

PhysMem: Scaling Test-Time Memory for Embodied Physical Reasoning

Haoyang Li[♠] Yang You[♣] Hao Su[♠] Leonidas Guibas[♣]
[♣]Stanford University [♠]UC San Diego

Abstract—Reliable object manipulation requires understanding physical properties that vary across objects and environments. Vision-language model (VLM) planners can reason about friction and stability in general terms, but they often cannot predict how a specific ball will roll on a particular surface or which stone will provide a stable foundation without direct experience. We present **PhysMem**, a memory framework that lets a VLM robot planner learn physical principles from interaction at test time, without updating model parameters. The system records experiences, drafts candidate hypotheses, and verifies them through targeted interaction before committing validated knowledge to guide future decisions. Verification before application is the central design choice: the system tests each hypothesis against new observations instead of applying retrieved experience directly, so that prior experience does not harden into a stale rule once physical conditions change. We evaluate **PhysMem** on three real-world manipulation tasks and simulation benchmarks across four VLM backbones. On a controlled brick insertion task, principled abstraction achieves 76% success against 23% for direct experience retrieval, and real-world experiments show consistent improvement over 30-minute deployment sessions. Website: <https://phys-mem.github.io/>

I. INTRODUCTION

*For the things we have to learn
before we can do them, we learn by doing them.*

Aristotle, *Nicomachean Ethics*

Vision-language models (VLMs) can describe physical concepts such as friction, balance, and momentum [17, 11, 25]. When deployed as robot planners, however, they often struggle to predict how these principles apply to a specific situation. A VLM planner may understand friction in the abstract yet misjudge how far a ball will roll on a given surface, or recognize stability as a concept yet fail to identify which irregular stone will provide a stable foundation. This gap between declarative knowledge and physical grounding is well documented [66, 29], and its consequences compound in planning, where a single misjudgment about contact or dynamics can invalidate an entire action sequence. This paper investigates whether a VLM robot planner can acquire useful physical understanding during deployment, through its own interaction, without updating model parameters. To isolate reasoning from execution, we separate high-level planning (VLM decisions) from low-level control (motion execution), ensuring that observed improvements reflect better physical understanding rather than better motor policies.

We focus on settings where the correct strategy is hard to infer from vision alone: tasks where the relevant physical parameters are not visually apparent and cannot be inferred

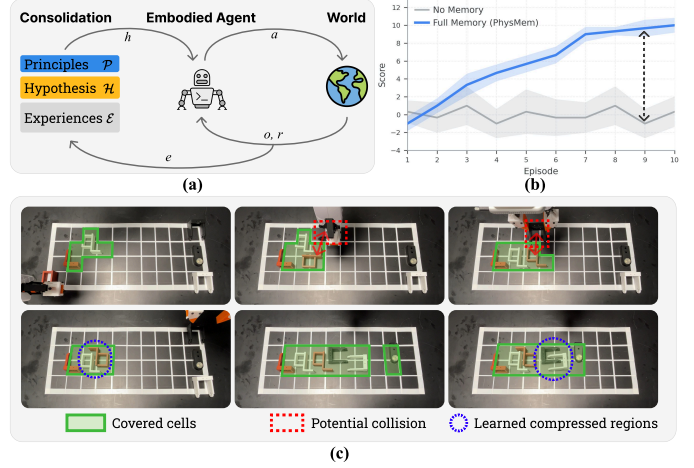


Fig. 1: **PhysMem** learns physical principles through interaction. (a) Continually Learn via the memory consolidation system maintains principles $\mathcal{P}$, hypotheses $\mathcal{H}$, and experiences $\mathcal{E}$, which guide the embodied agent’s actions a ; world feedback (observations o , rewards r) generates new experiences e that refine knowledge. (b) Test-time learning on *Parts Organization*: **PhysMem** (blue) improves continuously while no-memory baseline (gray) remains flat. (c) Qualitative results on *Parts Organization*: green boxes show covered cells, red dashed boxes indicate potential collisions avoided, and blue circles highlight learned space-saving strategies.

from pre-training data. Specifically, we consider the following 3 tasks: 1) *Parts Organization* requires learning spatial relationships between irregular shapes and available gaps; 2) *Ball Navigation* requires discovering contact dynamics through trial, since a soccer ball’s behavior under obstacles cannot be predicted from appearance; 3) *Balanced Stacking* requires stability judgments about stones whose mass distribution and surface friction only reveal themselves upon contact. Initial attempts on these tasks often fail because the planner lacks the specific physical intuition that only interaction can provide.

A natural approach is to add memory, storing experiences and retrieving them when the scene appears similar [9, 49, 51, 55]. However, embodied situations rarely repeat exactly, and retrieval-augmented approaches [34, 24, 36] apply past experience without verifying whether it still holds. Our experiments make this concrete: on a controlled brick insertion benchmark, retrieving raw experience reaches only 23% success while principled abstraction reaches 76%. The issue is that unverified memory hardens into a fixed rule, and a small change in friction or object shape then turns a useful heuristic into a

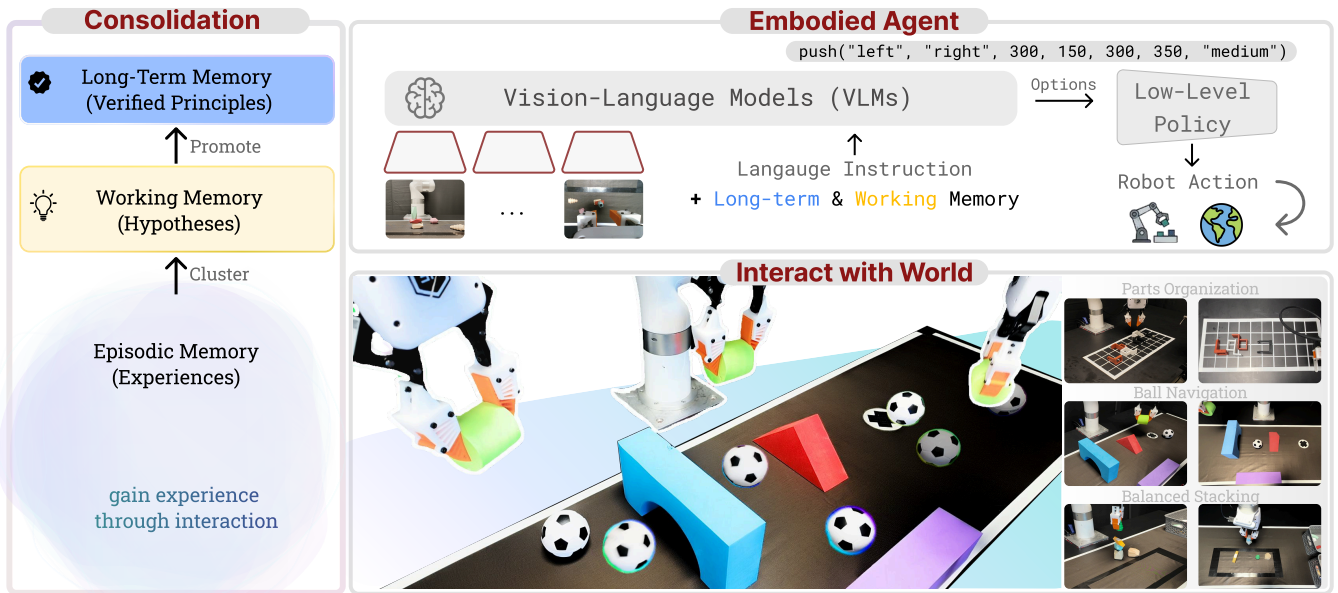


Fig. 2: **System overview of PhysMem.** **Left (Consolidation):** A three-tier memory system stores raw experiences in episodic memory, clusters them into testable hypotheses in working memory, and promotes verified knowledge to long-term memory as principles. The consolidation process continuously refines memory through interaction. **Top-right (Embodied Agent):** A Vision-Language Model receives language instructions along with retrieved principles and active hypotheses from memory, then outputs high-level decisions that are executed by a low-level policy. **Bottom-right (World Interaction):** The agent interacts with challenging physical tasks (Parts Organization, Ball Navigation, and Balanced Stacking), and outcomes feed back into the memory system as new experiences.

repeated error. Avoiding this failure mode calls for verified abstraction, not just better recall. People do not memorize every contact event. Instead, they form compact principles and revise them when evidence contradicts expectation [48].

We introduce *PhysMem*, a test-time memory framework that enables VLM robot planners to learn physical principles through a scientific memory loop. The system records experiences and detects surprises, defined as outcomes that violate currently held principles. It clusters related experiences to generate candidate hypotheses, uses action-level attribution to isolate planning decisions from execution noise, and verifies hypotheses through targeted interaction before promoting them to long-term storage. We further introduce Memory folding, which compresses supporting episodes into promoted principles, keeping context tractable over extended deployment [39]. The resulting principle set is human-readable and can be inspected, edited, or transferred to new settings.

Across real-world runs lasting over 30 minutes, *PhysMem* enables clear learning curves: for Parts Organization, performance improves from -1 to 9.7 with memory while remaining near 0 without; for Ball Navigation, the gap is 14.7 versus 0.7. In transfer experiments, prior principles provide strong starting points when physics are similar, but test-time adaptation (ours) becomes essential when dynamics change, improving success from 10% to 40% on novel ball types. Simulation experiments across four VLM backbones confirm that our principled abstraction outperforms direct retrieval by 53%, with performance stabilizing around 16–64 principles

after an initial high-variance phase. These results suggest that structured test-time learning can meaningfully improve VLM planners on physical tasks, producing principles that are both effective for decision-making and interpretable for human inspection.

Contributions. We make three contributions. (i) We introduce *PhysMem*, a test-time memory framework that uses targeted interaction to verify typed hypotheses about a deployment context before they condition a frozen VLM planner. (ii) We evaluate *PhysMem* on three real-world manipulation tasks and a brick-insertion simulator across four VLM backbones; principled abstraction reaches 76% on medium-difficulty insertion, a +23-point gain over the no-memory baseline (53%) and +15 points over a free-form Reflexion-style baseline (61%), while direct experience retrieval drops to 23%, below no memory. (iii) We show that the gain scales with base-VLM capability, rising from +5% on Gemini-ER-1.5 to +23% on Gemini-3-Flash, which fits the picture of a system surfacing what the model already knows rather than installing new physics.

II. RELATED WORK

A. Vision-Language-Action Models for Robot Planning

Vision-language models have enabled natural language task specification and common-sense reasoning for robot planning [2, 17, 28, 38]. The Robotics Transformer series [10, 11] demonstrated that web-scale pretraining transfers to robotic control, motivating large-scale cross-embodiment efforts [14, 45, 30]. Recent work has pushed toward more capable generalist policies through flow matching [8], embodied reason-

TABLE I: **Adaptation paradigms for deployed VLM/VLA agents.** `PhysMem` sits in a separate row: frozen base, structured memory, and a verification step before commitment.

Paradigm	Examples	Param. update	Memory content	Interp.	Inf. cost
Online RL	RECAP, $\pi_{0.6}$	✓ Gradient	Implicit weights	✗	sec
Trained memory VLA	MemER, MemoryVLA, MEM	✓ Imit. FT	Visual / latent / text	✗	ms
Reflection (in-episode)	Reflexion, Self-Refine	✗	Free-form text	✓	sec
Retrieval	RAG, MAP-VLA	✗	Raw experiences	△	ms
Verified abstraction (ours)	PhysMem	✗	Verified principles	✓	ms

ing [25, 44, 21], and chain-of-thought mechanisms [64, 68]. Comprehensive analyses [29, 37, 66] reveal that VLM general capabilities poorly predict downstream VLA performance, highlighting a persistent domain gap between declarative knowledge and physical grounding. These approaches generalize well from pre-trained knowledge but cannot adapt to novel physical properties encountered at test time, which is the gap `PhysMem` addresses by grounding VLM reasoning in interaction experience.

B. Adaptation Paradigms for Deployed Agents

How should a deployed VLM/VLA planner improve as it interacts with the world? The literature offers four broad answers, summarized in Table I. The most ambitious option updates the model itself, through online reinforcement learning [41, 62, 52, 47], meta-learning [22], test-time training [57], or imitation finetuning of a memory-augmented backbone, as in MemER [55], MemoryVLA [51, 35], MEM [61], and SAM2Act [18]. A second family keeps the base frozen and feeds it context through retrieval [34, 24, 36] or natural-language reflection [53, 42]. The piece they leave out is a check on whether a remembered experience still applies in the current scene; our experiments make this concrete: selective retrieval matches the no-memory baseline (53% on medium), and free-form reflection plateaus at 61%.

`PhysMem` sits in a fourth corner of this space: a frozen base paired with structured typed memory and a verification step before commitment. For that reason we benchmark against training-free proxies of MemER’s and MemoryVLA’s retrieval mechanisms rather than their full systems, since the parameter-update side of those methods is not directly comparable in a frozen-base setting.

C. Test-Time Learning in Robotics

Test-time adaptation enables robots to adjust to new conditions without full retraining. Meta-learning approaches [22, 23] learn initializations for rapid few-shot adaptation, while domain randomization [60, 3] addresses sim-to-real transfer. Sequence modeling formulations [13, 33] enable in-context learning without gradient updates. Recent work explores on-line adaptation for VLAs through reinforcement learning [41, 62], language corrections [52], and learning from deployment experience [47]. Test-time training methods [57, 31, 63] update models on unlabeled test data through self-supervision. These

methods adjust the policy implicitly rather than learning explicit principles. `PhysMem` instead produces human-readable hypotheses that can be inspected, edited, or transferred to new settings.

D. Memory in Embodied Agents

Memory systems enhance robot planning through experience accumulation [9, 49, 19]. Recent work integrates memory directly into VLAs through hierarchical retrieval [55], cognitive-inspired dual memory banks [51, 35], multi-scale embodied memory [61], and visual foundation models [18]. World models [26, 12, 1, 54] learn to imagine future scenarios, enabling planning without real-world interaction. Reflection-based approaches [53, 42] use LLMs to learn from failures, while retrieval-augmented methods [34, 24, 36] provide relevant past experiences. Lifelong learning methods [39, 43, 65] enable continuous accumulation without catastrophic forgetting. Reflection that triggers only on failures misses what successes have to teach, and retrieval without a check applies experience even when the world has moved on. `PhysMem`’s scientific memory loop does both: it draws hypotheses from successes and failures, tests each through targeted interaction, and keeps only the survivors as grounded principles.

III. METHOD

We present *PhysMem*, a test-time memory framework that lets a VLM robot planner learn physical principles through interaction. The mechanism is a scientific memory loop that drafts hypotheses from experience, checks each against targeted experiments, and commits only the survivors to guide future decisions.

A. Problem Formulation

We formulate physical manipulation as a sequential decision problem following the options framework [58]. Let $\mathcal{S}$ denote the state space and $\mathcal{O}$ the observation space (visual observations). At each decision point t , the agent receives observation $o_t \in \mathcal{O}$ and task description τ . Following Sutton et al., we define an *option* $\omega = \langle \mathcal{I}, \pi, \beta \rangle$ as a temporally-extended action consisting of an initiation set $\mathcal{I} \subseteq \mathcal{S}$, an intra-option policy $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$, and a termination condition $\beta : \mathcal{S} \rightarrow [0, 1]$.

