among existing robotic benchmarks. As a complement to the simulated benchmark, which focuses on scalability and reproducibility, we provide real-world tasks for physical evaluation. Specifically, we design 8 challenging real-world memory tasks, with the most complex demonstrations lasting over three minutes.

Furthermore, we design **PrediMem**, a dual-system VLA that pairs a high-level VLM planner to harness hierarchical memory with a low-level VLA actor. The VLM manages a memory bank, including a recent buffer and a keyframe buffer. To enhance the sensitivity to the choice of keyframes, it is combined with a predictive coding head to better understand world dynamics of events and the progression of tasks.

In summary, our contributions are:

- **Benchmark.** We introduce *RoboMemArena*, a challenging benchmark suitable for validating robotic memory. It is equipped with multimodal memory-related annotations, long-horizon and diverse tasks, while supporting real-world tasks.
- **Model.** We propose **PrediMem**, a dual-system memory VLA baseline with predictive coding.
- **Experiments.** We evaluate representative baselines and variants of PrediMem on *RoboMemArena*, showing insights into memory management, model architecture, and scaling laws for memory-augmented robotic manipulation.

2 Related Work

2.1 Robotic Memory Benchmarks

Existing robotic manipulation benchmarks (Xiang et al., 2020; Mu et al., 2024; Nasiriany et al., 2024; Tao et al., 2024; Li et al., 2023; Lu et al., 2024; Wang et al., 2025) cover broad objects, scenes, and skills, but many tasks remain locally observable and therefore do not isolate memory as the central bottleneck. Recent memory-oriented benchmarks (Cherepanov et al., 2026; Fang et al., 2025; Chen et al., 2026a; Dai et al., 2026) move closer to this goal, yet three gaps remain. First, they lack rich multimodal memory annotations that can directly supervise dual-system planners. Second, existing memory benchmarks are often limited in task scale and diversity. MemoryBench (Fang et al., 2025) and MIKASA (Cherepanov et al., 2026) are memory-focused, but both remain short-horizon and mainly evaluation-oriented. RMBench (Chen et al., 2026a) broadens memory-complexity settings, but the task coverage is relatively small. Third, most benchmarks are not paired with aligned real-world memory evaluations. RoboMME (Dai et al., 2026) standardizes multiple memory dimensions, but its annotations mainly focus on subtask-boundary keyframes and stage-level signals rather than richer multimodal memory supervision. By contrast, *RoboMemArena* addresses these gaps jointly with

native multimodal supervision, scalable memory-dependent manipulation tasks, and paired real-world memory evaluations.

2.2 VLA Models with Memory

Large-scale VLA pretraining has produced strong language-conditioned manipulation backbones (Kim et al., 2025; Black et al., 2025; Intelligence et al., 2025; Octo Model Team et al., 2024; Zheng et al., 2025; Chi et al., 2025; Zhao et al., 2023; Wen et al., 2025), with recent extensions adding multi-frame context and future-aware action modeling (Li et al., 2024; 2026a; Lin et al., 2025; Jang et al., 2025; Hu et al., 2026). We refer to VLAs that predict the next action from the current observation without event memory as *reactive policies*. These policies become brittle when task-relevant information lies in the past. Memory-augmented VLAs (Sridhar et al., 2026; Han et al., 2025; Shi et al., 2025; Hu et al., 2025; Koo et al., 2025; Li et al., 2025; 2026d; Torne et al., 2026; Sun et al., 2026; Wang et al., 2026; Fang et al., 2025; Bi et al., 2025; Yuan et al., 2026; Liu et al., 2026; Li et al., 2026c;b) address this limitation through visual retrieval, history reasoning, working memory, temporal caches, or multimodal memory compression. For keyframe selection, prior work uses gripper or velocity heuristics, progress-aware embeddings, or retrieved visual keyframes (James & Abbeel, 2022; Chen et al., 2026b; Sridhar et al., 2026). PrediMem differs by using predictive coding to reshape the shared VLM hidden space, allowing keyframes to be selected through the standard language model (LM) head without extra retrieval modules at inference.

3 RoboMemArena

This section presents the task suite (Section 3.1), the data generation pipeline (Section 3.2), a comparison with existing benchmarks (Section 3.3), and the evaluation protocol (Section 3.4).

3.1 Task Setting

Simulation. *RoboMemArena* is designed to evaluate the regime in which the next action depends on task-relevant information that is no longer visible. The 26 tasks cover four representative failure modes of reactive policies: **(1) Multi-Object Transferring.** The agent relocates multiple objects between visually identical containers and must remember the source–target mapping and which transfers have already been completed. **(2) Multi-Object Occlusion.** The agent places objects into drawers or cabinets that later become visually closed, so it must remember what was placed, where it was placed, and the prior state of each container. This category is the largest in our benchmark (11 tasks). **(3) Multi-Object Counting.** The agent must perform an action a specified number of times (*e.g.*, pour exactly twice), even when the scene before consecutive repetitions looks nearly identical. **(4) Multi-Object Sequence.** The correct downstream action depends on an earlier subtask outcome, such as placing a new object into the same container used in a previous step. The challenge is not only the hidden state, but also resolving references that span multiple operations.Rather than differing only in object identity, the tasks vary in scene, action structure, subtask order, and the task-relevant state to be remembered, as detailed for the largest category in Appendix J. Figure 2 shows one representative task per category, and Table 6 (Appendix E) describes all 26 tasks.

Real-world Tasks. Beyond simulation, *RoboMemArena* is paired with eight real-world memory tasks on a dual-arm platform (Table 4), which cover counting, occlusion, sequential execution, hidden-target tracking, object-state and target-identity memory, and memory conditioned on human demonstration. Demonstrations are collected by leader–follower teleoperation on the AgileX Cobot Mobile ALOHA platform at 30 Hz (100 successful demonstrations per task), and each task is evaluated with 50 rollouts. We use them as a physical validation set for the benchmark design. Detailed task descriptions and representative snapshots are provided in Table 8 and Figure 7 (Appendix G).

3.2 Automated Data Generation Pipeline

RoboMemArena resolves the usual trade-off between scalable automatic collection and fine-grained temporal annotation through three stages (Figure 1).

Stage 1. VLM-Driven Task Decomposition. Given a high-level instruction and the current RGB observation, a VLM proposes an ordered sequence of executable subtasks as scalable initial annotations in simulation. We then manually refine the subset of decompositions that are unsuitable

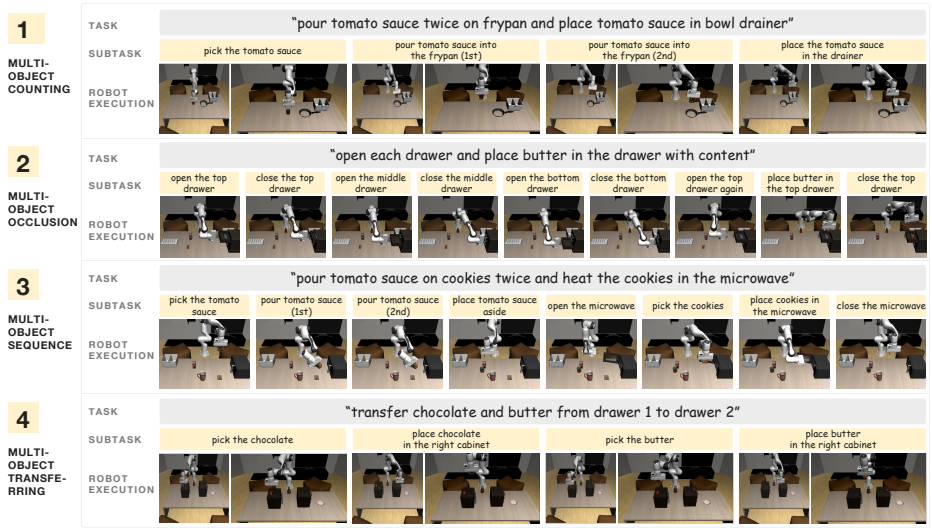


Figure 2: **Visualization of 4 task categories of RoboMemArena.** Each row shows the task instruction, subtask decomposition, and execution rollout for Multi-Object Counting, Occlusion, Sequence, and Transferring.

or inconsistent before downstream execution, and verify that each designated downstream subtask requires information from earlier observations or action outcomes. The prompt is designed to expose memory demands such as occlusion, counting, and order-dependent execution.

Stage 2. AnyGrasp-Based Autonomous Generation. Each subtask is executed autonomously using AnyGrasp (Fang et al., 2023), a 6-DoF grasp-pose estimator operating on point-cloud input. Estimated poses are dispatched to predefined primitives to generate action trajectories. Moreover, we add a post-condition checker that retries failed subtasks with updated grasp poses. This closed-loop execution keeps collection automatic while maintaining high success rates.