In our framework, a VLM-based high-level policy π_θ^H selects options based on observations, task context, and learned principles:

$$\omega_t = \pi_\theta^H(o_t, \tau, \mathcal{P}_t) \quad (1)$$

Algorithm 1 Scientific Memory Loop

Require: Experience buffer $\mathcal{E}$, Hypothesis store $\mathcal{H}$, Principle store $\mathcal{P}$

Require: Reflection model f_ϕ , thresholds $\tau_p, \tau_r, n_{\min}$

```
1: function RECORDEXPERIENCE( $o, \omega, r, c, s, \mathcal{P}_{\text{active}}$ )
2:    $e \leftarrow (o, \omega, r, c, s)$ 
3:    $\rho \leftarrow \text{Resonance}(e, \mathcal{P}_{\text{active}})$   $\triangleright$  Eq. 3
4:    $\mathcal{E} \leftarrow \mathcal{E} \cup \{e\}$ 
5:   if  $\rho < 1$  then  $\triangleright$  Surprising experience
6:     TRIGGERCONSOLIDATION( $e$ )
7:   else
8:     REINFORCEACTIVEPRINCIPLES( $\mathcal{P}_{\text{active}}$ )
9:   end if
10: end function

11: function CONSOLIDATE( $\mathcal{E}$ )
12:    $\{\mathcal{C}_k\} \leftarrow \text{ClusterBySymbolicState}(\mathcal{E})$ 
13:   for each cluster  $\mathcal{C}_k$  with  $|\mathcal{C}_k| \geq n_{\min}$  do
14:      $\mathcal{H}_k \leftarrow f_\phi(\mathcal{C}_k, \mathcal{P}, \mathcal{H})$   $\triangleright$  Eq. 4
15:      $\mathcal{H} \leftarrow \mathcal{H} \cup \mathcal{H}_k$ 
16:   end for
17: end function

18: function ATTRIBUTEANDPROMOTE( $\mathcal{E}, \mathcal{H}, \mathcal{P}$ )
19:   for each hypothesis  $h \in \mathcal{H}$  do
20:     Update  $\text{conf}(h)$  via action-level attribution  $\triangleright$  Eq. 5
21:     if  $\text{conf}(h) \geq \tau_p$  and  $|\mathcal{E}_{\text{support}}| \geq 3$  then
22:        $\mathcal{P} \leftarrow \mathcal{P} \cup \{h\}$   $\triangleright$  Promote to principle
23:        $\mathcal{E} \leftarrow \mathcal{E} \setminus \mathcal{E}_{\text{folded}}$   $\triangleright$  Memory folding
24:        $\mathcal{H} \leftarrow \mathcal{H} \setminus \{h\}$ 
25:     else if  $\text{conf}(h) \leq \tau_r$  and  $|\mathcal{E}_{\text{contradict}}| \geq 2$  then
26:        $\mathcal{H} \leftarrow \mathcal{H} \setminus \{h\}$   $\triangleright$  Refute hypothesis
27:     end if
28:   end for
29: end function
```

where $\mathcal{P}_t \subseteq \mathcal{P}$ denotes the set of active principles retrieved from memory at time t . The selected option ω_t is executed by a low-level policy π^L (which may be a motion planner, VLA, or other controller) until termination. The challenge is that physical properties (friction, dynamics, spatial relationships) vary across objects and environments, and cannot be fully captured by pre-trained VLM knowledge θ alone.

Our goal is to learn a principle set $\mathcal{P}^*$ that grounds the VLM’s reasoning in interaction experience, such that:

$$\mathbb{E} \left[\sum_{t=0}^T r_t \mid \pi_\theta^H(\cdot, \cdot, \mathcal{P}^*) \right] > \mathbb{E} \left[\sum_{t=0}^T r_t \mid \pi_\theta^H(\cdot, \cdot, \emptyset) \right] \quad (2)$$

where r_t is the reward at step t . Critically, $\mathcal{P}^*$ must be learned at test time without modifying the VLM parameters θ .

B. System Overview

Figure 2 illustrates the PhysMem architecture. The system consists of three components: (1) a VLM-based **Planner** that generates high-level decisions given observations and principles, (2) a **Memory** system organized in three tiers (episodic, working, and long-term), and (3) an **Executor** that carries out planned actions via low-level policies.

At each episode, the planner queries memory for relevant principles, makes decisions, and observes outcomes. What makes the loop work is that memory is not a static database; it evolves through a scientific memory loop that continuously refines raw experiences into verified principles. Where retrieval-augmented approaches apply retrieved knowledge as-is, PhysMem tests each hypothesis before commitment, which avoids the dogmatism problem of stale experience hurting performance.

C. Scientific Memory Loop

The scientific memory loop is the central mechanism of PhysMem. It moves raw experience through verified principles in four phases: experience collection, hypothesis generation, attribution, and principle promotion. Algorithm 1 provides the pseudocode.

a) Experience Collection with Resonance Checking:

We store experiences from both successful and unsuccessful interactions. Each experience $e = (o, \omega, r, c, s)$ records the observation o , selected option ω , outcome $r \in \{0, 1\}$, context c (task description, subtask), and symbolic state s (discrete features like action type, object properties).

A central mechanism is resonance checking, which measures how well an experience matches active principles. Let $\mathcal{P}_{\text{active}} \subseteq \mathcal{P}$ be the principles applied during decision-making. We compute a resonance score:

$$\rho(e, \mathcal{P}_{\text{active}}) = \frac{|\{p \in \mathcal{P}_{\text{active}} : \text{consistent}(e, p)\}|}{|\mathcal{P}_{\text{active}}|} \quad (3)$$

where $\text{consistent}(e, p)$ checks whether the experience outcome aligns with the principle’s prediction. When $\rho < 1$ (a “surprise”), the experience is prioritized for consolidation; when $\rho = 1$, the experience reinforces existing principles without triggering new hypothesis generation. This surprise-driven filtering focuses learning on novel situations.

b) *Hypothesis Generation:* Experiences are periodically clustered by symbolic similarity. For each cluster $\mathcal{C}_k$ with sufficient experiences ($|\mathcal{C}_k| \geq n_{\min}$), we use a reflection model f_ϕ (VLM in real-world, LLM in simulation) to generate hypotheses about patterns:

$$\mathcal{H}_k = f_\phi(\mathcal{C}_k, \mathcal{P}, \mathcal{H}_{\text{existing}}) \quad (4)$$

where $\mathcal{P}$ and $\mathcal{H}_{\text{existing}}$ provide context to avoid generating duplicate knowledge. Each hypothesis $h \in \mathcal{H}_k$ takes a typed form:

- **AVOID:** “Don’t do X when Y ” (from failures)
- **PREFER:** “Do X when Y ” (from successes)
- **SEQUENCE:** “Do X before Y ” (temporal constraints)

c) *Action-Level Attribution:* Hypotheses are judged by action-level outcomes, not episode-level success. For hypothesis h about action type a^* , we update confidence using only experiences where that specific action was attempted:

$$\text{conf}(h) \leftarrow \text{conf}(h) + \alpha \cdot \frac{|\{e \in \mathcal{E}_h : a_e = a^*, r_e = 1\}|}{|\{e \in \mathcal{E}_h : a_e = a^*\}|} \quad (5)$$

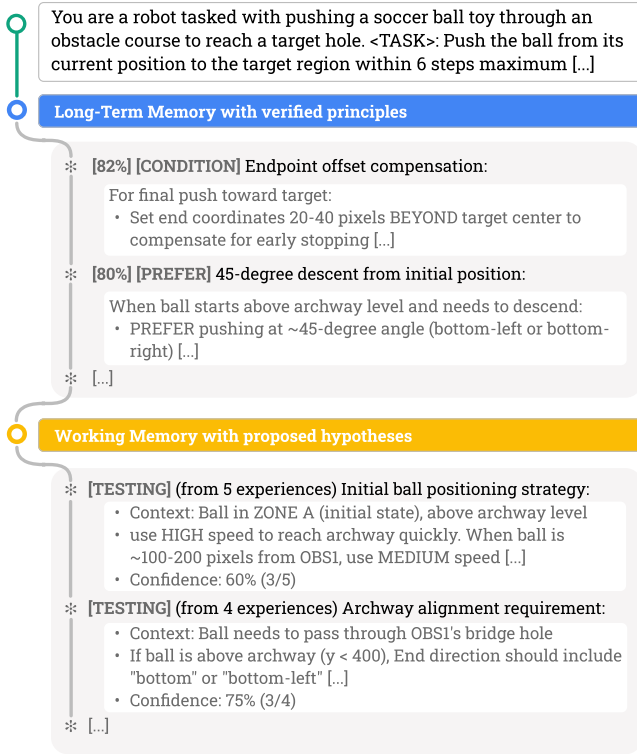


Fig. 3: **Memory injection into VLM prompts.** Verified principles (blue) and active hypotheses (yellow) are inserted into the planner’s context with confidence scores and typed constraints (PREFER, AVOID, SEQUENCE).

where $\mathcal{E}_h$ denotes experiences relevant to h and α is the learning rate. This isolates specific action effects from confounding factors.

d) Verification and Principle Promotion: Hypotheses achieving high confidence ($\text{conf}(h) \geq \tau_p$, typically 0.8) with sufficient supporting evidence ($|\mathcal{E}_{\text{support}}| \geq 3$) are promoted to principles. Upon promotion, source experiences are “folded” into the principle, compressing episodic memory while preserving learned knowledge:

$$\mathcal{P} \leftarrow \mathcal{P} \cup \{h\}, \quad \mathcal{E} \leftarrow \mathcal{E} \setminus \mathcal{E}_{\text{folded}} \quad (6)$$

Hypotheses with low confidence ($\text{conf}(h) \leq \tau_r$) and contradicting evidence are refuted and removed.

D. Memory Architecture

PhysMem organizes memory in three tiers: (1) **Episodic memory** stores raw experiences with symbolic state for efficient filtering, bounded by capacity N_{max} ; (2) **Working memory** holds hypotheses under test, each with confidence scores from supporting/contradicting evidence; (3) **Long-term memory** contains verified principles that guide decisions, with importance scores that decay over time ($\gamma = 0.995$) to forget outdated knowledge.

E. Retrieval and Application

At inference time, PhysMem retrieves relevant principles via symbolic filtering (matching action type and object properties) followed by semantic ranking. The top- k principles

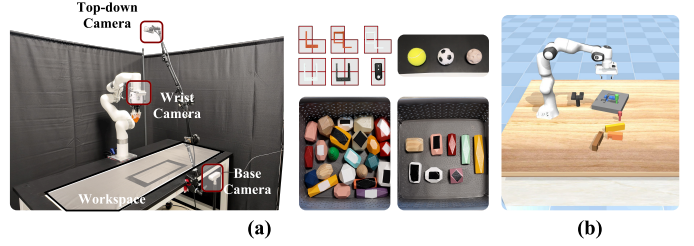


Fig. 4: **Experimental environments.** (a) Left: Real-world platform with xArm6 robot, fin-ray soft grippers, and multi-view RealSense cameras in an enclosed workspace. Right: The partial props used in the experiments. (b) Reflect-VLM simulation [20] with Franka Panda robot for large-scale experiments.

and active hypotheses are injected into the VLM prompt as shown in Figure 3. When resonance is low, the system prioritizes fresh learning over applying potentially misleading prior knowledge. Full implementation details are in Appendix A.

IV. EXPERIMENTAL SETUP

We evaluate PhysMem in two complementary settings: real-world manipulation tasks that require physical understanding VLMs lack from pretraining, and simulation experiments that enable large-scale analysis across VLMs and difficulty levels. Figure 4 shows both environments.

Our experimental design deliberately separates high-level reasoning from low-level control. The VLM outputs discrete decisions (which part to place, which direction to push, which stone to stack), while motion planning handles execution. This separation isolates the variable under study: if performance improves, the improvement must stem from better reasoning since the executor remains fixed. We evaluate both task success and plan quality, since success rate alone can be misleading, where a planner might achieve high success through conservative strategies that do not demonstrate physical understanding.

A. Real-World Tasks

Our real-world platform uses an xArm6 with the TPU-printed fin-ray soft grippers and Intel RealSense D435 cameras (top-down at 1280×720 , wrist-mounted at 640×480). For VLM planning, we use Gemini-3.0-Flash [15] with thinking mode. Hypothesis generation uses Qwen3-VL [4], running asynchronously every 15 seconds. Each task runs for 10–20 episodes. We design three tasks where correct reasoning requires physical parameters that vision alone cannot provide (Figure 5):

a) Parts Organization: Place 6 irregularly-shaped parts onto a 3×10 grid, minimizing total cells occupied. Each part occupies 2–4 cells; the VLM outputs placement indices. A good plan exploits interlocking geometries to pack efficiently. The challenge: part shapes allow overlapping in 3D when aligned correctly, but these spatial relationships only emerge through placement attempts.

b) Ball Navigation: Push a soccer ball through obstacles to reach the target within 6 steps. The VLM specifies push direction, coordinates, and speed; scoring rewards progress

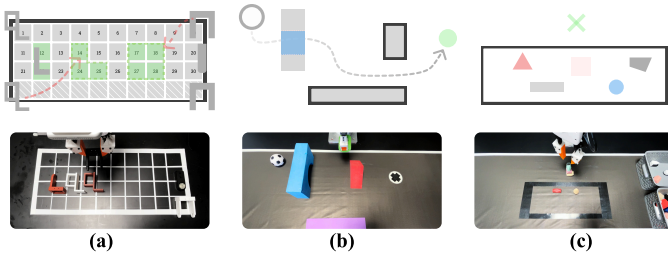


Fig. 5: **Real-world tasks.** Top: symbolic representations; bottom: actual setups. (a) Grid layout and placement trajectories for Parts Organization. (b) Obstacle course and ball trajectory for Ball Navigation. (c) Stone arrangement and stacking position for Balanced Stacking.

toward the goal. A good plan accounts for rolling distance and obstacle rebounds. The challenge: surface friction and ball elasticity vary across the workspace, requiring trial-and-error calibration.

c) **Balanced Stacking:** Build a stable tower from 5 balance stones with varying sizes, textures, and weight distributions. The VLM selects stacking order; scoring rewards height and penalizes collapses. A good plan sequences stones by stability contribution. The challenge: friction and weight distribution are invisible, and only contact reveals which pairings hold. Full task specifications appear in Appendix F.

B. Simulation Benchmark

We complement real-world evaluation with the Reflect-VLM brick insertion benchmark [20], a MuJoCo-based environment with a Franka Panda robot. The VLM must learn correct insertion ordering: placing the wrong brick first blocks subsequent insertions. Difficulty scales with brick count: easy (2–3), medium (4–5), hard (6–8). This controlled setting enables experiments at scale (500+ episodes) across multiple VLMs. Details appear in Appendix E.

V. EXPERIMENTS

Our experiments address five questions:

- 1) **Memory Paradigms:** How does PhysMem compare against existing memory paradigms (selective retrieval, token-memory, and reflection) for deployed VLM agents? (Section V-A)
- 2) **Physics Understanding:** Can PhysMem improve the VLM’s understanding of task-specific physical properties through interaction? (Section V-B)
- 3) **Decision Quality:** Does improved physics understanding lead to better embodied decision-making over time? (Section V-C)
- 4) **Memory Transfer:** Can experience-derived memory provide benefits in out-of-distribution scenarios, and when does it fail? (Section V-D)
- 5) **Architecture Design:** Which components of PhysMem are essential, and why do we need memory compression and forgetting? (Section V-G)

TABLE II: **Main results: PhysMem vs. memory paradigms on Brick Insertion** (100 episodes per cell, 3 seeds). MemER-style retrieval lands at the No-Memory baseline on medium (53%), and only PhysMem’s verified principles exceed No Memory on hard. Best per-difficulty in bold.