Stage 3. Multi-Conditioned Keyframe Extraction. Fixed-frequency sampling either misses state transitions or stores redundant static frames. Let a continuous trajectory be denoted as $\tau = \{(s_t, a_t)\}_{t=1}^{T}$, where s_t is the state and a_t is the action at timestep t. We extract the keyframe set $\mathcal{K} = \mathcal{K}_{\text{phys}} \cup \mathcal{K}_{\text{kin}}$ as the union of frames satisfying either of two physically grounded conditions:

1. **Physical interaction anchors.** Gripper-state transitions mark grasp closure and release. With the gripper state $g_t \in \{0, 1\}$ (1 = closed, 0 = open), the anchor set is $\mathcal{K}_{\text{phys}} = \{t \in [1, T] \mid g_t \neq g_{t-1}\}$.
2. **Kinematic inflections.** End-effector velocity minima and abrupt direction changes mark transitions between motion phases. Let $\mathbf{v}_t \in \mathbb{R}^3$ be the end-effector linear velocity. We identify a kinematic inflection at timestep t if the velocity magnitude drops below a threshold ϵ or the cosine similarity between consecutive velocity vectors falls below $\cos(\theta)$:

$$\mathcal{K}_{\text{kin}} = \left\{ t \in [1, T] \;\middle|\; \|\mathbf{v}_t\| < \epsilon \;\vee\; \frac{\mathbf{v}_t \cdot \mathbf{v}_{t-1}}{\|\mathbf{v}_t\| \, \|\mathbf{v}_{t-1}\|} < \cos(\theta) \right\}. \tag{1}$$

Together, these conditions select a compact set of event frames that reconstruct task progress and provide temporal supervision for VLMs without storing dense video.Human annotators then review the automatically extracted candidate keyframes and refine unsuitable selections before the trajectories are released.

3.3 Data Analysis

Benchmark Comparison. We provide a thorough comparison between *RoboMemArena* and 14 established benchmarks across 8 feature dimensions in Table 1. *RoboMemArena* is the only entry that satisfies all eight criteria.

Memory-Dependent Subtask Ratio. In total, *RoboMemArena* defines 151 distinct subtasks across its 26 tasks. We consider a subtask memory-dependent if its correct execution cannot be inferred

Table 1: **Comparison with popular benchmarks used in the robot learning literature.** *RoboMemArena* features long-horizon memory tasks, multimodal memory-related annotations, scalable generation pipelines, and paired real-world evaluations.

Benchmark	Long Horizon	Auto Instr. Gen.	Atomic Subgoals	Scalable Gen.	Autonomous Grasp	State Oracle	Native Keyframes	Real-World Eval.
RLBench (James et al., 2020)	×	×	×	✓	×	×	×	×
RoboCerebra (Han et al., 2025)	✓	✓	✓	×	×	✓	×	×
ARNOLD (Gong et al., 2023)	×	×	×	✓	×	×	×	×
ALFRED (Shridhar et al., 2020)	×	×	✓	✓	×	✓	×	×
CALVIN (Mees et al., 2022)	×	×	×	×	×	×	×	×
RoboCasa (Nasiriany et al., 2024)	×	✓	✓	✓	×	✓	×	✓
LIBERO-Long (Liu et al., 2023)	×	×	×	×	×	×	×	×
VLABench (Zhang et al., 2025)	×	✓	✓	✓	×	✓	×	×
RoboTwin (Mu et al., 2024)	×	✓	×	✓	×	×	×	✓
RMBench (Chen et al., 2026a)	×	✓	✓	✓	×	✓	✓	✓
RoboMME (Dai et al., 2026)	×	×	✓	✓	×	✓	✓	✓
BEHAVIOR-1K (Li et al., 2023)	✓	×	✓	×	×	✓	×	✓
MIKASA (Cherepanov et al., 2026)	×	×	✓	×	×	✓	×	✓
MemoryBench (Fang et al., 2025)	×	×	✓	×	×	✓	×	✓
RoboMemArena (Ours)	✓	✓	✓	✓	✓	✓	✓	✓

Note: **Long Horizon**: Tasks are categorized as long-horizon when the average trajectory length is greater than 1000 steps. **Auto Instr. Gen.**: Automated generation or augmentation of task instructions using language models, vision-language models, or multimodal models. **Atomic Subgoals**: Tasks provide explicit step-level subgoals, subtask annotations, or symbolic atomic goal predicates beyond a single final goal. **Scalable Gen.**: Automated batch generation of executable trajectories, demonstrations, or dense action annotations in simulation, rather than manual trajectory collection. **Autonomous Grasp**: Autonomous grasping through point-cloud-based grasp pose estimation, enabling executable grasp actions without manual grasp annotations. **State Oracle**: Programmatic signals for judging subtask or stage completion from simulator execution traces, object states, or symbolic predicates. **Native Keyframes**: Explicit temporal keyframe dataset construction for hierarchical supervision or memory-oriented evaluation. **Real-World Eval.**: The benchmark paper or project includes explicit real-robot evaluation associated with the benchmark setup.

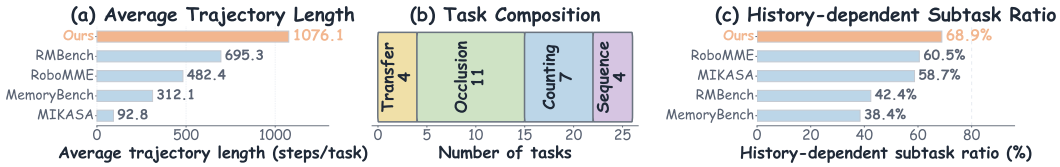


Figure 3: **Summary statistics of RoboMemArena.** Panels (a)–(c) respectively illustrate the average trajectory length, task composition, and history-dependent subtask ratio, highlighting the long-horizon nature of *RoboMemArena* and the prevalence of memory-conditioned subtasks relative to prior benchmarks. MIKASA denotes the RL suite MIKASA-Robo; control rates and durations in seconds are given in Table 9.

from the current observation alone and requires information from earlier subtasks or observations. Across all tasks in *RoboMemArena*, 104 of 151 subtasks are memory-dependent, giving a **68.9%** history-dependent subtask ratio. Figure 3(c) shows *RoboMemArena* also has the highest history-dependent subtask ratio among all robotic memory benchmarks. The calculation protocol is detailed in Appendix D.

Scale and Diversity. For each of the 26 tasks, we collect 100 successful demonstrations, yielding 2,600 long-horizon visual trajectories. These produce 15,100 keyframe-aligned short segments for hierarchical supervision. In terms of average trajectory length, our *RoboMemArena* is longer than existing robotic memory benchmarks, achieving 1,076 steps (53.8 s at 20 Hz) per task, as shown in Figure 3(a); Table 9 compares durations in seconds. Figure 3(b) shows the task composition of *RoboMemArena*, which includes 4 transferring tasks, 11 occlusion tasks, 7 counting tasks, and 4 sequence tasks.

3.4 Evaluation Protocol

Binary success alone is insufficiently informative for long-horizon memory tasks. Therefore, we report both full-task success and partial progress.

Task Success Rate (TSR). To determine whether a task is successful, we verify it through multiple stage-level predicates rather than only checking the final outcome. For the i-th task, we define K_i stage-level verification predicates $\psi(s_i^{(k)})$ for $k = 1, \ldots, K_i$, where $s_i^{(k)}$ denotes the execution state at the k-th verification stage. These predicates encode state conditions such as object location,

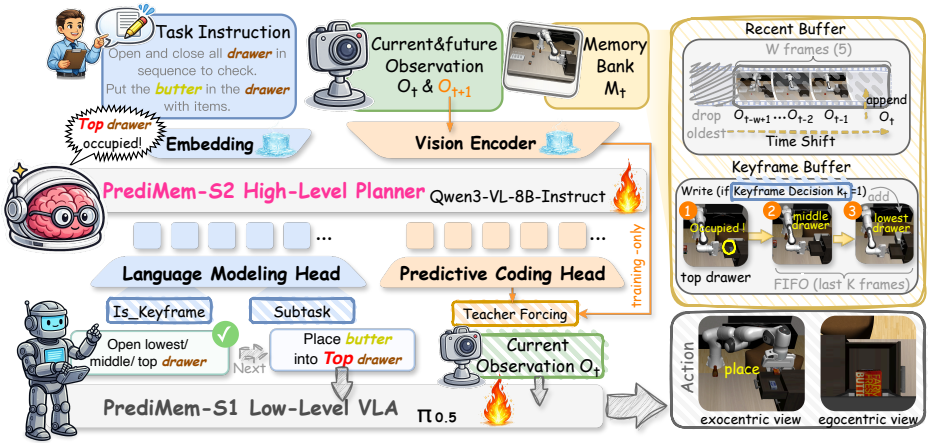


Figure 4: **The PrediMem pipeline.** The pipeline comprises two asynchronously coupled components: S1, a low-level action policy that executes the current subtask, and S2, a high-level planner that predicts keyframes and dispatches the next subtask. The predictive coding head path is training-only.

containment, visibility, and stage completion. Each predicate returns `True` if the corresponding condition holds. A task is deemed successful only when *all* predicates are satisfied:

$$\mathrm{TSR} = \frac{1}{N}\sum_{i=1}^{N}\prod_{k=1}^{K_i}\mathbf{1}\left[\psi\left(s_i^{(k)}\right)\right], \tag{2}$$

where N is the total number of evaluated tasks, and $\mathbf{1}[\cdot]$ denotes the indicator function, which equals 1 if the predicate is satisfied and 0 otherwise.