Method	Success Rate			Tokens
	Easy	Med.	Hard	
No Memory	83%	53%	28%	0.40×
Direct Retrieval	48%	23%	8%	0.55×
MemER-style Retrieval	85%	53%	19%	1.0×
MemoryVLA-style Token Memory	69%	35%	13%	2.5×
Reflexion-style	86%	61%	31%	1.2×
PhysMem (Full)	89%	76%	39%	1.0×

A. Comparison Against Memory Paradigms

We compare PhysMem against five baselines, one for each main memory paradigm used by deployed VLM agents, all run under the same frozen-base setup (Gemini-3-Flash, identical task budget, top- $K=5$ memory injection where applicable). MemER [55] and MemoryVLA [51] require parameter updates (imitation finetuning or end-to-end VLA training); running their full systems would take us out of the frozen-base setting, so we benchmark training-free proxies that isolate their retrieval mechanism (implementation details in Appendix A). The five baselines: **No Memory**; **Direct Retrieval** (top- K raw experiences by symbolic-state similarity, no abstraction); **MemER-style Retrieval** (VLM-judged comparative ranking, top- $K=5$, no verification); **MemoryVLA-style Token Memory** (full episode history accumulated as context); and **Reflexion-style** [53] (free-form reflection on successes and failures, retrieval by semantic similarity).

Three patterns come out of Table II, and together they pin the gain to abstraction and verification rather than memory access itself. Selective retrieval is no better than no memory: MemER-style retrieval lands at 53% on medium, exactly the No-Memory baseline, recovering Direct Retrieval’s 30-point loss but adding nothing on top. Unverified memory then hurts once situations stop repeating: on hard, all three unverified variants drop below the 28% No-Memory floor (Direct Retrieval 8%, MemoryVLA-style 13%, MemER-style 19%), and only PhysMem’s verified principles (39%) come out ahead. More context is not more help either: MemoryVLA-style trails MemER-style by 18 points at 2.5× the token cost, and PhysMem’s lead over MemER-style is largest on the harder tasks (4%/23%/20% on easy/medium/hard), the opposite of what a better-retrieval interpretation would predict.

B. Learned Physical Principles

Beyond task scores, we examine whether PhysMem produces reasoning that becomes better grounded in the physics of the scene. Figure 7 shows the resonance score ρ , which measures how well predictions align with observed outcomes. High resonance means the planner anticipated task dynamics

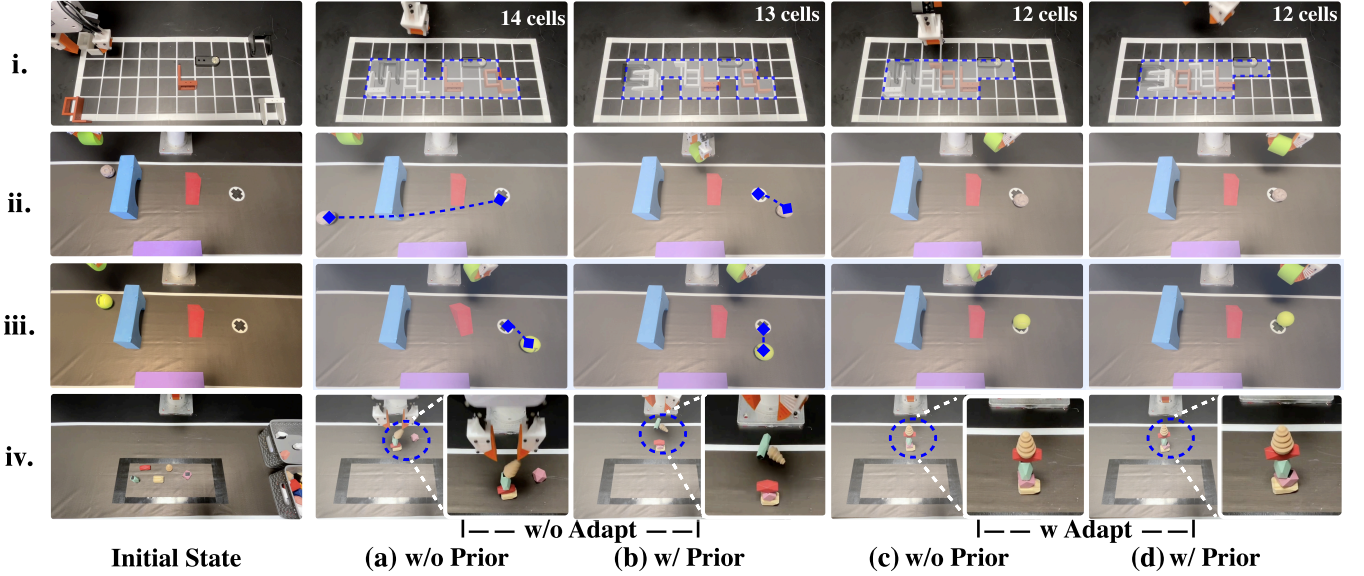


Fig. 6: **End-state keyframes across tasks and OOD variants.** A 2×2 ablation of prior knowledge and on-the-fly adaptation, shown as final states for four task instances. Rows: (i) Parts Organization on its OOD configuration (covered cells, lower is better); (ii) Ball Navigation with the in-distribution soccer ball; (iii) Ball Navigation with an OOD tennis ball; (iv) Balanced Stacking. Columns: (a) no prior, no adaptation; (b) prior only; (c) adaptation only; (d) full PhysMem. Across all four rows, the full setting reaches the most efficient or most stable end state; either component alone leaves a visible gap that Section V-D quantifies.

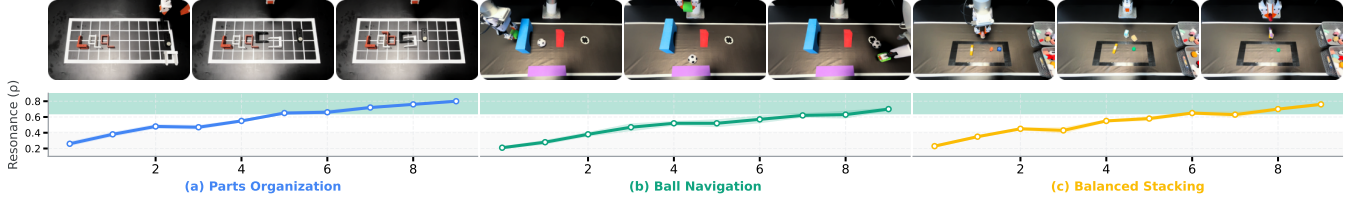


Fig. 7: **Resonance score evolution across tasks.** Each panel shows RGB keyframes alongside the resonance curve. Resonance ρ measures prediction-outcome alignment: low values indicate surprising outcomes that trigger learning; high values (green region) indicate principled reasoning. **Left:** Parts Organization progresses from scattered placement to efficient packing. **Middle:** Ball Navigation learns ball dynamics through trial. **Right:** Balanced Stacking discovers stable stone combinations. All tasks transition from reactive ($\rho < 0.5$) to rational ($\rho > 0.7$) behavior within 10 episodes.

correctly; low resonance means the outcome surprised the active principles. All three tasks follow a consistent pattern: ρ climbs from ≈ 0.2 in early episodes to ≈ 0.9 by episode 10, crossing $\rho = 0.7$ around episode 5–6, after which predictions match outcomes more often than not.

This metric captures reasoning quality that task scores alone cannot reveal. Raw success rates do not distinguish a planner that understands physics from one that finds safe but suboptimal strategies (always pushing slowly, avoiding complex placements). High resonance asks the planner’s internal model to actually predict the outcome rather than just produce an acceptable result. Episodes in the high-resonance regime ($\rho > 0.7$) score $2.3\times$ higher than episodes 1–3. The steady rise in ρ indicates that PhysMem accumulates verified physical understanding instead of overfitting to a narrow solution.

The principles themselves are human-readable: spatial constraints for Parts Organization (“avoid overlapping internal regions of q-shaped parts”), dynamics rules for Ball Navigation,

and stability heuristics for Balanced Stacking (“select the largest high-friction stone as base”). Persistent failures occur when learned principles conflict or do not apply. Full principle inventories appear in Appendix F-E.

C. Test-Time Evolution

To evaluate how PhysMem actually improves through interaction and how much experience is needed, we evaluate performance under different experience utilization levels (0%, 25%, 50%, 100%), with 3 runs per condition. The test-time learning curve is shown in Figure 8.

Without memory (0%), performance remains flat; with full memory, Parts Organization improves from $-1 \rightarrow 9.7$ and Ball Navigation shows an even larger gap (14.7 vs. 0.7). Task complexity determines experience requirements: Ball Navigation benefits from full experience (50% vs. 100%: 11.0 vs. 14.7), while Balanced Stacking shows diminishing returns (50% nearly matches 100%). The memory system efficiently extracts generalizable principles when task structure permits.

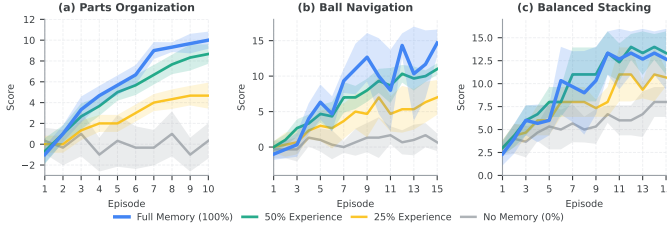


Fig. 8: **Test-time evolution across experience utilization levels.** Each panel shows learning curves with shaded standard deviation bands (3 runs per condition). **Blue:** Full memory (100%); **Green:** 50% experience; **Yellow:** 25% experience; **Gray:** No memory (0%). Without memory, performance remains flat across all tasks. Complex dynamics (Ball Navigation) benefit most from full experience, while simpler tasks (Balanced Stacking) show diminishing returns.

D. Memory Transfer

Can learned principles transfer to out-of-distribution (OOD) scenarios? We ablate prior knowledge (principles from in-distribution deployment) and test-time adaptation (via scientific memory loop). Each condition runs 10 trials. We report results in Table III and details in Appendix F-F.

The results reveal when transfer succeeds and fails. For Balanced Stacking, prior knowledge alone achieves 80% success because stability principles transfer well to new stones. For Parts Organization, prior knowledge improves score from -0.6→6.9 by avoiding common packing errors. In contrast, Ball Navigation with new ball types shows that prior knowledge matches zero-shot performance: the new balls have different dynamics, so prior principles do not transfer. Adding adaptation improves the score of Ball Navigation to 7.1 and 40% success as the system learns new friction and elasticity properties. The full PhysMem achieves the best performance across all tasks, confirming that prior knowledge and test-time adaptation are complementary.

The following experiments use the simulation benchmark for large-scale analysis across VLMs and difficulty levels. In simulation, we report success rates because the controlled environment enables fair comparison across hundreds of episodes, and the brick insertion task has a clear binary success criterion (all bricks correctly placed). Real-world tasks

TABLE III: **Memory transfer to OOD task variants.** We ablate prior knowledge (Prior) and test-time adaptation (Adapt). Prior knowledge helps when physics are similar (Parts Organization, Balanced Stacking). For Ball Navigation with new ball types, adaptation is essential since dynamics differ substantially. Best results in **bold**. **Green** : better; **Red** : worse.

		Parts Organization		Ball Navigation		Balanced Stacking	
Prior	Adapt	Score	Succ.	Score	Succ.	Score	Succ.
✗	✗	-0.6	0/10	1.6	1/10	6.7	4/10
✗	✓	3.3	1/10	5.5	2/10	8.3	7/10
✓	✗	6.9	3/10	2.9	1/10	9.2	8/10
✓	✓	8.3	4/10	7.1	4/10	12.3	9/10

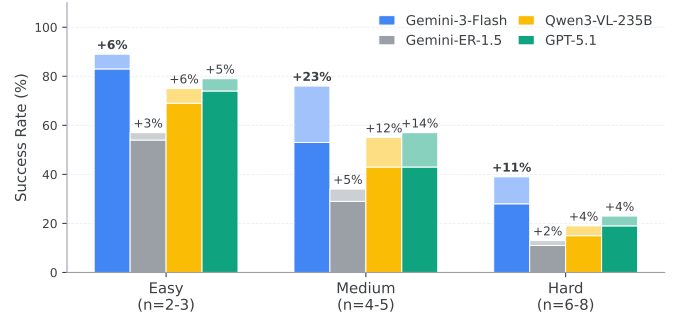


Fig. 9: **Success rates across VLMs and difficulty levels.** Bars show baseline performance (w/o PhysMem); annotations indicate improvement with test-time learning. Gemini-3-Flash dominates across all difficulties and shows the largest improvement. Weaker models benefit less from test-time learning, suggesting that memory amplifies existing capabilities rather than compensating for fundamental limitations.

use plan quality metrics to capture nuanced physical reasoning.

E. Scaling Across VLMs

How does test-time learning interact with VLM capability and task difficulty? We evaluate four VLMs with thinking mode: Gemini-3-Flash [15], Gemini-ER-1.5 [16], GPT-5.1 [46], and Qwen3-VL-235B [4]. Each condition runs 100 episodes. Figure 9 shows results. VLM capability sets the baseline. Gemini-3-Flash leads across difficulties, reaching 53% on medium against 29–43% for the other models. The gain from test-time learning then scales with that baseline: on medium, Gemini-3-Flash improves by +23% (53% to 76%), GPT-5.1 by +14%, Qwen3-VL by +12%, and Gemini-ER-1.5 by +5%. The pattern fits a system that surfaces what the model already knows, since the planner needs enough capability in the first place to draft and verify useful hypotheses. All difficulty levels benefit, with medium showing the largest gain (+23%); hard tasks still improve (+11%) because accumulated failures yield AVOID hypotheses that head off repeated mistakes.

F. Principle Scaling

How does principle count affect success rate? We run 500 episodes per difficulty using Gemini-3-Flash and measure performance as principles accumulate (1→128). Figure 10 reveals distinct patterns.

Medium tasks show the most pronounced scaling: performance rises from 55%→67% between 2 and 8 principles, then stabilizes (76% at 64+). Easy tasks saturate quickly (83%→89%). Hard tasks improve by +11% even though many remain unsolved; 68% of the principles learned at hard difficulty are avoidance constraints (versus 41% on medium), so the planner gets better largely by ruling moves out, not by solving the task end-to-end.

G. Ablations

Which components of PhysMem are essential, and how does their importance vary with task difficulty? We ablate

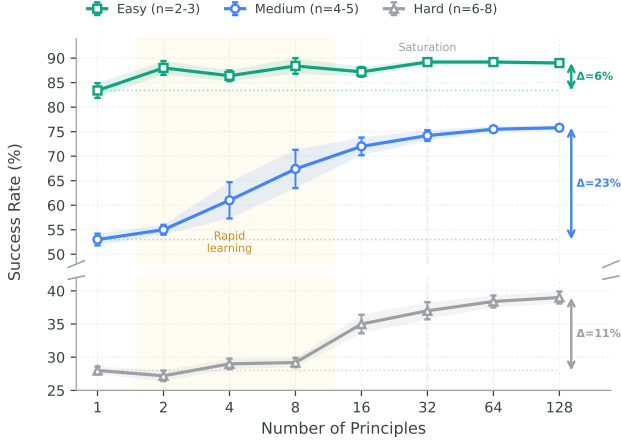


Fig. 10: **Principle scaling across difficulty levels.** Medium tasks (blue) show rapid learning between 2 and 8 principles before stabilizing. Easy tasks (green) saturate quickly with moderate improvement. Hard tasks (gray) show substantial improvement potentially through accumulated failure cases. Shaded region marks the “rapid learning” phase.

key modules across all difficulty levels (100 episodes each, Gemini-3-Flash), and report the results in Table IV.

TABLE IV: **Ablation study across difficulty levels.** Success rates and token consumption (on medium) for brick insertion. Direct retrieval validates the need for principle abstraction. Component importance scales with task complexity.