Cumulative Success Rate (CSR). Rather than requiring all-or-nothing success, CSR measures the *fraction* of verification stages that each task completes, thereby quantifying task progress:

$$\mathrm{CSR} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{K_i}\sum_{k=1}^{K_i}\mathbf{1}\left[\psi\left(s_i^{(k)}\right)\right]. \tag{3}$$

Tasks have 4 to 9 verification stages, most more than 5 (Figure 8, Appendix H), which gives CSR enough resolution to compare partial progress.

4 PrediMem: Building Hierarchical Memory with Predictive Coding

We introduce **PrediMem**, a hierarchical **Mem**ory framework with **Predi**ctive coding for embodied memory. It consists of a high-level planner (System 2, denoted by S2), a low-level execution policy (System 1, denoted by S1), a keyframe-grounded memory bank, and an auxiliary predictive coding head. As shown in Figure 4, the memory bank $\mathcal{M}_t$ combines a long-term keyframe buffer $\mathcal{M}_t^{\mathrm{key}}$ with a recent sliding window with fixed horizon $\mathcal{M}_t^{\mathrm{rec}}$, *i.e.*, $\mathcal{M}_t = \mathcal{M}_t^{\mathrm{key}} \cup \mathcal{M}_t^{\mathrm{rec}}$. S2 takes the current observation together with the memory bank to predict the current subtask and decide whether the current frame should be stored as a keyframe. Accepted keyframes are written back into the keyframe buffer, allowing the system to preserve decision-critical events beyond the recent observation window. Meanwhile, S1 predicts the freshest subtask-conditioned action chunk.

Predictive Coding. The key question is *when* to write frames to memory: over-storing wastes capacity, while missed transitions cause downstream errors. To enhance the model's sensitivity to keyframes and ability to capture future dynamics, we introduce predictive coding. To this end, we incorporate an additional predictive coding head f_{Pre} that predicts the visual feature of the subsequent frame from the S2 hidden state h_t, which encodes the instruction, memory context, and current observation: $\hat{Z}_{t+1} = f_{\mathrm{Pre}}(h_t)$, with supervision Z_{t+1} provided by the visual encoder of the VLM, *i.e.*, a frozen ViT. Following Cambrian-S (Yang et al., 2025), the predictive loss $\mathcal{L}_{\mathrm{Pre}}$ is formulated as the sum of a latent Mean Squared Error term and a cosine-distance term against the stop-gradient

teacher next-frame latent features:

$$\mathcal{L}_{\text{Pre}} = \text{MSE}\big(\hat{Z}_{t+1}, \text{sg}(Z_{t+1})\big) + \big(1 - \cos\big(\hat{Z}_{t+1}, \text{sg}(Z_{t+1})\big)\big). \tag{4}$$

Total Training Loss. The final objective for S2 combines next-token prediction with the predictive coding loss $\mathcal{L}_{\text{S2}} = \mathcal{L}_{\text{text}} + 0.1\mathcal{L}_{\text{Pre}}$. Here, $\text{sg}(\cdot)$ denotes the stop-gradient operator, and $\mathcal{L}_{\text{text}}$ denotes the next-token prediction loss for subtask generation and keyframe decisions. The loss function used for S1 follows the official flow-matching objective of $\pi_{0.5}$ (Black et al., 2025).

Inference. At inference, the predictive coding head is removed, so PrediMem keeps the architecture and cost of a standard dual-system framework; the two systems run asynchronously (Appendix C).

5 Experiments

We evaluate *RoboMemArena* and the **PrediMem** framework around five questions:

Q1. Does *RoboMemArena* expose a memory gap in existing VLAs, and can PrediMem close it?
Q2. Does the end-to-end trained robot memory system surpass powerful closed-source agents?
Q3. How much do the predictive coding head and the keyframe bank contribute, and how does predictive coding shape the learned memory representations?
Q4. How does the scaling of memory influence model performance?
Q5. How do different baselines perform in the real-world evaluation of *RoboMemArena*?

5.1 Experimental Setup

Baselines. We compare against $\boldsymbol{\pi_{0.5}}$ (Black et al., 2025), which is a reactive VLA that acts only on the current observation. We also compare with **HiF-VLA** (Lin et al., 2025), which models hindsight, insight, and foresight motion representations, **MemoryVLA** (Shi et al., 2025), which uses token-level working memory, and **MemER** (Sridhar et al., 2026), which follows a dual-system design with keyframe retrieval.

Implementation. We report TSR and CSR (Section 3.4), averaged over three training seeds with 100 rollouts per simulated task (Appendix I). PrediMem builds on Qwen3-VL-8B-Instruct with the vision tower frozen and the remaining modules fully fine-tuned for 2 epochs on $4\times$H100 with learning rate 1×10^{-5}. The predictive coding head uses latent MSE and cosine losses (weight 0.1 each). The recent buffer holds 5 frames, and the keyframe buffer is uncapped. The prompt format is given in Appendix B.

5.2 Main Results in Simulation (Q1, Q2)

Table 2(a) shows $\pi_{0.5}$ reaches 21.5% average TSR and 38.6% average CSR. Because $\pi_{0.5}$ is a reactive policy without explicit history modeling, it attains relatively high success rates only on certain sequential tasks, where many intermediate steps remain governed by local visual regularities and do not inherently require recalling distant past events. However, when a task requires remembering an earlier state, it fails almost completely, e.g., by reopening a drawer it has already inspected (Appendix F). HiF-VLA introduces richer motion representations, which help short-horizon execution, but it does not explicitly store task-level events such as object placements, previous drawer states, or action counts. MemoryVLA stores history as transformer tokens, but this token-level memory is not explicitly aligned with the sparse physical transitions that determine task progress in our benchmark. MemER performs better than purely reactive baselines because it retrieves visual keyframes and follows a dual-system design. However, constrained by its limited perception of task dynamics, the high-level VLM is often unable to select informative keyframes.

In comparison, PrediMem combines an explicit keyframe bank with predictive coding. The keyframe bank preserves task-relevant events beyond the recent observation window, while predictive coding makes the high-level VLM more sensitive to physical state transitions. This hierarchical memory input and precise subtask management bring the highest TSR and CSR among all methods.

End-to-end Training vs. Closed-Source VLMs as S2. All high-level planners share the same prompt, subtask interface, and S1 executor (Appendix I.1). Table 2(b) shows that closed-source VLMs fall far below trainable baselines: even the state-of-the-art GPT-5.4 (OpenAI, 2026) achieves

Table 2: **Comparison on RoboMemArena.** Per-category TSR/CSR (%) and overall averages, as means over three training seeds (standard deviations in Appendix I). Ground Truth feeds ground-truth subtasks to the same S1. Ground Truth shown in gray as an oracle reference.

Method	Transferring		Occlusion		Counting		Sequence		Average	
	TSR	CSR	TSR	CSR	TSR	CSR	TSR	CSR	TSR	CSR
(a) Baselines										
$\pi_{0.5}$	20.0	42.8	12.7	17.2	14.3	50.9	60.0	71.6	21.5	38.6
HiF-VLA	17.5	38.9	12.7	27.1	8.6	45.9	42.5	70.2	16.9	40.6
MemoryVLA	15.0	37.2	7.3	13.1	14.3	55.1	37.5	65.2	15.0	36.1
MemER	20.0	36.1	16.4	33.2	27.1	65.1	65.0	79.1	27.3	49.3
(b) Frozen / oracle references										
Qwen3-VL-8B (frozen)	15.0	34.6	0.0	6.8	9.3	44.6	7.5	39.2	6.0	26.2
GPT-5.4	13.8	32.9	1.8	9.2	12.9	50.7	15.0	47.3	8.7	29.9
Ground Truth	32.5	54.8	33.6	49.8	51.4	75.6	85.0	92.3	46.1	64.1
(c) Component variants of PrediMem										
w/o Predictive Coding Head	25.0	43.7	19.5	30.2	38.6	61.8	63.8	80.7	32.3	48.6
w/o Keyframe Bank	17.5	33.3	6.4	22.8	20.0	61.7	45.0	66.3	17.7	41.6
(d) Backbone variants of PrediMem										
w/ Qwen3-1.7B	15.0	31.6	7.3	20.8	28.6	60.2	50.0	73.9	20.8	41.2
w/ Qwen3-4B	20.0	42.1	18.2	34.7	38.6	64.9	65.0	84.6	31.2	51.6
PrediMem (Ours)	**22.5**	**45.2**	**27.3**	**38.4**	**45.7**	**69.3**	**72.5**	**89.5**	**38.5**	**55.6**

only 8.7% TSR. We attribute this gap to limited memory generalization to unseen robotic scenarios and an insufficient understanding of physical actions in VLMs trained primarily on vision-language tasks. These results suggest that VLMs must be trained on robotic data to serve effectively as memory management modules in robotic systems.

Decoupled Planner–Executor Diagnostics. To disentangle high-level planning from low-level execution, the Ground Truth row replaces S2 outputs with ground-truth subtasks while retaining S1. Its 46.1% TSR characterizes end-to-end performance under oracle high-level guidance (Appendix I.2). We separately evaluate S2 on 500 held-out successful trajectories without invoking low-level control. On history-dependent subtasks, PrediMem achieves 80.2% frame-wise active-subtask agreement, versus 74.5% for MemER and 32.6% for GPT-5.4 (Table 10). This cleanly separates both components.

5.3 Ablation Studies (Q3)

Predictive Coding. Table 2(c) shows the ablation results of predictive coding and the keyframe bank. For transferring tasks, removing the predictive coding head has a limited effect because its relevant state changes are direct. In contrast, predictive coding becomes more important for occlusion, counting, and sequence tasks, where the model must detect subtle state transitions such as drawer closure, object disappearance, repeated pouring, or the completion of an ordered step. By predicting future visual representations during training, predictive coding makes the hidden states more sensitive to these transition points and can recover discriminative cues beyond the sparsely annotated keyframes, mitigating imperfections in both keyframe labeling and SFT supervision.

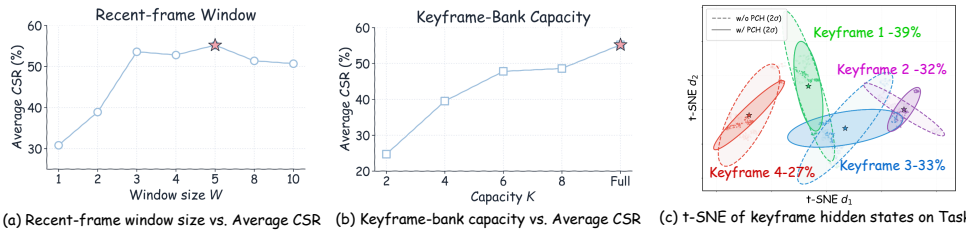
Loss Weights. A coefficient is introduced to balance the predictive loss $\mathcal{L}_{\text{Pre}}$ and the instruction-tuning loss $\mathcal{L}_{\text{text}}$. As shown in Table 3, our ablation study indicates that 0.1 yields the best performance.

Table 3: **Ablation of the $\mathcal{L}_{\text{Pre}}$ weight.**

Weights	0.0	0.1	0.5	1
TSR	32.3%	**38.5%**	31.0%	29.8%

Keyframe Bank. Removing the keyframe bank also causes a broader drop because task-relevant events are no longer preserved once they leave the recent-frame window. For occlusion and sequence tasks, the correct decision often depends on earlier placements, inspections, or ordering decisions that are no longer visible. Thus, explicit keyframe memory is necessary for accurate high-level planning.

Visualization of Keyframe Representation. Figure 5(c) shows a t-SNE view of keyframe-related hidden states on Task 1 (Appendix I.4). Without predictive coding, keyframe classes overlap strongly; with it, each class becomes more compact and the classes separate, indicating more discriminative keyframe representations.



(a) Recent-frame window size vs. Average CSR (b) Keyframe-bank capacity vs. Average CSR (c) t-SNE of keyframe hidden states on Task~1.

Figure 5: **Analyses of memory behavior.** (a) and (b) shows the sensitivity of average CSR to the recent-buffer size and keyframe-bank capacity. (c) is a t-SNE view showing that predictive coding yields tighter, more discriminative keyframe clusters.

5.4 Scaling Studies (Q4)

Scaling of Memory Bank. Figure 5(a,b) summarizes how performance changes as the recent-frame buffer and keyframe buffer are varied. With only one or two recent frames, S2 lacks enough temporal evidence to detect state transitions and select keyframes reliably. A 3–5 frame window is sufficient for most short-term changes, while larger windows add redundant visual context, increase VLM latency, and make S2 refreshes less synchronized with the low-level VLA execution loop. For the keyframe bank, CSR is very low with only two stored keyframes because early observations are quickly evicted in long-horizon tasks. Increasing the capacity to 4–8 improves performance, and the best result is obtained with the uncapped keyframe bank, which preserves early decision-critical observations such as the first drawer state after later drawers have been inspected.

Scaling of S2. In Table 2(d), we train and evaluate our method using Qwen3 backbones with different parameters. Under controlled settings with matched pretraining data and model architecture, we find that scaling up the S2 model consistently improves performance across all four task categories. This trend suggests that larger models provide stronger reasoning and memory capabilities, which are particularly beneficial for our benchmark.

5.5 Real-World Experiments (Q5)

We compare PrediMem against $\pi_{0.5}$ and MemER on the eight real-world memory tasks introduced in Section 3.1, with 50 rollouts per task (Table 4). $\pi_{0.5}$ reaches only 23.8% average success because it selects actions from the current frame alone and cannot reliably use previous counts, hidden target locations, or demonstrated action order. MemER improves the average to 41.8%, benefiting from the high-level memory signal provided by retrieved keyframes. PrediMem achieves the highest average success rate (51.0%) and the best result on six of the eight tasks; MemER is better on Shell Game and Wash Target Fruit. IHMB, which requires reproducing a full breakfast-preparation sequence from a human demonstration, remains difficult for all methods (at most 6%); excluding it, PrediMem averages 57.4% on the other seven tasks.

Table 4: **Real-world success rates (%)** over 50 rollouts per task (tasks in Table 8).

Task	$\pi_{0.5}$	MemER	PrediMem
Pour×2	28	22	**62**
Brush	14	46	**60**
Transfer	56	80	**88**
Shell	12	**52**	46
IHMB	0	4	**6**
Drawer	14	36	**48**
Wash Fruit	28	**40**	38
Press×2	38	54	**60**
Average	23.8	41.8	**51.0**

6 Conclusion

We presented *RoboMemArena*, a diverse and challenging robotic memory benchmark that combines native keyframe-centered multimodal supervision, scalable long-horizon trajectory generation, and paired real-world memory evaluation. Across the benchmark, 68.9% of subtasks depend on past observations, making memory a central requirement rather than an optional enhancement. We also introduced **PrediMem**, a dual-system memory framework whose predictive coding objective makes hidden states more sensitive to physical state transitions without adding inference-time cost.

AI USE STATEMENT

Generative AI tools are part of the benchmark construction. A VLM proposes the initial subtask decompositions for data generation, which human annotators review and refine before execution (Section 3.2 and Appendix A), and an LLM performs a first-pass labeling of memory-dependent subtasks for the larger benchmarks we compare against, with ambiguous cases checked manually (Appendix D). GPT-5.4 and Qwen3-VL are also evaluated as high-level planners (Section 5.2). In addition, we used LLM-based assistants for language editing and for LaTeX and reference formatting. The authors reviewed all AI-assisted outputs and take full responsibility for the content of this paper.

REPRODUCIBILITY STATEMENT

Section 3.2 describes the data generation pipeline, and Appendices A and B give the prompts used for data generation and for training S2. Section 3.4 defines the evaluation metrics, Appendix D specifies how the memory-dependent subtask ratio is computed, and Appendices E and G list all simulated and real-world tasks. Training details of PrediMem are given in Section 5.1, its asynchronous inference protocol in Appendix C, and seeds, rollout counts, control rates, and real-world data collection in Appendices G and I. The dataset and code are available at https://huggingface.co/datasets/4tvBXBHgz/RoboMemArena and https://anonymous.4open.science/r/RoboMemArena-anonymous-C544.