Configuration	Success Rate			Tokens
	Easy	Med.	Hard	
PhysMem (Full)	89%	76%	39%	1.0×
Direct Retrieval	48%	23%	8%	0.55×
w/o Resonance	81%	58%	21%	1.3×
w/o Verification	85%	64%	27%	0.85×
w/o Forgetting	91%	78%	36%	3.4×
w/o Working Mem.	84%	69%	28%	0.75×

Principle abstraction is the load-bearing component. Direct retrieval degrades from 48% on easy to 23% on medium and 8% on hard, since state matching becomes fragile with complex dependency graphs while principled abstraction holds up.

Component importance scales with difficulty. Resonance filtering prevents conflicting principles from degrading decisions (-8% on easy, -18% on hard). Verification ensures hypothesis quality before promotion (-4%→-12%). Working memory enables hypothesis exploration, with increasing importance on harder tasks (-5%→-11%).

Forgetting trades accuracy against efficiency. Keeping all experiences gives a small bump on easy and medium (+2%) but hurts performance on hard (-3%) because accumulated noise interferes with decisions, and at 3.4× the token cost

the trade-off favors forgetting. Full ablations appear in Appendix B.

VI. DISCUSSION & LIMITATION

a) Observation Space: Physical learning depends on observation modalities that capture properties vision alone cannot. Our experiments use visual observations and outcome feedback, but the physical world offers richer signals (tactile, force, and audio) that could accelerate principle discovery. Recent work shows that VLMs contain latent physical knowledge that tactile grounding can activate [27], and audio sensing reveals material properties inaccessible to vision [40]. Active perception strategies [56] could further enable targeted exploration. PhysMem’s architecture naturally extends to such inputs: principles could emerge from tactile failure patterns or acoustic contact signatures.

b) Reasoning Representation: Text-based principles excel at discrete rules but struggle with continuous dynamics like trajectories or force profiles. Language cannot fully capture the continuous nature of physical interaction [5, 59]. Visual chain-of-thought reasoning [67] achieves 17% improvements over text-based approaches by predicting future frames, and reasoning through continuous visual tokens [50] or latent world models [7] suggests future systems may bypass language entirely for physics prediction. Our verification loop could operate on visual state predictions rather than text: world models that imagine outcomes [26, 12, 6] naturally complement the scientific loop by providing richer hypothesis testing.

c) Why Abstraction Beats Retrieval: The direct-experience-retrieval ablation makes the point cleanly: raw episodic replay fails because situations never repeat exactly, which lines up with classical findings in episodic memory research [9, 49]. Principles work because they abstract over specific instances. The transfer experiments show the flip side: abstraction has limits, and irrelevant principles hurt performance once the world drifts. Where retrieval-augmented approaches [34, 24] apply experience without checking whether it still holds, PhysMem runs that check, and the ability to revise beliefs is what makes the loop a learning system rather than a memorizing one.

d) Limitations: Our work focuses on high-level planning rather than low-level control. Integrating learned principles into end-to-end VLA execution is still open [47, 25], and current VLAs struggle when task descriptions include novel physical constraints; closing that gap is a natural next step. For continuous real-world deployment, environment reset is a practical bottleneck: when objects fall off tables or parts break, the system needs human intervention, and autonomous recovery and repair would be required for truly lifelong learning [39].

VII. CONCLUSION

We presented PhysMem, a test-time memory framework that lets a VLM robot planner learn physical principles through

interaction. The scientific loop of hypothesis generation, verification, and promotion produces knowledge that is both effective and interpretable. Across three real-world tasks and simulation benchmarks, principled abstraction outperforms raw retrieval, and the ability to revise beliefs is what makes the system robust under shifting physical conditions. A natural next step is to let the same verification step run against world-model rollouts, so that text principles and learned dynamics can share one loop.

REFERENCES

- [1] Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical AI. *arXiv preprint arXiv:2501.03575*, 2025.
- [2] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Guber, Karol Gopalakrishnan, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Proceedings of the Conference on Robot Learning (CoRL)*, pages 287–318, 2022.
- [3] Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafał Józefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *International Journal of Robotics Research*, 39(1):3–20, 2020.
- [4] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Kegin Chen, Zesen Cheng, Lianhao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025. URL <https://arxiv.org/abs/2511.21631>.
- [5] Vahid Balazadeh, Mohammadmehdi Ataei, Hyunmin Cheong, Amir Hosein Khasahmadi, and Rahul G Krishnan. Physics context builders: A modular framework for physical reasoning in vision-language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7318–7328, 2025.
- [6] Philip J. Ball, Jakob Bauer, Frank Belletti, Bethanie Brownfield, Ariel Ephrat, Shlomi Fruchter, Agrim Gupta, Kristian Holsheimer, Aleksander Holynski, Jiri Hron, Christos Kaplanis, Marjorie Limont, Matt McGill, Yanko Oliveira, Jack Parker-Holder, Frank Perbet, Guy Scully, Jeremy Shar, Stephen Spencer, Omer Tov, Ruben Villegas, Emma Wang, Jessica Yung, Cip Baetu, Jordi Berbel, David Bridson, Jake Bruce, Gavin Buttmore, Sarah Chakera, Bilva Chandra, Paul Collins, Alex Cullum, Bogdan Damoc, Vibha Dasagi, Maxime Gazeau, Charles Gbadamosi, Woohyun Han, Ed Hirst, Ashyana Kachra, Lucie Kerley, Kristian Kjems, Eva Knoepfel, Vika Koriakin, Jessica Lo, Cong Lu, Zeb Mehring, Alex Moufarek, Henna Nandwani, Valeria Oliveira, Fabio Pardo, Jane Park, Andrew Pierson, Ben Poole, Helen Ran, Tim Salimans, Manuel Sanchez, Igor Saprykin, Amy Shen, Sailesh Sidhwani, Duncan Smith, Joe Stanton, Hamish Tomlinson, Dimple Vijaykumar, Luyu Wang, Piers Wingfield, Nat Wong, Keyang Xu, Christopher Yew, Nick Young, Vadim Zubov, Douglas Eck, Dumitru Erhan, Koray Kavukcuoglu, Demis Hassabis, Zoubin Ghahramani, Raia Hadsell, Aäron van den Oord, Inbar Mosseri, Adrian Bolton, Satinder Singh, and Tim Rocktäschel. Genie 3: A new frontier for world models. 2025.
- [7] Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, Hongyan Zhao, Hanyu Liu, Zhizhong Su, Lei Ma, Hang Su, and Jun Zhu. Motus: A unified latent action world model. *arXiv preprint arXiv:2512.13030*, 2025.
- [8] 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. corr, abs/2410.24164, 2024. doi: 10.48550. *arXiv preprint ARXIV:2410.24164*.
- [9] Charles Blundell, Benigno Uribe, Alexander Pritzel, Yazhe Li, Avraham Ruderman, Joel Z Leibo, Jack Rae, Daan Wierstra, and Demis Hassabis. Model-free episodic control. *arXiv preprint arXiv:1606.04460*, 2016.
- [10] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.
- [11] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.
- [12] Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *Forty-first International Conference on Machine Learning*, 2024.
- [13] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021.
- [14] Open X-Embodiment Collaboration, Abby O’Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew Wang, Andrey Kolobov, Anikait Singh, Animesh Garg, Aniruddha Kembhavi, Annie Xie, Anthony Brohan, Antonin Raffin, Archit Sharma, Arefeh Yavary, Arhan Jain, Ashwin Balakrishna, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Blake Wulfe, Brian Ichter, Cewu Lu, Charles Xu, Charlotte Le,

- Chelsea Finn, Chen Wang, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Christopher Agia, Chuer Pan, Chuyuan Fu, Coline Devin, Danfei Xu, Daniel Morton, Danny Driess, Daphne Chen, Deepak Pathak, Dhruv Shah, Dieter B  chler, Dinesh Jayaraman, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Ethan Foster, Fangchen Liu, Federico Ceola, Fei Xia, Feiyu Zhao, Felipe Vieira Frujeri, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Gilbert Feng, Giulio Schiavi, Glen Berseth, Gregory Kahn, Guangwen Yang, Guanzhi Wang, Hao Su, Hao-Shu Fang, Haochen Shi, Henghui Bao, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homanga Bharadhwaj, Homer Walke, Hongjie Fang, Huy Ha, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jad Abou-Chakra, Jaehyung Kim, Jaimyn Drake, Jan Peters, Jan Schneider, Jasmine Hsu, Jay Vakil, Jeannette Bohg, Jeffrey Bingham, Jeffrey Wu, Jensen Gao, Jiaheng Hu, Jiajun Wu, Jialin Wu, Jiankai Sun, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jimmy Wu, Jingpei Lu, Jingyun Yang, Jitendra Malik, Jo  o Silv  rio, Joey Hejna, Jonathan Boohar, Jonathan Tompson, Jonathan Yang, Jordi Salvador, Joseph J. Lim, Junhyek Han, Kaiyuan Wang, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Black, Kevin Lin, Kevin Zhang, Kiana Ehsani, Kiran Lekkala, Kirsty Ellis, Krishan Rana, Krishnan Srinivasan, Kuan Fang, Kunal Pratap Singh, Kuo-Hao Zeng, Kyle Hatch, Kyle Hsu, Laurent Itti, Lawrence Yunliang Chen, Lerrel Pinto, Li Fei-Fei, Liam Tan, Linxi "Jim" Fan, Lionel Ott, Lisa Lee, Luca Weihs, Magnum Chen, Marion Lepert, Marius Memmel, Masayoshi Tomizuka, Masha Itkina, Mateo Guaman Castro, Max Spero, Maximilian Du, Michael Ahn, Michael C. Yip, Mingtong Zhang, Mingyu Ding, Minh  o Heo, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Muhammad Zubair Irshad, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Ning Liu, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Osbert Bastani, Pannag R Sanketi, Patrick "Tree" Miller, Patrick Yin, Paul Wohlhart, Peng Xu, Peter David Fagan, Peter Mitrano, Pierre Sermanet, Pieter Abbeel, Priya Sundareshan, Qiuyu Chen, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Mart  n-Mart  n, Rohan Bajjal, Rosario Scalise, Rose Hendrix, Roy Lin, Runjia Qian, Ruohan Zhang, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Shan Lin, Sherry Moore, Shikhar Bahl, Shivin Dass, Shubham Sonawani, Shubham Tulsiani, Shuran Song, Sichun Xu, Siddhant Haldar, Siddharth Karamcheti, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Subramanian Ramamoorthy, Sudeep Dasari, Suneel Belkhale, Sungjae Park, Suraj Nair, Suvir Mirchandani, Takayuki Osa, Tanmay Gupta, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Thomas Kollar, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Trinity Chung, Vidhi Jain, Vikash Kumar, Vincent Vanhoucke, Vitor Guizilini, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiangyu Chen, Xiaolong Wang, Xinghao Zhu, Xinyang Geng, Xiyuan Liu, Xu Liangwei, Xuanlin Li, Yansong Pang, Yao Lu, Yecheng Jason Ma, Yejin Kim, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Yilin Wu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yongqiang Dou, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yue Cao, Yueh-Hua Wu, Yujin Tang, Yuke Zhu, Yunchu Zhang, Yunfan Jiang, Yunshuang Li, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zehan Ma, Zhuo Xu, Zichen Jeff Cui, Zichen Zhang, Zipeng Fu, and Zipeng Lin. Open X-Embodiment: Robotic learning datasets and RT-X models. <https://arxiv.org/abs/2310.08864>, 2023.
- [15] Google DeepMind. Gemini 3 flash: Frontier intelligence built for speed, 2025. URL <https://blog.google/products-and-platforms/products/gemini/gemini-3-flash/>. Accessed: 2026-01-21.
- [16] Google DeepMind. Gemini embodied reasoning 1.5: Multi-step reasoning for robotic planning, 2025. URL <https://deepmind.google/technologies/gemini/>. Accessed: 2026-01-21.
- [17] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, et al. PaLM-E: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.
- [18] 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.
- [19] Kuan Fang, Alexander Toshev, Li Fei-Fei, and Silvio Savarese. Scene memory transformer for embodied agents in long-horizon tasks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 538–547, 2019.
- [20] Yunhai Feng, Jiaming Han, Zhuoran Yang, Xiangyu Yue, Sergey Levine, and Jianlan Luo. Reflective planning: Vision-language models for multi-stage long-horizon robotic manipulation, 2025. URL <https://arxiv.org/abs/2502.16707>.
- [21] Figure AI Team. Helix: A vision-language-action model for generalist humanoid control. *Technical Report*, 2025.
- [22] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning (ICML)*, pages 1126–1135, 2017.
- [23] Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning. In *Proceedings of the Conference on Robot Learning (CoRL)*, pages 357–368, 2017.
- [24] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia,

- Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.
- [25] Gemini Robotics Team, Abbas Abdolmaleki, Anthony Brohan, Noah Brown, Konstantinos Bousmalis, Chelsea Finn, Karol Hausman, Sergey Levine, et al. Gemini robotics 1.5: Pushing the frontier of generalist robots with advanced embodied reasoning, thinking, and motion transfer. *arXiv preprint arXiv:2510.03342*, 2025.
- [26] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.
- [27] Jialei Huang, Shuo Wang, Fanqi Lin, Yihang Hu, Chuan Wen, and Yang Gao. Tactile-VLA: Unlocking vision-language-action model’s physical knowledge for tactile generalization. *arXiv preprint arXiv:2507.09160*, 2025.
- [28] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. In *Proceedings of the Conference on Robot Learning (CoRL)*, pages 1769–1782, 2022.
- [29] Kento Kawaharazuka, Jihoon Oh, Jun Yamada, Ingmar Posner, and Yuke Zhu. Vision-language-action models for robotics: A review towards real-world applications. *IEEE Access*, 13:162467–162504, 2025.
- [30] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Chelsea Finn, Sergey Levine, and Percy Liang. OpenVLA: An open-source vision-language-action model. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2024.
- [31] Sungjun Kim, Gyeongrok Oh, Heeju Ko, Daehyun Ji, Dongwook Lee, Byung-Jun Lee, Sujin Jang, and Sangpil Kim. Test-time adaptation for online vision-language navigation with feedback-based reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2025.
- [32] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv:2304.02643*, 2023.
- [33] Michael Laskin, Luyu Wang, Junhyuk Oh, Emilio Parisotto, Stephen Spencer, Richie Steiber, DJ Strouse, Steven Hansen, Angelos Fiez, Max Simchowitz, et al. In-context reinforcement learning with algorithm distillation. In *International Conference on Learning Representations (ICLR)*, 2023.
- [34] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 9459–9474, 2020.
- [35] Hongxin Li, Zeyu Wang, Xu Yang, Yuran Yang, Shuqi Mei, and Zhaoxiang Zhang. MemoNav: Working memory model for visual navigation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17913–17922, 2024.
- [36] Runhao Li, Wenkai Guo, Zhenyu Wu, Huazhe Xu, et al. MAP-VLA: Memory-augmented prompting for vision-language-action model in robotic manipulation. *arXiv preprint arXiv:2511.09516*, 2025.
- [37] Xinghang Li, Peiyan Li, Minghuan Liu, Dong Wang, Jirong Liu, Bingyi Kang, Xiao Ma, Tao Kong, Hanbo Zhang, and Huaping Liu. Towards generalist robot policies: What matters in building vision-language-action models. *arXiv preprint arXiv:2412.14058*, 2024.
- [38] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500, 2023.
- [39] Bo Liu, Xuesu Xiao, and Peter Stone. A lifelong learning approach to mobile robot navigation. *IEEE Robotics and Automation Letters*, 6(2):1090–1097, 2021.
- [40] Jiaxun Liu and Boyuan Chen. SonicSense: Object perception from in-hand acoustic vibration. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2024.
- [41] Yanjiang Luo, Zhecheng Wang, Xiaoyu Zhang, Zhixuan Xu, Zhengrong Lu, Yanjie Qu, and Huazhe Xu. Improving vision-language-action model with online reinforcement learning. *arXiv preprint arXiv:2501.01734*, 2025.
- [42] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems (NeurIPS)*, 36, 2023.
- [43] Yuan Meng, Zhenshan Bing, Xiangtong Yao, Kejia Chen, Kai Huang, Yang Gao, Fuchun Sun, and Alois Knoll. Preserving and combining knowledge in robotic lifelong reinforcement learning. *Nature Machine Intelligence*, 2025.
- [44] 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.
- [45] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreber, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.
- [46] OpenAI. GPT-5.1: Advanced multimodal reasoning model, 2025. URL <https://openai.com/gpt-5>.
- [47] Physical Intelligence Team, Kevin Black, Noah Brown, Chelsea Finn, Karol Hausman, Brian Ichter, Sergey Levine, Karl Pertsch, Lucy Xiaoyang Shi, et al. $\pi_{0.6}^*$:

- A VLA that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025.
- [48] Karl Popper. *The Logic of Scientific Discovery*. Routledge, 1959.
 - [49] Alexander Pritzel, Benigno Uria, Sriram Srinivasan, Adrià Puigdomènech, Oriol Vinyals, Demis Hassabis, Daan Wierstra, and Charles Blundell. Neural episodic control. In *International Conference on Machine Learning (ICML)*, pages 2827–2836, 2017.
 - [50] Yiming Qin, Bomin Wei, Jiaxin Ge, Konstantinos Kallidromitis, Stephanie Fu, Trevor Darrell, and Xudong Wang. Chain-of-visual-thought: Teaching vlms to see and think better with continuous visual tokens. *arXiv preprint arXiv:2511.19418*, 2025.
 - [51] Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xianguyu Zhang, and Gao Huang. Memoryvla: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2508.19236*, 2025.
 - [52] Lucy Xiaoyang Shi, Zheyuan Hu, Tony Z. Zhao, Archit Sharma, Karl Pertsch, Jianlan Luo, Sergey Levine, and Chelsea Finn. Yell at your robot: Improving on-the-fly from language corrections. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.
 - [53] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 36, 2023.
 - [54] SIMA Team, Adrian Bolton, Alexander Lerchner, et al. SIMA 2: A generalist embodied agent for virtual worlds. *arXiv preprint arXiv:2512.04797*, 2025.
 - [55] 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.
 - [56] Venkatesh Sripada, Samuel Carter, Frank Guerin, and Amir Ghalamzan. AP-VLM: Active perception enabled by vision-language models. *arXiv preprint arXiv:2409.17641*, 2024.
 - [57] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International Conference on Machine Learning (ICML)*, pages 9229–9248. PMLR, 2020.
 - [58] Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999.
 - [59] Tongxuan Tian, Haoyang Li, Bo Ai, Xiaodi Yuan, Zhiao Huang, and Hao Su. Diffusion dynamics models with generative state estimation for cloth manipulation. *Conference on Robot Learning (CoRL)*, 2025.
 - [60] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, 2017.
 - [61] Marcel Torne, Karl Pertsch, Homer Walke, Kyle Vedder, Suraj Nair, Brian Ichter, Allen Z. Ren, Haohuan Wang, Jiaming Tang, Kyle Stachowicz, Karan Dhabalia, Michael Equi, Quan Vuong, Jost Tobias Springenberg, Sergey Levine, Chelsea Finn, and Danny Driess. Mem: Multi-scale embodied memory for vision language action models. *arXiv preprint arXiv:2603.03596*, 2026.
 - [62] Shoukai Xu, Minghui Tan, Liu Liu, Zhong Zhang, Peilin Zhao, et al. Test-time adapted reinforcement learning with action entropy regularization. In *Forty-second International Conference on Machine Learning*.
 - [63] Minjong Yoo, Jinwoo Jang, Sihyung Yoon, and Honguk Woo. World model implanting for test-time adaptation of embodied agents. In *International Conference on Machine Learning (ICML)*, 2025.
 - [64] Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2024.
 - [65] Jesse Zhang, Minh Heo, Zuxin Liu, Erdem Biyik, Joseph J. Lim, Yao Liu, and Rasool Fakoor. EXTRACT: Efficient policy learning by extracting transferable robot skills from offline data. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2024.
 - [66] Jianke Zhang, Xiaoyu Chen, Qiuyue Wang, Mingsheng Li, Yanjiang Guo, Yucheng Hu, Jiajun Zhang, Shuai Bai, Junyang Lin, and Jianyu Chen. Vlm4vla: Revisiting vision-language-models in vision-language-action models. *arXiv preprint arXiv:2601.03309*, 2026.
 - [67] Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, et al. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 1702–1713, 2025.
 - [68] Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, et al. CoT-VLA: Visual chain-of-thought reasoning for vision-language-action models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

APPENDIX A METHOD IMPLEMENTATION DETAILS

This appendix provides complete algorithmic details for PhysMem that were omitted from the main paper due to space constraints.

A. Experience Data Structure

Each experience e in episodic memory contains the following fields:

- **eid**: Unique identifier
- **observation**: Visual observation (image or keyframes)
- **action**: Selected option/action
- **outcome**: Success (1) or failure (0)
- **context**: Task description, subtask identifier
- **symbolic_state**: Discrete features for filtering:
 - **action_type**: e.g., “pick”, “insert”, “push”
 - **object_properties**: shape, size, material
 - **dependencies_satisfied**: boolean
 - **progress**: task completion fraction
- **resonance_score**: Alignment with active principles
- **fail_tag**: Failure category (if applicable)
- **timestamp**: Episode and step indices

B. Hypothesis Generation

Prompt Template. The reflection model receives:

- 1) 3–5 sample experiences from the cluster
- 2) Extracted patterns (action types, object properties, outcomes)

- 3) Existing principles and hypotheses (to avoid duplication)
- 4) Task context

Output Format. Each hypothesis includes:

- **Type**: AVOID, PREFER, SEQUENCE, etc.
- **Statement**: Natural language rule
- **Applicable Actions**: When this rule applies
- **Trigger Conditions**: Contextual requirements

C. Experience Clustering

We use hierarchical agglomerative clustering on symbolic state features:

- 1) Extract text embedding from symbolic state
- 2) Compute pairwise cosine similarity
- 3) Apply hierarchical clustering with threshold $\tau = 0.6$
- 4) Retain clusters with $\geq n_{\min} = 2$ experiences

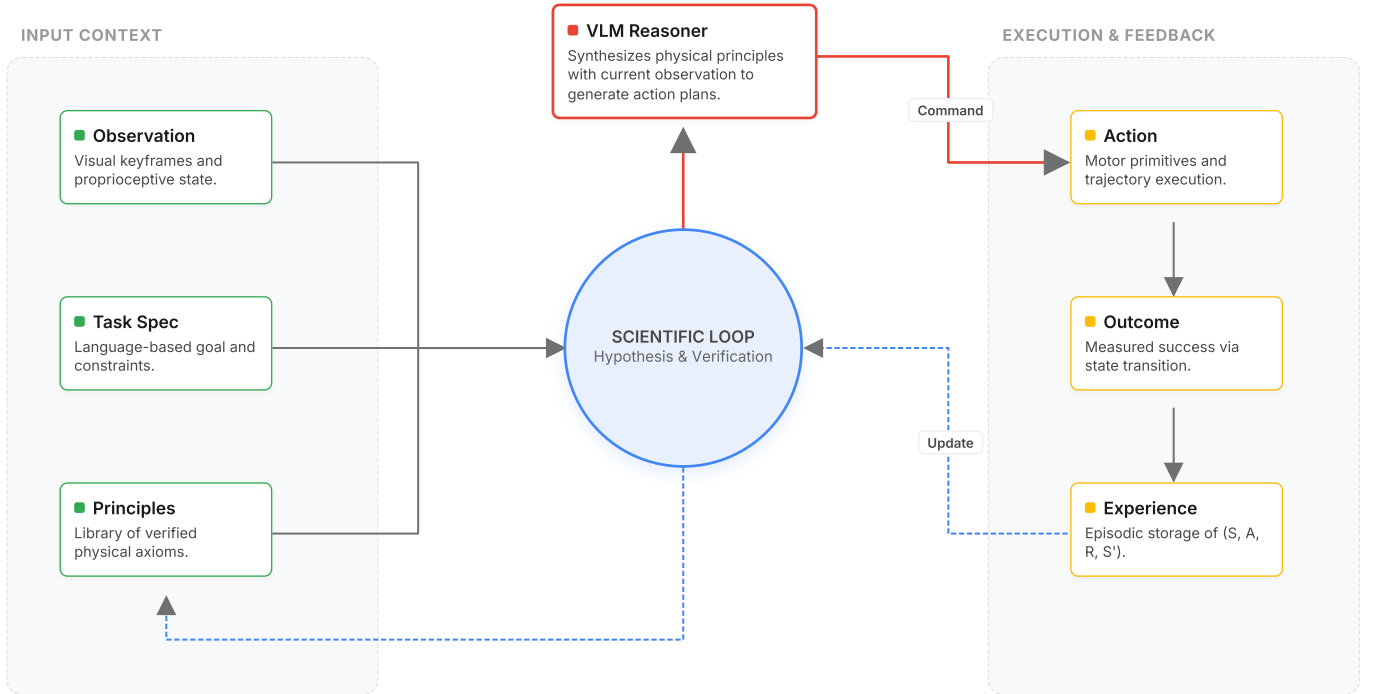


Fig. 11: **PhysMem overall pipeline.** PhysMem enables VLM robot planners to learn physical principles through interaction. The scientific loop transforms raw experiences into verified knowledge: collecting observations, generating hypotheses, verifying through action-level attribution, and promoting validated principles to guide future decisions.

D. Confidence Update Rules

The action-level attribution updates hypothesis confidence as follows:

$$\text{conf}(h) \leftarrow \begin{cases} \min(1.0, \text{conf}(h) + 0.1 \cdot r_a) & \text{if } r_a \geq 0.7 \\ \max(0.0, \text{conf}(h) - 0.1 \cdot (1 - r_a)) & \text{if } r_a \leq 0.3 \\ \text{conf}(h) \pm 0.02 & \text{otherwise} \end{cases} \quad (7)$$

where r_a is the action-level success rate for actions matching the hypothesis.

Promotion Criteria.

- Confidence ≥ 0.8
- Supporting episodes ≥ 3
- Accuracy $\geq 85\%$ (supporting / total)

Refutation Criteria.

- Confidence ≤ 0.3
- Contradicting episodes ≥ 2

E. Memory Management

Capacity and Forgetting. Episodic memory is bounded at $N_{\max} = 3000$ experiences. When capacity is reached, garbage collection removes:

- 1) Folded experiences older than TTL (100 episodes)
- 2) Oldest experiences by priority (folded > old failures > old successes)

Principle Decay. Principles decay following an Ebbinghaus forgetting curve:

$$\text{score}_{t+1} = \text{score}_t \cdot \gamma, \quad \gamma = 0.995 \quad (8)$$

This yields approximately 50% retention after 138 episodes without reinforcement.

F. Hyperparameter Summary

Table V lists all hyperparameters used in our experiments.

TABLE V: Hyperparameter settings.

Parameter	Value	Description
<i>Memory Capacity</i>		
$N_{\max}$	3000	Max episodic experiences
Folded TTL	100	Episodes before folded exp. removal
<i>Consolidation</i>		
Interval	50	Episodes between consolidation
Min cluster size	2	Experiences per cluster
Similarity threshold	0.6	Clustering threshold
Max hypotheses/cluster	3	Hypothesis generation cap
<i>Principle Management</i>		
Promotion threshold	0.8	Confidence for promotion
Refutation threshold	0.3	Confidence for refutation
Min supporting	3	Episodes for promotion
Max in prompt	5	Principles injected
Decay factor γ	0.995	Ebbinghaus decay

G. Examples of Refuted Hypotheses

The examples below are drawn from candidate hypotheses that triggered the refutation criterion ($\text{conf}(h) \leq \tau_r = 0.3$ with $|\mathcal{E}_{\text{contradict}}| \geq 2$) and were therefore removed before they could condition the planner. Each entry was plausible at generation time—it was generated from a cluster of two or more experiences sharing a symbolic-state pattern—but proved empirically wrong in the deployment context once verification accumulated contradicting evidence. These cases show that verification prunes plausible-but-incorrect hypotheses rather than only catching obvious errors.

Parts Organization. Two refuted hypotheses from real-world deployment:

```
## [REFUTED] Hypothesis (from 3 experiences)

Type:          PREFER
Statement:     PREFER placing q-shaped parts at
               rotation=0; rotation=90 leads to
               wasted cells.
Generated:     episode 4
Cluster:       3 experiences, all q-shaped parts
               placed at rotation=0 with
               successful packing
Initial conf:  0.62
Final conf:    0.18 (refuted at episode 9)
Lifespan:      5 episodes

Why plausible at generation time:
Three early successful episodes happened to use
rotation=0 for white-q and red-q.
The cluster shared symbolic state (action=place
, shape=q, rotation=0,
outcome=+1), and the reflection model
abstracted "rotation=0 is safer for
q-shapes."

Contradicting evidence (sample):
- Experience #34: red-q at rotation=0 collided
  with adjacent white-q's [a,c]
  region (-1). The right-aligned internal edge
  of red-q overlapped with
  white-q's left edge.
- Experience #41: white-q at rotation=180
  succeeded by inverting its [c]
  region toward an empty cell, enabling tighter
  packing than rotation=0
  would have allowed.

Why wrong in context:
Optimal q-rotation depends on the neighbor's
edge alignment, not on a global
rotation preference. The early cluster
confounded "rotation=0 worked" with
"the neighbors happened to be empty cells." The
correct rule (later promoted)
conditions on the adjacent part's [a,c]/[b,d]
occupancy.
```

[REFUTED] Hypothesis (from 2 experiences)

Type: SEQUENCE
Statement: SEQUENCE: place black-I before any U-shaped part to reserve corner cells.
Generated: episode 3
Initial conf: 0.55
Final conf: 0.22 (refuted at episode 8)
Lifespan: 5 episodes

Why plausible at generation time:
Two early high-scoring episodes happened to place black-I first; the small contact footprint left more flexibility for later U-shape placement.

Contradicting evidence (sample):
- Experience #29: Placing black-I first forced white-U into the center, blocking red-q from interlocking on the boundary. Final score: -0.4 (sub-optimal packing).
- Experience #37: Placing white-U first (rotation=180) and slotting black-I into the U's empty [a,c] region produced score 9.2, the best of the run.

Why wrong in context:
The two early successes were a small-sample artifact; the real constraint is that 4-cell parts (U-shapes) have less placement flexibility than 2-cell parts (I-shapes), so U-shapes should be placed first. The refuted hypothesis inverted the correct size-constrained ordering. The contradicting evidence directly informed the later SEQUENCE principle "place 4-cell parts before 2-cell parts."

Ball Navigation. Two refuted hypotheses from real-world deployment:

[REFUTED] Hypothesis (from 2 experiences)

Type: PREFER
Statement: PREFER high speed when ball is in Zone A (initial), regardless of distance to OBS1.
Generated: episode 2
Initial conf: 0.60
Final conf: 0.15 (refuted at episode 7)
Lifespan: 5 episodes

Why plausible at generation time:
Two early Zone A pushes used high speed and successfully reached the archway region. The cluster (action=push, zone=A, speed=high, outcome=+3) suggested a simple zone-to-speed mapping.

Contradicting evidence (sample):
- Experience #18: Ball at (y=350, x=420), ~150 px from OBS1 archway. High-speed push caused the ball to overshoot through the archway and land on top of OBS3 (y=680). Stuck state, episode terminated. Score: -1.
- Experience #24: Ball at (y=380, x=380), ~120 px from archway. High-speed diagonal push rebounded off OBS1 wall (missed bridge hole by ~30px). Score: -2.