REFERENCES

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, et al. Motus: A unified latent action world model. *arXiv preprint arXiv:2512.13030*, 2025.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, brian ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *9th Annual Conference on Robot Learning*, 2025.

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*, 2026a.

Yipeng Chen et al. Non-markovian long-horizon robot manipulation via keyframe chaining. *arXiv preprint arXiv:2603.01465*, 2026b.

Egor Cherepanov, Nikita Kachaev, Alexey Kovalev, and Aleksandr Panov. Memory, benchmark & robots: A benchmark for solving complex tasks with reinforcement learning. *The Fourteenth International Conference on Learning Representations*, 2026.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11), 2025.

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.

Hao-Shu Fang, Chenxi Wang, Hongjie Fang, Minghao Gou, Jirong Liu, Hengxu Yan, Wenhai Liu, Yichen Xie, and Cewu Lu. Anygrasp: Robust and efficient grasp perception in spatial and temporal domains. *IEEE Transactions on Robotics*, 39(5), 2023. doi: 10.1109/TRO.2023.3281153.

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. *Forty-second International Conference on Machine Learning*, 2025.

Ran Gong, Jiangyong Huang, Yizhou Zhao, Haoran Geng, Xiaofeng Gao, Qingyang Wu, Wensi Ai, Ziheng Zhou, Demetri Terzopoulos, Song-Chun Zhu, Baoxiong Jia, and Siyuan Huang. Arnold: A benchmark for language-grounded task learning with continuous states in realistic 3d scenes. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. doi: 10.1109/ICCV51070.2023.01873.

Songhao Han, Boxiang Qiu, Yue Liao, Siyuan Huang, Chen Gao, Shuicheng Yan, and Si Liu. Robocerebra: A large-scale benchmark for long-horizon robotic manipulation evaluation. *arXiv preprint arXiv:2506.06677*, 2025.

Qingda Hu, Ziheng Qiu, Zijun Xu, Kaizhao Zhang, Xizhou Bu, Zuolei Sun, Bo Zhang, Jieru Zhao, Zhongxue Gan, and Wenchao Ding. Resolving state ambiguity in robot manipulation via adaptive working memory recoding. *arXiv preprint arXiv:2512.24638*, 2025.

Yutong Hu, Jan-Nico Zaech, Nikolay Nikolov, Yuanqi Yao, Sombit Dey, Giuliano Albanese, Renaud Detry, Luc Van Gool, and Danda Paudel. Ar-vla: True autoregressive action expert for vision-language-action models. *arXiv preprint arXiv:2603.10126*, 2026.

Physical Intelligence, Ali Amin, et al. $\pi^*_{0.6}$: a vla that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025.

Physical Intelligence, Bo Ai, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Greg Balke, Kevin Black, George Bokinsky, Shihao Cao, Thomas Charbonnier, et al. $\pi_{0.7}$: A steerable generalist robotic foundation model with emergent capabilities. *arXiv preprint arXiv:2604.15483*, 2026.

Stephen James and Pieter Abbeel. Coarse-to-fine q-attention with learned path ranking. In *arXiv preprint arXiv:2204.01571*, 2022.

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), 2020. doi: 10.1109/LRA.2020.2974707.

Huiwon Jang, Sihyun Yu, Heeseung Kwon, Hojin Jeon, Younggyo Seo, and Jinwoo Shin. Contextvla: Vision-language-action model with amortized multi-frame context. *arXiv preprint arXiv:2510.04246*, 2025.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, et al. Openvla: An open-source vision-language-action model. *Conference on Robot Learning*, 2025.

Myungkyu Koo, Daewon Choi, Taeyoung Kim, Kyungmin Lee, Changyeon Kim, Younggyo Seo, and Jinwoo Shin. Hamlet: Switch your vision-language-action model into a history-aware policy. *arXiv preprint arXiv:2510.00695*, 2025.

Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, et al. Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation. *Conference on Robot Learning*, 2023.

Hao Li, Shuai Yang, Yilun Chen, Xinyi Chen, Xiaoda Yang, Yang Tian, Hanqing Wang, Tai Wang, Dahua Lin, Feng Zhao, et al. Cronusvla: Towards efficient and robust manipulation via multi-frame vision-language-action modeling. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(22), 2026a.

Ji Li, Bo Wang, Jing Xia, Mingyi Li, and Shiyan Hu. Himm: Human-inspired long-term memory modeling for embodied exploration and question answering. *arXiv preprint arXiv:2602.15513*, 2026b.

Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, et al. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024.

Runhao Li, Wenkai Guo, Zhenyu Wu, Changyuan Wang, Haoyuan Deng, Zhenyu Weng, Yap-Peng Tan, and Ziwei Wang. Map-vla: Memory-augmented prompting for vision-language-action model in robotic manipulation. *arXiv preprint arXiv:2511.09516*, 2025.

Ruoran Li, Xinghua Zhang, Haiyang Yu, Shitong Duan, Xiang Li, Wenxin Xiang, Chonghua Liao, Xudong Guo, Yongbin Li, and Jinli Suo. Mempo: Self-memory policy optimization for long-horizon agents. *arXiv preprint arXiv:2603.00680*, 2026c.

Zaijing Li, Bing Hu, Rui Shao, Gongwei Chen, Dongmei Jiang, Pengwei Xie, Jianye Hao, and Liqiang Nie. Global prior meets local consistency: Dual-memory augmented vision-language-action model for efficient robotic manipulation. *arXiv preprint arXiv:2602.20200*, 2026d.

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, 2023.

Zhuoyang Liu, Jiaming Liu, Hao Chen, Jiale Yu, Ziyu Guo, Chengkai Hou, Chenyang Gu, Xiangju Mi, Renrui Zhang, Kun Wu, et al. Last _{0}: Latent spatio-temporal chain-of-thought for robotic vision-language-action model. *arXiv preprint arXiv:2601.05248*, 2026.

Haoran Lu, Ruihai Wu, Yitong Li, Sijie Li, Ziyu Zhu, Chuanruo Ning, Yan Shen, Longzan Luo, Yuanpei Chen, and Hao Dong. Garmentlab: a unified simulation and benchmark for garment manipulation. *Proceedings of the 38th International Conference on Neural Information Processing Systems*, 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), 2022. doi: 10.1109/LRA.2022.3180108.

Yao Mu, Tianxing Chen, Shijia Peng, Zanxin Chen, Zeyu Gao, Yude Zou, Lunkai Lin, Zhiqiang Xie, and Ping Luo. Robotwin: Dual-arm robot benchmark with generative digital twins (early version). *European Conference on Computer Vision*, 2024.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. *arXiv preprint arXiv:2406.02523*, 2024.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. *Proceedings of Robotics: Science and Systems*, 2024.

OpenAI. Introducing gpt-5.4. *OpenAI Blog. https://openai.com/index/introducing-gpt-5-4*, 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*, 2025.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020.

Ajay Sridhar, Jennifer Pan, Satvik Sharma, and Chelsea Finn. Memer: Scaling up memory for robotic control via experience retrieval. *The Fourteenth International Conference on Learning Representations*, 2026.

Jun Sun, Boyu Yang, Jiahao Zhang, Ning Ma, Chencheng Wu, Siqing Zhang, Yiou Huang, Qiufeng Wang, Shan Liang, and Yaran Chen. Tempofit: Plug-and-play layer-wise temporal kv memory for long-horizon vision-language-action manipulation. *arXiv preprint arXiv:2603.07647*, 2026.

Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse-kai Chan, et al. Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai. *arXiv preprint arXiv:2410.00425*, 2024.

Marcel Torne, Karl Pertsch, Homer Walke, Kyle Vedder, Suraj Nair, Brian Ichter, Allen Z Ren, Haohuan Wang, Jiaming Tang, Kyle Stachowicz, et al. Mem: Multi-scale embodied memory for vision language action models. *arXiv preprint arXiv:2603.03596*, 2026.

Yuran Wang, Ruihai Wu, Yue Chen, Jiarui Wang, Jiaqi Liang, Ziyu Zhu, Haoran Geng, Jitendra Malik, Pieter Abbeel, and Hao Dong. Dexgarmentlab: Dexterous garment manipulation environment with generalizable policy. *arXiv preprint arXiv:2505.11032*, 2025.