Why wrong in context:
Zone alone is not the right conditioning variable; distance to OBS1 archway matters. The two early successes happened at >200px distance where high speed is acceptable. The correct rule (later promoted as a verified principle) is distance-conditional: >200px -> high; 100-200px -> medium; <100px -> low. The refuted hypothesis would have caused systematic OBS3-stuck failures had it been promoted.

[REFUTED] Hypothesis (from 3 experiences)

Type: AVOID
Statement: AVOID end_direction containing "bottom" after passing through the archway.
Generated: episode 5
Initial conf: 0.58
Final conf: 0.20 (refuted at episode 11)
Lifespan: 6 episodes

Why plausible at generation time:
Three post-archway pushes with "bottom" or "bottom-left" end directions resulted in the ball drifting toward the OBS3 danger zone (y > 550). The reflection model generalized "bottom = danger after archway."

Contradicting evidence (sample):
- Experience #44: Post-archway push with end_direction="bottom-right" at LOW speed successfully descended to target without entering the OBS3 zone. Score: +5.
- Experience #51: end_direction="bottom" at LOW speed from (y=430, x=520) reached the target in one step. Score: +9 (target reached with 2 steps remaining out of 6).

Why wrong in context:
The three original experiences that supported the hypothesis all used HIGH/MEDIUM speed, confounding direction with speed. Once speed is held fixed, "bottom" directions at LOW speed are in fact preferred for target approach (as the contradicting cases above show). The refuted hypothesis would have pushed the planner toward inefficient lateral pushes; the correct rule conditions on speed, not on direction alone.

Balanced Stacking. Two refuted hypotheses from real-world deployment:

[REFUTED] Hypothesis (from 3 experiences)

Type: PREFER
Statement: PREFER selecting the largest stone as the base, regardless of surface material.
Generated: episode 2
Initial conf: 0.65
Final conf: 0.18 (refuted at episode 8)
Lifespan: 6 episodes

Why plausible at generation time:
Three early towers used the largest available stone as the base and reached at least 3 layers before any collapse. The cluster (step=1, size=large, outcome=+1) suggested size alone determines base suitability.

Contradicting evidence (sample):
- Experience #21: PAINT_SQR_L_C (largest stone, painted/smooth surface) selected as base. Second stone (3M_SQR_M_C) slid off during placement. Tower collapsed at step 2. Score: -2.
- Experience #28: WOOD_DMND_L_H (large but horizontal-bar geometry) selected as base. Stable for one stone, collapsed when third stone was placed off-center. Score: -4.

Why wrong in context:
Size is correlated with stability but not causal -- friction class and contact area matter more. The three early successes coincidentally had high-friction surfaces (3M, BLK) on the largest stones. The correct rule (later promoted) is conjunctive: "largest stone WITH highest friction class (3M > BLK > WHT > WOOD > PAINT) AND flat compact orientation." The refuted hypothesis would have chosen PAINT_SQR_L_C or WOOD_DMND_L_H as the base, both of which are high collapse risk.

[REFUTED] Hypothesis (from 2 experiences)

Type: AVOID
Statement: AVOID placing HEX-shaped stones at any layer; their small contact area is unstable.
Generated: episode 4
Initial conf: 0.50
Final conf: 0.25 (refuted at episode 9)
Lifespan: 5 episodes

Why plausible at generation time:
Two episodes had HEX stones placed mid-tower (step 2 or 3) and collapsed.
The reflection model attributed collapse to the hexagonal cross-section's reduced contact area.

Contradicting evidence (sample):
- Experience #33: 3M_HEX_L_C placed at step 4 (top layer) on a stable 3-layer base. Tower remained stable; final score +4.
- Experience #39: BLK_HEX_M_C placed at step 5 (top) on a 4-layer tower.
No collapse; score +5.

Why wrong in context:
HEX instability is layer-dependent, not absolute. At lower layers HEX cross-sections cannot bear weight from above; at top layers there is no weight to bear, so the small contact is fine.
The correct rule (later promoted) is a SEQUENCE constraint: "place HEX stones at steps 4-5, not earlier." The blanket AVOID would have eliminated viable top-layer placements.

Brick Insertion (simulation). Two refuted hypotheses from the Reflect-VLM brick insertion benchmark:

[REFUTED] Hypothesis (from 4 experiences, hard difficulty)

Type: AVOID
Statement: AVOID inserting blue before red.
Generated: episode 6 (difficulty: hard)
Initial conf: 0.70
Final conf: 0.20 (refuted at episode 14)
Lifespan: 8 episodes

Why plausible at generation time:
On hard difficulty, four early episodes had insert(blue) attempts fail when red was still on the table. The cluster (action=insert, target=blue, red_remaining=true, outcome=fail) suggested a direct order constraint between blue and red.

Contradicting evidence (sample):
- Experience #67: Inserted blue successfully while red was still on the table; the blocker was actually green, which had been relocated in the prior step. blue-before-red worked once green was out of the way.
- Experience #74: insert(red) BEFORE blue also failed, because green still blocked red's slot. The blue/red order had no causal role in either direction.

Why wrong in context:
The four early failures all coincidentally had green in a position that blocked blue's slot. Symbolic-state clustering matched on (target=blue, red_remaining=true) but missed the true blocker (green). The correct rule (later promoted) is "AVOID insert(X) when target_blocked_by includes any color Y," conditioning on the actual blocking variable rather than the spurious blue-red correlation. This is a textbook confounding-variable failure that verification correctly caught.

```

## [REFUTED] Hypothesis (from 2 experiences,
medium difficulty)

Type:          PREFER
Statement:     PREFER reorient before every
               insert action for elongated bricks.
Generated:     episode 3    (difficulty: medium)
Initial conf:  0.55
Final conf:    0.22    (refuted at episode 10)
Lifespan:      7 episodes

Why plausible at generation time:
Two early episodes had failed insert attempts
for elongated bricks that
succeeded after a reorient action. The
reflection model abstracted
"elongated bricks always need reorientation."

Contradicting evidence (sample):
- Experience #45: Elongated yellow brick picked
  up in correct orientation
  directly from the table; insert succeeded
  without reorient. Adding a
  reorient would have consumed an action step
  unnecessarily.
- Experience #52: After reorient on an already-
  correctly-oriented brick,
  the brick ended up MIS-oriented (rotated 90
  degrees away from target).
  Subsequent insert failed; required a second
  reorient to recover.

Why wrong in context:
Reorientation is conditional on the brick's
pickup orientation, not on its
shape. Mandatory reorient introduces failures
when the brick was already
correct. The correct rule (promoted later)
conditions on a quaternion-error
check between the held brick's orientation and
the target slot orientation,
not on shape category. This refuted hypothesis
was recovered by the
failure-mode category "over-generalization" (
  Section \ref{app:failure-modes}):
a narrow-context observation expanded into a
universal rule.

```

Aggregate refutation statistics. Across the four task suites, verification refuted approximately 31% of generated hypotheses (87 of 281 across all real-world and simulation runs) before they could condition the planner. Refuted hypotheses had a median lifespan of 6 episodes (IQR 4–9) and a median initial confidence of 0.58, indicating that they were not obviously bad at generation time—they accumulated contradicting evidence only after targeted interaction tested their predictions. Without verification (Table IV, *w/o Verification*: –12% on medium), these plausible-but-incorrect hypotheses would have been promoted to principles and degraded the planner’s decisions; their refutation is the load-bearing function of the verification mechanism.

TABLE VI: **Complete ablation results across difficulty levels.** Success rates and token consumption on simulation benchmark (100 episodes each). Token consumption is normalized relative to the full system at each difficulty level. Δ indicates absolute change from the full system. Best accuracy in **bold**.

Configuration	Easy (2–3 bricks)			Medium (4–5 bricks)			Hard (6–8 bricks)		
	Success	Δ	Tokens	Success	Δ	Tokens	Success	Δ	Tokens
PhysMem (Full)	89%	—	1.0×	76%	—	1.0×	39%	—	1.0×
<i>Memory Architecture Ablations</i>									
w/o Episodic Memory	54%	−35	0.3×	37%	−39	0.25×	14%	−25	0.2×
w/o Working Memory	84%	−5	0.85×	69%	−7	0.8×	28%	−11	0.75×
w/o Long-term Memory	81%	−8	1.15×	64%	−12	1.25×	26%	−13	1.35×
<i>Mechanism Ablations</i>									
w/o Resonance Filtering	81%	−8	1.15×	58%	−18	1.3×	21%	−18	1.45×
w/o Verification	85%	−4	0.9×	64%	−12	0.85×	27%	−12	0.8×
w/o Forgetting	91%	+2	1.8×	78%	+2	3.4×	36%	−3	4.8×
w/o Folding	88%	−1	1.4×	74%	−2	2.1×	37%	−2	2.8×
<i>Simplified Baselines</i>									
Direct Retrieval	48%	−41	0.45×	23%	−53	0.55×	8%	−31	0.50×
Only Episodic	52%	−37	0.6×	28%	−48	0.7×	10%	−29	0.65×
Only Principles	67%	−22	0.4×	51%	−25	0.35×	21%	−18	0.3×

APPENDIX B COMPLETE ABLATION STUDIES

We conducted comprehensive ablations to validate each component of PhysMem across all three difficulty levels. Each configuration runs 100 episodes using Gemini-3-Flash with thinking mode. We report both success rate and relative token consumption (normalized to the full system at each difficulty).

A. Full Results

Table VI presents complete ablation results with token consumption analysis.

B. Analysis by Component

1) *Memory Architecture Ablations:* **Episodic memory** is the load-bearing tier. Without it, raw experiences are not stored at all and the system has nothing to draft hypotheses from; performance collapses to 54%, 37%, and 14% across the three difficulties (about 25 to 39 points below the full system). Token usage drops to 0.2–0.3× for the trivial reason that nothing is being consolidated.

Working memory holds hypotheses while they are being tested, and its value grows with task complexity (−5% on easy, −7% on medium, −11% on hard). On easy tasks the right move is usually obvious enough that hypothesis exploration adds little; on hard tasks the planner needs to actively test patterns like “does brown block pink?” across several episodes, which is exactly what working memory makes possible.

Long-term memory is what lets verified hypotheses persist as principles between episodes. Without it the system rediscovers the same patterns every run, which costs −8% on easy and grows to −12% to −13% on medium and hard, where complex dependency chains depend on stable principles. Tokens climb to 1.15–1.35× because the model is generating hypotheses repeatedly that ought to have been settled already.

2) *Mechanism Ablations:* **Resonance filtering** retrieves principles based on prediction–outcome alignment rather than raw similarity, and like working memory it grows in importance with difficulty (−8% on easy, −18% on medium and hard). On harder tasks more principles accumulate (especially AVOID constraints from failures), and without filtering the planner receives a steady stream of conflicting guidance.

Verification checks each hypothesis against later experiences before letting it shape the prompt, and it matters most when hypotheses get harder to vet by inspection (−4% on easy, −12% on medium and hard). Easy hypotheses tend to be self-evidently right or wrong from a single rollout; medium and hard hypotheses are about partial orderings and dependency chains, where guessing wrong propagates downstream.

Forgetting (Ebbinghaus-style decay of old principles) reveals an accuracy–efficiency trade-off that depends on difficulty. Disabling it gives a small bump on easy and medium (+2% each) but at 1.8× to 3.4× the tokens. On hard tasks the accumulated noise actually *hurts* accuracy (−3% at 4.8× tokens), because stale principles begin competing with newer, more accurate ones.

Folding compresses redundant experiences into representative clusters. Removing it costs almost nothing in accuracy (−1% to −2%) but multiplies token cost by 1.4× to 2.8× across difficulties, so its value is efficiency rather than accuracy.

3) *Simplified Baselines:* **Direct Retrieval** stores raw experiences and pulls the nearest one by state-embedding distance. It fails across the board (48%, 23%, 8% across easy/medium/hard), and the failure mode is the same in every case: small state differences make the retrieved action a bad fit for the new scene, which is exactly the slot abstraction is meant to fill.

Only Episodic. Keeping episodic memory but skipping hypotheses and principles is only marginally better than Direct Retrieval (52%, 28%, 10%). Richer experience records help a

little; the gap from the full system (−37 to −48) is again the size of what the symbolic abstraction layer adds.

Only Principles. Pre-loading a fixed principle set without any test-time adaptation reaches moderate accuracy (67%, 51%, 21%) on tasks where the prior generalizes well, but it cannot recover from a deployment that drifts away from the prior. The 18- to 25-point gap to the full system is the value of adaptation on top of a good prior.

C. Summary of the Ablation

Five patterns come out of the table together. The three-tier memory is the only configuration that holds up across difficulties: removing any tier (episodic, working, or long-term) costs anywhere from 5 to 39 points, with episodic memory carrying the largest weight (−35 to −39) and working and long-term memory growing in importance on harder tasks. Principle abstraction does most of the lifting; the two baselines that fall back on raw experience (Direct Retrieval and Only Episodic) drop 37–53 points on medium and hard, which is the size of what the abstraction layer adds. Within the mechanism, resonance filtering, verification, and working memory each grow more important as tasks get harder, and the relative ordering is stable across difficulties. Forgetting trades a small accuracy cost for a 3–5 $\times$ token saving, and on hard tasks it actually helps because stale principles begin to interfere with newer ones. Finally, a fixed prior alone (Only Principles) covers a fair amount of ground but cannot recover when the deployment drifts from the prior, leaving an 18- to 25-point gap that test-time adaptation closes.

APPENDIX C

ROBUSTNESS, FAILURE MODES, AND RUNTIME

A. Sensitivity Analysis

PhysMem’s design includes three threshold parameters: a promotion threshold τ_p (default 0.8), a clustering threshold τ_c (default 0.7), and a decay rate γ (default 0.995). On medium-difficulty brick insertion (Gemini-3-Flash, 100 episodes per condition):

- $\tau_p \in \{0.6, 0.7, 0.8, 0.9\}$: 71%, 74%, 76%, 73%. Optimum is broad; $[0.7, 0.9]$ stays within 3% of default.
- $\tau_c \in \{0.5, 0.6, 0.7, 0.8\}$: 72%, 75%, 76%, 74%. Insensitive within realistic range.
- $\gamma \in \{0.99, 0.995, 0.999, 1.0\}$: 74%, 76%, 77%, 73%. Mild decay helps; no-decay underperforms because stale principles compete with newer ones.

Across $4 \times 4 \times 4 = 64$ threshold configurations, 59 land within 5% of the default (76%), and even the worst (66%) still beats every non-PhysMem baseline in the main paradigm comparison, so the qualitative findings do not depend on a particular choice of thresholds.

B. Failure Mode Analysis

We manually categorize the 24% of medium-difficulty failures (24/100 episodes) into three modes.

Over-generalization (40%). A principle that was committed from a narrow context gets re-applied where it no longer

holds. For example, an AVOID principle learned for thin bricks (“do not place piece B before piece A”) is retrieved for a wide-brick scene where the order constraint is reversed. Stronger context conditioning during retrieval would help here; the current symbolic-state matcher catches object type but not always size.

Delayed correction (35%). A principle that has stopped being true (a friction estimate from earlier episodes, say) is only refuted after several contradicting experiences accumulate, a 2–4 episode lag on average. Lowering the refutation threshold τ_r or weighting contradicting evidence more heavily would shorten the lag, at the cost of over-correcting.

Planner–principle mismatch (25%). The principle is correct but the VLM planner ignores or misapplies it, which is a prompt-formatting issue rather than a memory issue. Tighter principle templates in the planner prompt or a structured-output planner head are the natural fixes.

The first two categories (75% together) are addressable by refining the memory mechanism. The third sits at the limit of what frozen-base prompting can do and would need execution-layer integration to fix.

C. Runtime Overhead

PhysMem adds memory operations on two timescales. Per-step retrieval (symbolic filtering, semantic ranking, prompt injection) adds about 180 ms per planning step on a single workstation, dominated by the VLM-based relevance judgment. Asynchronous consolidation (clustering, hypothesis drafting, action-conditioned confidence updates) runs in a background thread every 15 s with an average wall-clock cost of 1.4 s per cycle and does not block the planner. Total wall-clock overhead averages around 8% of base VLM inference time on the brick-insertion benchmark and tracks the same range on the real-world tasks. Token overhead is reported in the main paradigm comparison (1.0 $\times$ for PhysMem versus 0.55 $\times$ for Direct Retrieval and 2.5 $\times$ for MemoryVLA-style).

APPENDIX D

HYPOTHESIS AND PRINCIPLE QUANTITY ANALYSIS

We investigated the optimal number of hypotheses to generate per cluster and principles to include in prompts.

TABLE VII: **Quantity configuration results.** H/C = hypotheses per cluster, P/P = principles in prompt. Evaluated on medium difficulty (100 episodes each).

Config	H/C	P/P	Success	Principles	Tokens
Sparse	1	3	71%	12	$0.7\times$
Balanced	2	5	76%	18	$1.0\times$
Dense	3	10	74%	31	$1.6\times$
Hypo-Heavy	3	3	73%	26	$1.2\times$
Principle-Heavy	1	10	72%	14	$1.1\times$

Analysis. The results show moderate variance across configurations (71–76%), with the “balanced” configuration (2 hypotheses per cluster, 5 principles in prompt) achieving the best performance. Sparse configurations under-explore the hypothesis space, while dense configurations introduce noise from unverified hypotheses. The balanced setting provides a favorable trade-off between exploration coverage and computational cost ($1.0\times$ baseline tokens).

APPENDIX E

SIMULATION EXPERIMENT DETAILS

This appendix provides comprehensive details for our simulation-based experiments using the Brick Assembly environment. The simulation enables rapid, reproducible evaluation of PhysMem across many episodes with ground-truth feedback.

A. Environment Overview

We use the Reflect-VLM simulation environment [20], built on MuJoCo physics simulation with a Franka Emika Panda robot arm.

Simulator Configuration.

- **Physics Engine:** MuJoCo 3.x with realistic contact dynamics
- **Robot:** Franka Panda 7-DOF arm with parallel-jaw gripper
- **Control:** Differential Inverse Kinematics with nullspace control
- **Timestep:** 1ms simulation, 20ms control frequency (frame skip = 20)
- **Rendering:** 1280×720 RGB images, offscreen rendering

B. Brick Insertion: Brick Assembly

The brick assembly task requires the robot to insert multiple colored bricks into a board fixture, testing dependency reasoning and sequential planning.

1) *Task Description: Setup.* A board fixture with insertion slots is placed on the table, surrounded by 2–5 colored bricks with varying shapes. Each brick must be inserted into its designated slot on the board. Bricks are procedurally generated with:

- **Shape variation:** Rectangular blocks, elongated beams, L-shapes, and nail-like pegs
- **Size variation:** 3–25 voxel units in length, 3–8 in width
- **Color coding:** 9 distinct colors (red, blue, green, yellow, orange, purple, brown, pink, gray)
- **Slot features:** Through-slots requiring specific orientations

Objective. Insert all bricks into their designated board slots. Success requires each brick to be within 5mm of its target position and correctly oriented (quaternion error < 0.02 radians).

Challenge. The task is challenging because: (1) bricks have **dependency constraints**, where some bricks physically block insertion of others until removed, (2) bricks may land in incorrect orientations requiring **reorientation**, (3) the VLM must infer the **correct insertion sequence** from visual observation of spatial relationships, and (4) failed insertions provide only sparse error signals without explicit explanation of the blocking relationship.

Dependency Graph. Dependencies are computed from voxel spatial intersections: if brick *A*’s insertion path intersects brick *B*’s current position, then *B* must be removed before *A* can be inserted. This creates a directed acyclic graph of assembly constraints that the VLM must implicitly discover through trial and error.

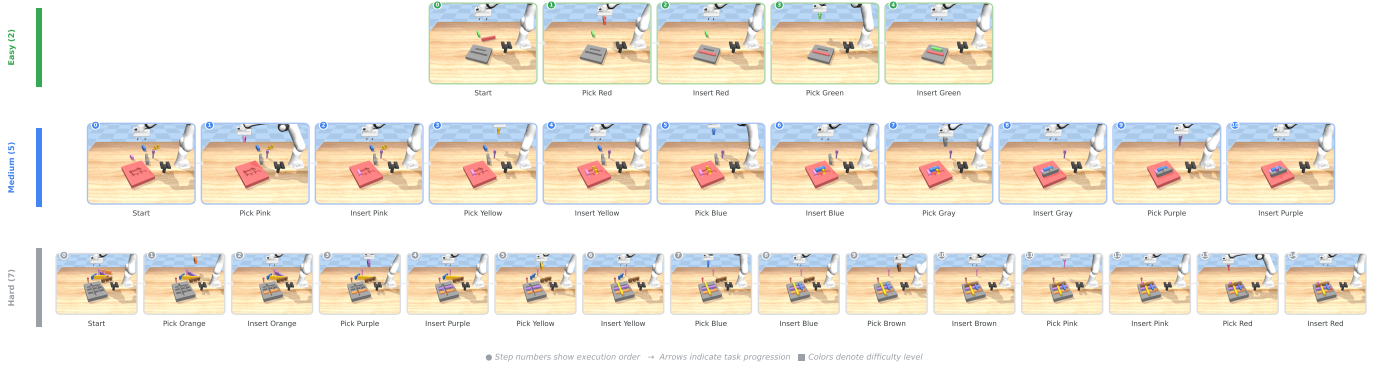


Fig. 12: **Assembly task demonstration across difficulty levels.** Each row shows the oracle planner’s execution sequence from initial state to completion. Step numbers indicate execution order. **Easy** (green, 2 bricks): 5 steps total, minimal dependencies. **Medium** (blue, 5 bricks): 11 steps with moderate dependency chains. **Hard** (gray, 7 bricks): 15 steps requiring complex ordering to avoid blocking. The VLM must learn correct insertion sequences through trial and error, as inserting bricks in the wrong order causes failures that provide only sparse error signals.

Difficulty Levels. We evaluate across three difficulty levels based on brick count and dependency complexity (Figure 12):

- **Easy (2–3 bricks):** Minimal dependencies, typically requiring 5–7 pick-and-place steps. Most orderings succeed.
- **Medium (4–5 bricks):** Moderate dependency chains where 2–3 bricks may block others. Requires 9–11 steps and partial ordering constraints.
- **Hard (6–8 bricks):** Complex dependency graphs with multiple blocking relationships. Requires 13–17 steps; incorrect orderings lead to repeated failures until the VLM learns the constraints.

2) **VLM Interface: Visual Input.** The VLM receives two images per decision step:

- 1) **Goal Image:** Target assembly configuration showing all bricks correctly inserted in the board (reference for completion)
- 2) **Current Image:** Current workspace state captured from a fixed “table_back” camera showing:
 - Board fixture with any already-inserted bricks
 - Remaining bricks on the table surface
 - Robot gripper (potentially holding a brick)

Both images are resized to 512×512 pixels and encoded as base64 PNG for transmission to the VLM API.

Action Output. The VLM outputs one of five action primitives:

- **pick up [color]:** Grasp a brick from the table or board
- **put down [color]:** Release held brick onto the table
- **reorient [color]:** Rotate held brick using a fixture surface to achieve correct insertion orientation
- **insert [color]:** Insert held brick into its board slot
- **done:** Declare task completion

The [color] argument must match one of the available brick colors (e.g., “red”, “blue”, “green”).

3) **Scoring Mechanism:** Each action returns an error code indicating success or failure:

- **err = 0:** Action executed successfully

- **err = -1:** Invalid precondition (e.g., “pick up” while already holding an object, “insert” without holding anything)
- **err = -2:** Object already in specified state (e.g., “pick up” an object already in hand)

No-Progress Detection. Beyond error codes, we detect semantic failures:

- **Insert no-progress:** Action succeeds (err=0) but gripper still holds the brick, indicating the slot is blocked
- **Pick no-progress:** Action succeeds but holding state unchanged

Episode Success. An episode succeeds when `env.is_success()` returns true, verified by:

- All bricks within 5mm of target position
- All bricks correctly oriented (quaternion error < 0.02)
- Special case: nail-type bricks must be upright (z-axis alignment > 0.9)

Auto-Completion. The system checks `is_success()` after every action. If the goal is achieved mid-episode (before the VLM outputs “done”), the episode automatically terminates as successful.

4) **Low-Level Executor:** Given a VLM action command, the low-level controller executes:

Pick Up.

- 1) Identify target brick by color name
- 2) Compute grasp pose based on brick geometry
- 3) Move gripper to approach position above brick
- 4) Lower gripper and close fingers to grasp
- 5) Lift brick to safe transport height

Insert.

- 1) Verify brick is in hand and correctly oriented
- 2) Align brick’s insertion site with board’s target hole
- 3) Lower brick along insertion trajectory
- 4) Release gripper once contact is detected
- 5) Return to home position

Reorient.

- 1) Place brick on fixture surface
- 2) Release and re-grasp from side orientation

- 3) Rotate to achieve upright configuration
- 4) Verify orientation before proceeding

All motions use differential IK with collision avoidance and velocity limits.

5) *Symbolic State Extraction*: For memory-based learning, we extract symbolic state features from each experience:

- **action_type**: pick, insert, reorient, or putdown
- **target_signature**: Shape description of target brick (e.g., “elongated rectangular block”)
- **holding_signature**: Shape of currently held brick (if any)
- **dependencies_satisfied**: Boolean indicating if all prerequisite bricks are inserted
- **target_blocked_by_colors**: List of brick colors currently blocking the target slot
- **remaining_pieces**: Colors of bricks not yet inserted
- **progress**: Fraction of bricks successfully inserted (0–1)

These features enable the consolidation engine to cluster similar experiences and generate hypotheses about blocking relationships without requiring explicit dependency graph access.

C. Example Learned Principles

Example principles discovered by `PhysMem` during simulation experiments:

- **SEQUENCE**: “When inserting elongated blocks, ensure all blocking pieces in the target slot are removed first”
- **AVOID**: “Don’t attempt to insert a brick when dependencies are not satisfied; the slot will be physically blocked”
- **PREFER**: “If an insert action fails repeatedly, put down the current brick and remove the blocking piece first”
- **AVOID**: “Don’t reorient the same brick more than twice. If orientation still seems wrong, the issue may be slot blocking”
- **SEQUENCE**: “For L-shaped bricks, insert the base portion before attempting to insert adjacent pieces”

D. Simulation Scaling Experiments

The simulation scaling experiments (VLM difficulty scaling and principle scaling) are presented in the main text (Section V-E and Section V-F). This section provides additional implementation details.

Evaluation Protocol. For each VLM and difficulty level combination, we run 100 independent episodes with fresh random seeds. Success is defined as correctly inserting all bricks within the maximum step limit. We report mean success rate with standard deviation computed across 3 random seeds for variance estimation.

Principle Counting. The principle count at each checkpoint is determined by the number of verified principles in long-term memory. We measure performance at powers of 2 (1, 2, 4, 8, 16, 32, 64, 128 principles) by saving memory snapshots during a 500-episode training run and evaluating each snapshot on a held-out test set of 50 episodes.

APPENDIX F

REAL-WORLD EXPERIMENT DETAILS

This appendix provides comprehensive details for our three real-world manipulation tasks. Each task is designed to test `PhysMem`’s ability to learn physics-grounded principles from interaction experience in settings where pre-trained VLM knowledge is insufficient.

A. Hardware Configuration

All experiments share a common hardware platform:

- **Robot Arm**: xArm6 6-DOF robotic arm
- **End Effector**: TPU-printed fin-ray effect soft grippers, providing compliant grasping for irregular objects
- **Camera System**:
 - **Top-down**: Intel RealSense D435, 1280×720 @ 15fps, mounted above workspace for global scene observation
 - **Wrist-mounted**: Intel RealSense D435, 1280×720 @ 15fps, for close-up manipulation views
 - **Base camera** (optional): Intel RealSense D435, 1280×720 @ 15fps, for side-angle views
 - **External cameras**: For documentation and qualitative analysis
- **Compute**: NVIDIA RTX 4090 for VLM inference through the API.

Model Configuration.

- **VLM Planner**: Gemini-3.0-Flash with thinking mode enabled, invoked at episode-level for high-level planning decisions
- **Reflection Model**: Qwen3-VL with thinking mode, running asynchronously every 15 seconds for hypothesis generation and consolidation

B. Parts Organization

The parts organization task requires the robot to efficiently pack irregularly-shaped 3D-printed components onto a discrete grid, testing spatial reasoning and learning optimal placement strategies through trial and error.

1) **Task Description: Setup.** Five distinct 3D-printed parts (labeled 001–005) are presented to the robot in fixed order. Each part occupies 2–4 adjacent grid cells in various configurations (L-shapes, U-shapes, I-shapes, T-shapes). The workspace contains a 3×10 grid (30 total cells) where parts must be placed sequentially.

Placement Constraints. To simplify the decision space, the selectable grid region is constrained to **adjacent areas** relative to already-occupied cells. Specifically, the VLM can only select grid cells within ± 3 columns from the current occupied boundary. This prevents the VLM from placing parts in isolated regions and encourages compact packing strategies. Note that parts may share the same grid cell indices when their 3D structures allow vertical overlap (e.g., one part’s overhang above another part’s base).

Objective. The robot must place all five parts onto the grid while minimizing the total number of occupied grid cells. This requires learning efficient packing strategies, including optimal rotation angles and placement positions that account for part geometries and remaining space.

Challenge. The task is challenging because: (1) optimal placement depends on the sequence of previously placed parts, (2) irregular part shapes create complex spatial constraints, (3) the VLM must reason about rotation effects on cell occupancy, (4) parts can structurally overlap in 3D while sharing grid indices, and (5) the constrained selection region requires planning within local neighborhoods.

2) **VLM Interface: Visual Input.** The VLM receives three images per decision step:

- 1) **Wrist Camera RGB:** Close-up view of the current part to be placed, showing detailed geometry and grasping orientation
- 2) **Enhanced Third-View Camera:** Top-down view of the grid workspace with visual overlays showing:
 - Grid cell boundaries and numbering (1–30)
 - Currently occupied cells (highlighted)
 - Selectable placement region (± 3 columns from occupied boundary)
- 3) **Part Template with Grid Watermark:** Reference image showing the current part type with a grid watermark overlay indicating which cells the part would occupy at different rotation angles. This helps the VLM understand the mapping between rotation and grid occupancy.

Action Output. The VLM outputs depend on the part type:

- **2-grid and 3-grid parts:** Output only the grid cell indices (e.g., `place(3, 13, 14)` for 3 cells)
- **4-grid parts:** Output grid cell indices **and** rotation angle (e.g., `place(6, 7, 16, 17, rotation=90)`)

The rotation angle is required for 4-grid parts because different rotations can result in the same grid occupancy pattern but different physical orientations.

3) **Part Naming and Spatial Coordinate System:** To enable precise spatial reasoning and hypothesis formation about part interactions, we introduce a standardized naming convention and a template-based coordinate system.

Part Naming Convention. Each part is uniquely identified by $\{\text{color}\}-\{\text{shape}\}$:

- **Colors:** black, white, red
- **Shapes:** L-shape (3 cells), q-shape (3 cells), I-shape (2 cells), U-shape (4 cells)

The six parts in this task are: red-L, white-q, red-q, white-U, black-U, black-I.