Zhenan Wang, Yanzhe Wang, Meixuan Ren, Peng Li, Yang Liu, Yifei Nie, Limin Long, Yun Ye, Xiaofeng Wang, Zhen Zhu, et al. Tacmamba: A tactile history compression adapter bridging fast reflexes and slow vla reasoning. *arXiv preprint arXiv:2603.01700*, 2026.

Junjie Wen, Yichen Zhu, Jinming Li, Zhibin Tang, Chaomin Shen, and Feifei Feng. Dexvla: Vision-language model with plug-in diffusion expert for general robot control. *Conference on Robot Learning*, 2025.

Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. Sapien: A simulated part-based interactive environment. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 2020.

Shusheng Yang, Jihan Yang, Pinzhi Huang, Ellis Brown, Zihao Yang, Yue Yu, Shengbang Tong, Zihan Zheng, Yifan Xu, Muhan Wang, et al. Cambrian-s: Towards spatial supersensing in video. *arXiv preprint arXiv:2511.04670*, 2025.

Mingfeng Yuan, Hao Zhang, Mahan Mohammadi, Runhao Li, Jinjun Shan, and Steven L. Waslander. Star: Scalable task-conditioned retrieval for long-horizon multi-modal robot memory. *IEEE Robotics and Automation Letters*, 11(5), 2026. doi: 10.1109/LRA.2026.3677723.

Shiduo Zhang, Zhe Xu, Peiju Liu, Xiaopeng Yu, Yuan Li, Qinghui Gao, Zhaoye Fei, Zhangyue Yin, Zuxuan Wu, Yu-Gang Jiang, et al. Vlabench: A large-scale benchmark for language-conditioned robotics manipulation with long-horizon reasoning tasks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 11142–11152, 2025.

Tony Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *Robotics: Science and Systems XIX*, 2023.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, et al. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. *arXiv preprint arXiv:2510.10274*, 2025.

A VLM INPUT PROMPT FOR DATA GENERATION

We use a VLM input prompt template to generate long-horizon robot data by decomposing a coarse task description into executable subtasks and matching each subtask to a predefined planner. Below we show a representative input-only example.

System Prompt.

```
You are a robotic data-generation assistant for long-horizon,
memory-dependent manipulation tasks.

You will be given:
1. One RGB image of the current environment.
2. A coarse long-horizon task description.

Your job is to:
- decompose the task into an ordered list of executable subtasks;
- assign each subtask to one predefined planner from
  {Move, Place, Pour, Open, Close};
- keep the subtasks grounded in the visible scene;
- preserve memory-dependent structure when later steps depend on earlier
  placements, occluded objects, or counted actions;
- return strict JSON only.

Return exactly one field:
- subtasks: a list of ordered subtask entries, each with
  step_id, subtask, planner, and target.
```

User Prompt.

```
Current environment image:
<image>

Long-horizon task description:
Pour tomato sauce over the cookies and heat them, then pour milk into a cup.

Please decompose this task into executable subtasks and assign the
predefined planner to each step.
```

B VLM TRAINING PROMPT AND JSON FORMAT

We train S2 with a fixed multi-image prompt template. Below we show a representative prompt template for one task.

System Prompt.

```
You are a robotic planning assistant specialized in memory-based task
understanding.

Your task is to infer the current primitive action from multi-image
observation:
1. Historical keyframes from earlier in the same trajectory.
2. A recent 5-timestep visual window ending at the current frame.

Important rules:
- Historical keyframes are earlier than the current window.
- Each timestep contains two images: one agentview_rgb image and one
  eye_in_hand_rgb image.
- The recent 5-timestep window is the primary evidence for current action.
- If there is no keyframe inside the current window, keyframe_positions
  must be an empty list.
- Return strict JSON only. Do not output extra text.
- Return exactly two fields:
  - current_primitive: one primitive from the predefined task1 primitive set.
  - keyframe_positions: 1-indexed keyframe positions inside the recent
    5-timestep window.
```

User Prompt.

```
Global task: Infer the current primitive action from recent visual history
for a task that stores the cookies and the tomato sauce into the same
target container.

Scene description:
- The manipulation happens on a tabletop.
- The target container is the basket on the right side of the table.
```

```
- The square object near the middle is the cookies item.
- The cylindrical object near the middle is the tomato sauce container.
- Each timestep contains two images: one agentview_rgb image and one
  eye_in_hand_rgb image.
- Camera order for every timestep: agentview_rgb, eye_in_hand_rgb.

Current observation:
Recent visual context: 5 consecutive timesteps ending at the current frame
(10 images total; two per timestep, ordered as agentview_rgb followed by
eye_in_hand_rgb):
<image> <image> <image> <image> <image>
<image> <image> <image> <image> <image>

Return strict JSON with fields current_primitive and keyframe_positions
(1-indexed positions inside the recent 5-timestep context).
```

Assistant Label JSON.

```
{"current_primitive":"pick cookies","keyframe_positions":[]}
```

Top-Level JSONL Format.

```
{
  "qid": "seed100_order0_win0_r0",
  "messages": [
    {"role": "system", "content": "..."},
    {"role": "user", "content": "..."},
    {"role": "assistant", "content":
      "{\"current_primitive\":\"pick cookies\",\"keyframe_positions\":[]}"}
  ],
  "images": ["...", "... x10"],
  "metadata": {
    "task_id": 1,
    "prompt_style": "breakfast_like",
    "current_primitive": "pick cookies",
    "keyframe_positions": [],
    "camera_keys": ["agentview_rgb", "eye_in_hand_rgb"],
    "history_keyframe_count": 0,
    "num_context_frames": 5,
    "num_context_images": 10,
    "image_size": [256, 256],
    "...": "other window/index fields"
  }
}
```

C Asynchronous Inference Protocol

Given instruction ℓ and observation o_0, S2 runs asynchronously on the recent-frame buffer and memory bank $\mathcal{M}_t$, emitting subtask c_t and keyframe decision k_t; the buffered subtask is overwritten whenever a newer S2 result arrives. S1 takes the current observation o_t together with the latest subtask c_t at the higher control rate and produces an action chunk

$$a_t = \pi_{\mathrm{S1}}(o_t,\, c_t), \tag{5}$$

where π_{S1} denotes the S1 low-level policy (a VLA action head), o_t is the current visual observation, c_t is the freshest subtask prediction produced by S2 (overwritten whenever a newer S2 result arrives), and a_t is the action chunk executed in the next control window. In our implementation, S2 runs at 1.06 Hz and S1 at 3.40 Hz, so each S2 update overlaps roughly 3.21 S1 chunks. Because $\mathcal{M}_t$ retains prior events, the agent can recover without restarting. Table 5 shows the detailed runtime profile of PrediMem's asynchronous inference loop.

PrediMem Inference Protocol

Input: Task instruction ℓ, initial observation o_0
Output: Action sequence $\{a_1, \ldots, a_T\}$

$\mathcal{M}_0 \leftarrow \emptyset$; $g \leftarrow \varnothing$
Initialize recent-frame buffer with o_0
for $t = 1$ to T **do**
  $a_t \leftarrow \pi_{\mathrm{S1}}(o_t, g)$ *// High-frequency execution loop*
  if S2 is idle and the recent window is ready **then**
    Trigger S2$(\ell, o_t, \mathcal{M}_t^{rec}, \mathcal{M}_t^{key})$ asynchronously *// Predict latest subtask*
  end if
  if S2 result $(g_{\mathrm{new}}, k_\tau)$ is available **then**
    $g \leftarrow g_{\mathrm{new}}$ *// Refresh current subtask*
    if $k_\tau = 1$ **then** $\mathcal{M}_t^{\mathrm{key}} \leftarrow \mathcal{M}_t^{\mathrm{key}} \cup \{o_\tau\}$ **end if** *// Memory write*
  end if
  $\mathcal{M}_t^{\mathrm{rec}} \leftarrow$ last W frames *// Update recent sliding window*
end for

Figure 6: **Inference protocol of PrediMem.** S2 asynchronously predicts the latest subtask and keyframe decisions from recent visual history, while S1 executes precise actions at a high control frequency under the freshest available subtask.

Table 5: **Runtime profile of the asynchronous inference loop.** Runtime measurements of our asynchronous implementation.

Metric	PrediMem asynchronous implementation
High-level VLM refresh rate	1.06 Hz steady state (p50 0.939 s)
High-level VLM latency	p50 0.939 s; p95 1.136 s; mean 1.752 s including cold start
Low-level VLA frequency	3.40 Hz (mean 0.294 s; p50 0.289 s; p95 0.365 s)
VLM:VLA scheduling	one VLM update spans ∼3.21 VLA chunks

D Memory-Dependent Subtask Ratio Annotation

We define a subtask as *memory-dependent* when the correct current subtask cannot be determined from the current observation alone and requires more information from earlier subtasks or observations. Equivalently, a subtask is counted as memory-dependent if removing the execution history or observation would make the correct high-level decision ambiguous. For task i with n_i subtasks and m_i memory-dependent subtasks, we compute the task-level ratio as

$$r_i = \frac{m_i}{n_i}. \tag{6}$$

The benchmark-level memory-dependent subtask ratio is computed over all subtasks:

$$R_{\mathrm{mem}} = \frac{\sum_i m_i}{\sum_i n_i}. \tag{7}$$

For RoboMemArena, this gives $R_{\mathrm{mem}} = 104/151 = 68.9\%$, where 104 denotes the total number of memory-dependent subtasks across all tasks, and 151 denotes the total number of subtasks by summing all tasks.

Our definition encompasses these **four common forms** of memory demand, but is not limited to them. **Occlusion** requires remembering the location of a target object after it becomes hidden inside a container, for example, when the robot must place another object into the same container. **Counting** requires remembering how many times an action has already been performed, such as whether it has already been poured once and needs to be poured again. **Transferring** requires remembering the source–target mapping between containers that are visually similar. **Sequence** requires remembering which prerequisite subtasks have already been completed before executing the next step. For example, in a drawer-search task, the first drawer opening may not require memory, whereas later subtasks should depend on which drawers have already been inspected and what was observed in each drawer. In this case, if 7 out of 9 subtasks require such history-dependent information, the memory ratio is $7/9$.

For benchmarks with a small number of tasks, we manually inspect task descriptions, keyframe annotation and subtask decompositions. For larger benchmarks, we use an LLM-assisted first pass with a fixed rubric and then manually check the ambiguous cases. The simplified rubric is:

```
We classify each subtask as memory-dependent or memory-free according to
whether it requires information from earlier observations or task states.

A subtask is memory-free if the required object and target state are visible
in the current observation, e.g., picking up a visible cup or placing a
visible cup on the table.

A subtask is memory-dependent if its execution depends on earlier task
state or observation, including but not limited to:
1. Occlusion: Remembering that an object was placed into an occlusive container.
2. Counting: Remembering how many times an action has already been performed.
3. Transferring: Remembering the source-target mapping between similar containers.
4. Sequence: Remembering which prerequisite subtasks have already been completed.
```

E BENCHMARK TASK DETAILS

Table 6: **Benchmark task descriptions.** Overview of all 26 RoboMemArena tasks, their corresponding memory types, average total timesteps, and key challenges.

Task Name	Memory Type	Avg. #Steps	Task Challenge	Brief Description
`Task Suite: Multi-Object Transferring`				
Transfer Chocolate Butter	T	866	transferring	Pick and place chocolate and butter from plate1 to plate2, respectively.
Transfer Butter Cheese	T	779	transferring	Pick and place butter and cheese from plate1 to plate2, respectively.
Transfer Popcorn Cookies	T	779	transferring	Pick and place popcorn and cookies from plate1 to plate2, respectively.
Transfer Sauce Milk Juice	T	1265	transferring	Pick and place tomato sauce, milk, and orange juice from cabinet1 to cabinet2.
`Task Suite: Multi-Object Occlusion`				
Put Butter in Not-Empty Drawer	O	1020	occlusion	Open all drawers in order. Put butter into the drawer that already contains an object.
Put Butter in Empty Drawer	O	1806	occlusion	Open all drawers in order. Put butter into the empty drawer.
Put Cookies Butter into Drawer Respectively	O + C	1835	occlusion, multi-placement	Open all drawers in order. Put cookies into the top drawer and put butter into another drawer.
Put Cookies Chocolate into Middle Drawer	O	1370	occlusion	Open all drawers in order. Put cookies into the middle drawer and then put chocolate into the same drawer.
Put Butter Cookies into Middle Drawer	O	1377	occlusion	Open all drawers in order. Put butter into the middle drawer and then put cookies into the same drawer.
Put Cookies Chocolate into Drawer Respectively	O	1832	occlusion, multi-placement	Open all drawers in order. Put cookies into the top drawer and put chocolate into another drawer.
Put Butter Chocolate into Middle Drawer	O	1502	occlusion	Open all drawers in order. Put butter into the middle drawer and then put chocolate into the same drawer.
Put Cookies Chocolate into Microwave	O	1195	occlusion	Put cookies into the microwave and then put chocolate into the location where the cookies were placed.

Continued on next page...

Table 6: **Benchmark task descriptions.** (Continued)

Task Name	Memory Type	Avg. #Steps	Task Challenge	Brief Description
Put Butter Chocolate into Microwave	O	1175	occlusion	Put butter into the microwave and then put chocolate into the location where the butter was placed.
Put Cream Popcorn into Microwave	O	1175	occlusion	Put cream into the microwave and then put popcorn into the location where the cream was placed.
Put Cookies Popcorn into Microwave	O	1195	occlusion	Put cookies into the microwave and then put popcorn into the location where the cookies were placed.
`Task Suite: Multi-Object Counting`				
Pour Sauce on Cookies $\times 2$ Place Sauce into Drainer	C + O	624	counting, occlusion	Pour tomato sauce over cookies twice and place the sauce bottle into the bowl drainer.
Pour Sauce on Frypan $\times 2$ Place Sauce into Drainer	C + O	537	counting, occlusion	Pour tomato sauce over the frypan twice and place the sauce bottle into the bowl drainer.
Pour Sauce Twice over Chocolate in Frypan Place Sauce into Drainer	C	910	counting	Pick and place chocolate into the frypan, pour tomato sauce over it twice, then place the sauce bottle into the bowl drainer.
Pour Sauce $\times 2$ over Butter in Frypan	C	958	counting	Put butter into the frypan and pour sauce over it twice.
Pour Wine into Mug Twice	C	472	counting	Pour wine into the mug twice.
Pour Milk Twice over Butter in Frypan	C	1055	counting	Pick and place butter into the frypan, then pour milk over it twice.
Pour Milk $\times 2$ over Mug Place Milk into Drainer	C + O	594	counting, occlusion	Pick milk from the table, pour it into the mug twice, then place the milk container into the bowl drainer.
`Task Suite: Multi-Object Sequence`				
Put Cookies Sauce into Basket in Order	S	742	sequence	Pick and place cookies into the basket, then pick and place tomato sauce into the same basket.
Put Butter Popcorn into Basket in Order	S	708	sequence	Pick and place butter into the basket, then pick and place popcorn into the same basket.
Put Cream Chocolate into Basket in Order	S	708	sequence	Pick and place cream into the basket, then pick and place chocolate into the same basket.
Pour Sauce $\times 2$ Put Cookies into Microwave	S	1565	sequence	Pour tomato sauce over cookies twice, then put the cookies into the microwave.

F Reactive Policy Failure Modes

The quantitative gap between reactive and memory-augmented policies maps to two concrete failures. First, the reactive policy cannot distinguish whether a drawer has already been checked once the visual state resets. Second, it cannot preserve the instruction-level constraint that all drawers must be opened before the final placement. Detailed failure modes are provided in Table 7.

Table 7: **Failure modes of the reactive $\pi_{0.5}$ baseline on a memory-dependent drawer task.**

Failure mode	Description
State aliasing	After the first drawer is closed, the observation becomes visually similar to the initial frame. Without memory, the policy repeatedly opens the same drawer and enters a loop.
Constraint forgetting	After finding one locally valid empty drawer, the policy places the butter immediately and forgets the global instruction to inspect all drawers before deciding.

G Real-World Task Details

Data collection and evaluation. Real-world demonstrations are collected with the AgileX Cobot leader–follower teleoperation system, which consists of two leader arms and two follower arms. A human operator moves the leader arms, and the follower arms track their joint commands in real time; the system records visual observations, follower-arm states, and teleoperation actions at 30 Hz. We collect 100 successful demonstrations per task and train all methods with the settings in Section 5.1. Each task is evaluated with 50 independent rollouts (Table 4). Drawer Memory, Wash Target Fruit, and Press Button Twice extend the suite with object-state, target-identity, and action-history memory, respectively.

Table 8: **Real-world task descriptions.** Overview of the eight physical-robot tasks, their corresponding memory types, average total timesteps, and key challenges.

Task Name	Memory Type	Avg. #Steps	Task Challenge	Brief Description
`Task Suite: Real-World Evaluation`				
Pour Bottle ×2	C	866	counting	Pour water from the bottle into the cup twice.
Brush Plates with Swap	C + S	779	sequence	Brush three plates in order.
Transfer Objects	T	779	transferring	Transfer the watermelon and the carrot from one plate to another.
Shell Game	O + S	779	occlusion, tracking	Hide the target under one cup, swap the positions of the three cups, and identify the cup containing the target.
Imitate Human to Make Breakfast (IHMB)	C + S	7265	imitation, sequence	A human demonstrates how to make breakfast, and the robot imitates the demonstrated breakfast-making sequence.
Drawer Memory	O	1024	object state	Remember which drawer already contains an object.
Wash Target Fruit	O	2101	target identity	Remember the previously specified fruit that should be washed.
Press Button Twice	C	604	counting	Remember how many times the button has already been pressed.

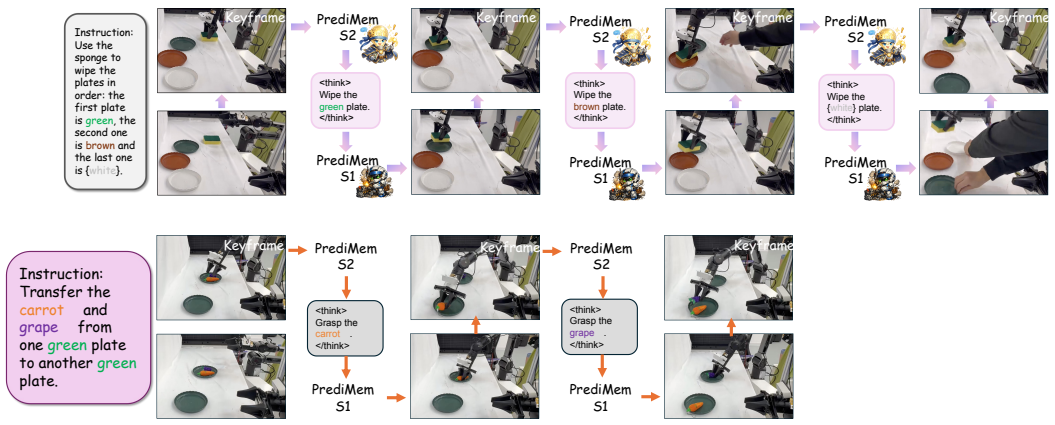


Figure 7: **Real-world task demonstrations.** Representative snapshots from our physical-robot evaluation suite. The figure summarizes the real-world task settings, example execution frames, and the dual-arm platform layout in a format similar to standard real-robot evaluation overviews.

H VERIFICATION-STEP DISTRIBUTION

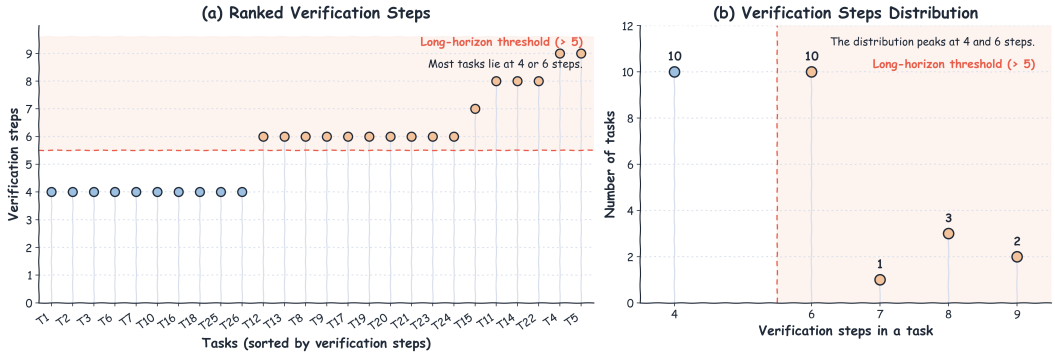


Figure 8: **Distribution of verification steps per task. (a)** Each task contains 4–9 verification steps. Most exceed the long-horizon threshold of five steps. **(b)** Overall histogram of verification step counts. The distribution ensures that the cumulative success rate (CSR) metric has sufficient resolution to distinguish agents with differing levels of memory capability.

I ADDITIONAL EVALUATION DETAILS

Protocol and statistics. Simulated observations and actions are recorded at 20 Hz. Each method is trained with three independent seeds and evaluated with 100 independent rollouts per simulated task, and Table 2 reports means across seeds. The cross-seed standard deviations of the average TSR are 2.9, 3.4, 3.2, 4.3, and 3.9 points for $\pi_{0.5}$, HiF-VLA, MemoryVLA, MemER, and PrediMem, respectively. Because step counts depend on the control rate, Table 9 also reports task durations in seconds.

Table 9: **Control rate and task duration of robotic memory benchmarks.** Steps are counted at each benchmark's native control rate. MIKASA refers to the RL suite MIKASA-Robo (Cherepanov et al., 2026); the later MIKASA-Robo-VLA suite, which adds short-, medium-, and long-horizon tiers, is not included.

Benchmark	Control rate	Avg. steps / task	Avg. seconds / task
MIKASA-Robo (RL)	20 Hz	92.8	4.64
MemoryBench	10 Hz	312.1	31.21
RoboMME	20 Hz	482.4	24.12
RMBench	20 Hz	695.3	34.77
RoboMemArena (ours)	20 Hz	1,076.0	53.80

I.1 Closed-Source VLM Planners

All high-level planners, including GPT-5.4 and the frozen Qwen3-VL-8B, receive the same planner prompt with the task instruction, scene description, visual observations, primitive vocabulary, and JSON output format (Appendix B). They output subtasks through the same interface, drive the same low-level VLA executor, and follow the same evaluation protocol. The closed-source VLMs receive the same task-specific context as PrediMem.

I.2 Oracle Setting and Failure Analysis

The Ground Truth setting replaces the high-level subtasks with ground-truth subtasks while retaining the same learned VLA executor. Its TSR is therefore bounded by low-level grasping, control, and environment interaction rather than by memory. Full-task success also compounds across stages: assuming independent stage outcomes, an eight-stage task with 90% success per stage succeeds only $0.9^8 \approx 43\%$ of the time, which is why we also report CSR. To characterize the remaining failures, we randomly sampled 200 failed Ground Truth rollouts and assigned each to one mutually exclusive category according to its first incomplete stage: grasping or grasp-retention failures (37%), target-object or target-receptacle grounding failures (29%), insufficient action magnitude for drawer operations, pouring, or placing (23%), and other failures (11%).

I.3 High-Level Planner Diagnostic

To evaluate high-level memory without low-level execution errors, we use 500 held-out successful reference trajectories and annotate every frame with its ground-truth active subtask. Given only the observations up to the current frame, the high-level model predicts the active subtask; no low-level policy is executed, so grasping and control errors are excluded. Table 10 reports frame-wise agreement with the ground truth over all frames and over frames within subtasks annotated as history-dependent. Because the trajectories are successful references, this diagnostic characterizes the high-level planner only; it does not attribute the end-to-end gaps in Table 2 to memory or execution.

Table 10: **Frame-wise active-subtask agreement (%)** of high-level planners on 500 held-out successful trajectories.

Method	All subtasks	History-dependent subtasks
PrediMem (S2)	**72.3**	**80.2**
MemER	68.5	74.5
GPT-5.4	39.2	32.6

I.4 t-SNE of Keyframe Representations

For Figure 5(c), we extract final-layer hidden representations of S2 for the same input samples with and without predictive coding, aggregate token features into one embedding per sample, and project these embeddings to two dimensions with t-SNE.

J SCOPE AND TASK STRUCTURE

RoboMemArena targets task-relevant memory in long-horizon tabletop manipulation. The task suite varies in scene, action structure, subtask order, and the state that must be remembered, while object changes within a shared structure serve as controlled variants. Table 11 illustrates this design for the 11-task occlusion category.

Table 11: **Structural organization of the 11 occlusion tasks.** The tasks form six groups defined by their scene–action structure and memory requirement. Object combinations within a group serve as controlled variants rather than the sole source of task diversity.

Scene	Action	Object variants	What must be remembered
Drawers	Inspect all drawers, then place into the non-empty one	butter	Which drawer was non-empty
Drawers	Inspect all drawers, then place into the empty one	butter	Which drawer was empty
Drawers	Place two objects into different drawers	cookies + butter; cookies + chocolate	Object-to-drawer associations
Drawers	Place two objects into the same drawer	cookies + chocolate; butter + cookies; butter + chocolate	Drawer of the first object
Microwave	Place two block-like objects, reusing the first location	cookies + chocolate; butter + chocolate	First placement location
Microwave	Place objects of different shapes, reusing the first location	cream + popcorn; cookies + popcorn	First placement location