Template-Based Coordinate System. Since all parts occupy at most 4 cells, we represent each part’s local geometry using a 2×2 template grid with cells labeled $[a, b]$ (top row) and $[c, d]$ (bottom row):

```

+---+---+
| a | b |
+---+---+
| c | d |
+---+---+

```

Each shape’s default (0°) cell occupancy is defined as:

- **L-shape:** occupies $[a, c, d]$ – vertical bar on left, horizontal extension at bottom-right
- **q-shape:** occupies $[a, c, d]$ – similar to L but with different internal geometry
- **I-shape:** occupies $[a, c]$ – vertical bar on the left column
- **U-shape:** occupies $[a, b, c, d]$ – all four cells

Rotation Transformations. Counterclockwise rotation transforms the template coordinates as follows:

- **90° CCW:** $[[a, b], [c, d]]$

→

$[[b, d], [a, c]]$ (original $a \rightarrow c, b \rightarrow a, c \rightarrow d, d \rightarrow b$)

- **180° CCW:** $[[a, b], [c, d]]$

→

$[[d, c], [b, a]]$ (original $a \rightarrow d, b \rightarrow c, c \rightarrow b, d \rightarrow a$)

- **270° CCW:** $[[a, b], [c, d]]$

→

$[[c, a], [d, b]]$ (original $a \rightarrow b, b \rightarrow d, c \rightarrow a, d \rightarrow c$)

This allows consistent reference to part sub-regions regardless of rotation. For example, the “left vertical portion” of a U-shape is always $[a, c]$, whether the part is at 0° or rotated.

Part-Specific Offset Information. Even when occupying the same template cells, parts have internal geometric biases:

- **black-U / white-U:** At 0° , the physical mass is biased toward $[b, d]$; cells $[a, c]$ contain mostly empty space within the “U” opening.
- **white-q:** At 0° , the left edge aligns with the left boundary of $[a, c]$.
- **red-q:** At 0° , the right edge aligns with the right boundary of $[a, c]$.
- **black-I:** At 0° , the top and right edges align with the top-right of $[a, c]$.

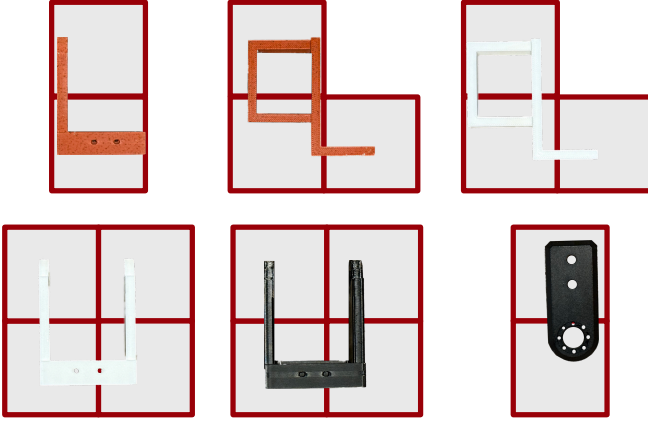


Fig. 13: **Parts Organization: Component shapes and grid occupancy.** Each panel shows a 3D-printed part overlaid on its 2×2 template grid, illustrating which cells ($[a, b, c, d]$) each part occupies. **Top row:** red-L (2 cells, occupies $[a, c]$), red-q (3 cells, $[a, c, d]$), white-q with red-L interlocking example. **Bottom row:** white-U (4 cells, $[a, b, c, d]$), black-U (4 cells), black-I (2 cells, $[a, c]$). Red grid lines mark template boundaries. Note the internal geometric biases: U-shapes have empty space in $[a, c]$ (the “U” opening), while q-shapes have opposite edge alignments (white-q left-aligned, red-q right-aligned).

These offsets determine whether adjacent parts can physically interlock or overlap when sharing grid cells.

Using Coordinates for Spatial Reasoning. This coordinate system enables precise hypothesis formation about part interactions:

- “PREFER placing the $[c]$ region of white-q toward the right side”
- “AVOID complete overlap between the $[a, c]$ regions of two q-shaped parts”
- “When white-U is rotated 180° , its $[a, c]$ region can interlock with another U-shape’s $[b, d]$ region”

Figure 13 visualizes all six components with their grid cell occupancy patterns, illustrating the template-based coordinate system in practice.

4) **Scoring Mechanism: Step-Level Feedback.** Each placement action receives immediate feedback:

- **+1:** Successful placement in a safe configuration
- **-1:** Collision occurs during placement, but the plan was theoretically valid (indicates execution error or unforeseen physical interaction)
- **-2:** Invalid or unreasonable plan (e.g., out-of-bounds grid number, selecting non-adjacent cells, impossible rotation angle)

Error Limits. To prevent excessive failures and ensure episode progression:

- **Per-part limit:** Maximum 3 failed attempts per part. After 3 failures, the system skips to the next part.
- **Cumulative limit:** Maximum 5 total errors per episode. Exceeding this terminates the episode early.

Episode-Level Score. Final episode success is measured by the total grid cell occupancy after all five parts are placed. The normalized score is computed as:

$$\text{Score} = 5 - 5 \times \text{clip} \left(\frac{s - s_{\min}}{s_{\max} - s_{\min}}, 0, 1 \right) \quad (9)$$

where s is the actual grid occupancy, $s_{\min}$ is the theoretical minimum occupancy (optimal packing), and $s_{\max}$ is the maximum possible occupancy. A score of 5 indicates optimal packing, while 0 indicates worst-case packing.

5) **Safety Mechanism:** Before executing each placement, the system performs collision checking using Segment Anything Model (SAM) [32]:

- 1) Generate segmentation mask for the part to be placed
- 2) Generate masks for all currently placed parts
- 3) Compute mask overlap between proposed placement and existing parts
- 4) Reject placement if overlap exceeds threshold; return collision penalty

This SAM-based collision detection prevents physical damage while providing informative negative feedback for learning.

6) **Low-Level Executor:** Given a valid VLM plan (grid cell indices and optional rotation), the low-level controller executes a heuristic pick-and-place sequence:

- 1) **Grasp Pose:** Use a fixed pre-defined grasp pose for each part type, determined through offline calibration. This ensures consistent grasping regardless of VLM decisions.
- 2) **Coordinate Transformation:** Convert target grid cell indices to 3D world coordinates using camera-to-robot calibration matrices.
- 3) **Placement Execution:** Execute pick-and-place motion with the gripper oriented according to the specified rotation angle.
- 4) **Release and Retract:** Open gripper and retract to a safe observation pose.

7) **Evidence Recording:** For memory formation and learning, the system records keyframe evidence at critical moments:

- **Step Start:** Capture visual state before any action is taken (current grid occupancy, part to be placed)
- **Post-Stabilization:** Capture visual state after the robot retracts and the scene stabilizes (placement outcome, new occupancy)

These keyframes, along with the VLM’s action and resulting score, form the episodic memory entries used for principle extraction.

8) *Memory Content Examples:* We provide concrete examples of how the coordinate system enables structured hypothesis formation and principle extraction.

Episodic Memory (Experience Records). Each experience captures the context, action, outcome, and spatial reasoning:

```
Experience #47:
- Part: white-U (4 cells, [a,b,c,d])
- Action: place(5, 6, 15, 16, rotation=180)
- Outcome: SUCCESS (+1)
- Context: black-U already at cells (3,4,13,14)
  with rotation=0
- Observation: white-U's [a,c] region (now
  bottom-left after 180 deg)
  interlocked with black-U's [b,d] region
  without collision
```

```
Experience #52:
- Part: red-q (3 cells, [a,c,d])
- Action: place(7, 17, 18)
- Outcome: COLLISION (-1)
- Context: white-q at cells (6, 16, 17) with
  rotation=0
- Observation: red-q's [a,c] region overlapped
  with white-q's [a,c]
  region; internal geometries conflicted
  despite same template cells
```

Working Memory (Hypotheses Under Testing). Hypotheses are generated from clustered experiences, using template coordinates:

```
## Hypotheses (Under Testing)

1. [TESTING] (from 3 experiences) When placing
  U-shaped parts:
  - At 180 deg rotation, the [a,c] region
    becomes physically
    accessible for interlocking with another U
    's [b,d] region
  - Confidence: 67% (2/3 successes with this
    pattern)

2. [TESTING] (from 4 experiences) For q-shaped
  part pairs:
  - AVOID having both parts' [a,c] regions
    overlap on same cells
  - The [d] region of one q-shape CAN overlap
    with [c] of another
  - Confidence: 75% (3/4 successes following
    this rule)

3. [TESTING] (from 2 experiences) Black-I
  placement strategy:
  - The internal bias (top-right of [a,c])
    allows black-I to fit
    into the empty [a,c] space of U-shaped
    parts
  - Requires U-shape at 0 deg (opening faces
    left)
```

Long-Term Memory (Verified Principles). Principles are promoted after sufficient verification (confidence ≥ 0.8):

```
## Verified Principles
```

1. [92%] [PREFER] U-shape interlocking strategy
:
When two U-shapes must share adjacent cells,
place the second
at 180 deg rotation so its [a,c] region
interlocks with the
first U's [b,d] region. The complementary
internal biases
(physical mass in [b,d], empty space in [a,c
) enable overlap.
2. [88%] [AVOID] Q-shape [a,c] conflict:
Never place two q-shaped parts with their [a
,c] regions on
overlapping grid cells. Despite occupying
the same template
positions, their internal geometries (left-
aligned vs right-
aligned) create physical collisions.
3. [85%] [PREFER] Q-shape complementary
placement:
Place white-q's [c] region adjacent to (or
sharing cells with)
red-q's [d] region. Their opposite internal
biases create
physical clearance for successful overlap.
4. [82%] [SEQUENCE] Size-constrained ordering:
Place U-shaped parts (4 cells) before q-
shaped parts (3 cells).
U-shapes have less placement flexibility and
benefit from
early positioning when more grid space is
available.

Complete Specifications. The full coordinate system reference, spatial reasoning guidelines, and decision rules incorporated into the VLM prompt for Parts Organization are elided in this release.

C. Ball Navigation

The ball navigation task requires the robot to push a common soccer ball toy through an obstacle course to reach a target location, testing the VLM’s ability to adapt to complex ball rolling dynamics and select appropriate push strategies.

1) **Task Description: Setup.** A soccer ball toy is placed on a tabletop workspace containing multiple obstacles. The ball’s starting position is randomly initialized within a fixed area and manually placed. To facilitate smooth rolling interaction, the gripper holds a cylindrical foam piece that contacts the ball during pushing.

Task Stages. The ball must pass through at least three stages to reach the target:

- 1) **Stage 1:** Pass through the first obstacle’s “bridge hole”
 - 2) **Stage 2:** Navigate around all obstacles and approach the target area
 - 3) **Stage 3:** Fine-tune position and reach the final target
- The goal is to complete all stages within a maximum of 6 push actions.

Ball Dynamics. According to momentum principles and empirical testing, the ball’s landing position after each push depends on: (1) gripper movement direction, (2) gripper stopping point, (3) gripper movement speed, (4) the ball’s intrinsic dynamics properties, and (5) environmental factors.

Evidence Recording. For each interaction step, we record keyframes from the top-down camera at two time points: (1) the start of the step, and (2) after a fixed duration when the ball has stabilized. These keyframes serve as **evidence** for the experience stored in memory.

Challenge. The task is challenging because: (1) ball rolling dynamics are difficult to predict precisely, (2) obstacle configurations require multi-step indirect pushing paths, (3) different speed levels produce significantly different ball behaviors, and (4) the VLM must learn ball-specific strategies through trial and error.

2) **VLM Interface: Visual Input.** The VLM receives two images per decision step:

- 1) **Wrist Camera:** Close-up view of the ball and nearby obstacles, providing detailed local context for push planning
- 2) **Top-Down Camera:** Bird’s-eye view of the entire workspace showing:
 - Current ball position
 - All obstacle locations and shapes
 - Target region location
 - Trajectory history from previous pushes (if any)

Action Output. The VLM outputs a push action combining visual grounding with text specification:

- **start_direction:** Direction from ball center to gripper start position, from 8 options: left, right, top, bottom, top-left, top-right, bottom-left, bottom-right
- **end_direction:** Direction from ball center to gripper end position (same 8 options)
- **start_y, start_x:** Pixel coordinates where gripper starts the push (note: y before x following Gemini visual grounding convention)

- **end_y, end_x:** Pixel coordinates defining push endpoint
- **speed:** Push velocity level $\in \{\text{low}, \text{medium}, \text{high}\}$

Coordinate System. Pixel values are scaled to a normalized 0–1000 range regardless of original image dimensions. The top-left corner is (0,0) and the bottom-right is (1000,1000). These normalized coordinates are converted to absolute pixel positions based on the camera frame’s actual width and height.

Direction Specification. The **start_direction** and **end_direction** fields explicitly encode the positional relationship between the gripper points and the ball’s current position. This forces the VLM to reason about push geometry and enhances its understanding of the push strategy, although these direction labels are not directly passed to the low-level executor.

Endpoint Constraint. To ensure physically reasonable push distances, the **end point must lie within a circle of radius 150mm centered at the start point**. The low-level executor clips any endpoint that exceeds this constraint.

Example: `push("left", "right", 300, 150, 300, 350, "medium")` specifies a push where the gripper starts on the left side of the ball and pushes toward the right at medium speed.

3) **Scoring Mechanism:** The scoring system provides dense feedback for learning:

- +1: Gripper successfully affects ball state (ball moves)
- −2: Gripper collides with any obstacle during interaction
- −2: VLM outputs invalid start/end point (in unsafe zone)
- +3: Ball successfully passes through obstacle 1’s bridge hole
- +5: Ball reaches the final target at step i ($i \leq 6$)
- $+2 \times (6 - i)$: Early completion bonus for reaching target at step i

Episode Termination. The episode terminates when: (1) the ball reaches the target (success), (2) the VLM outputs unsafe points more than 3 times consecutively, (3) 6 steps are exhausted without reaching target (failure), or (4) the ball exits the workspace boundaries (failure). Upon termination, the ball is manually reset to a random starting position.

4) **Safety Mechanism:** To prevent gripper-obstacle collisions during push execution:

- Segment all obstacles on the tabletop using the top-down camera
- Generate obstacle masks via SAM segmentation
- Apply morphological dilation to expand obstacle regions by a safety margin (similar to costmap inflation in ROS navigation stack)
- Create an “inflated obstacle map” representing unsafe zones
- Check if the planned start point or end point falls within unsafe zones
- Reject plans where either point enters the inflated obstacle region

This inflation-based approach provides conservative collision avoidance while allowing the ball itself to pass close to obstacles. If the VLM outputs unsafe points more than 3 consecutive times, the episode is terminated.

5) *Low-Level Executor*: Given a valid VLM push specification, the low-level controller implements a heuristic push policy:

- Convert normalized pixel coordinates (0–1000) to absolute pixels based on camera frame dimensions
- Convert 2D pixel coordinates to 3D world coordinates using camera calibration
- Move gripper to the start point position
- Rotate gripper to align with the push direction (start → end vector)
- Execute constant-velocity linear motion toward the end point:
 - Low: 150 mm/s
 - Medium: 300 mm/s
 - High: 450 mm/s
- **Vertical offset**: The end point is offset slightly upward along the z-axis compared to the start point, simulating the human behavior of pushing while gradually lifting. This facilitates ball rolling and adds complexity to the motion dynamics.
- **Endpoint clipping**: Final end point is clipped to ensure it remains within the 150mm radius circle centered at the start point
- Retract gripper after push completion

6) *Obstacle Layout and Spatial Zones*: The workspace contains three fixed obstacles and a target region:

Obstacle Configuration.

- **Obstacle 1 (Blue)**: Archway block on the LEFT side of the workspace. Has a “bridge hole” at approximately $y \in [400, 480]$ that the ball *must* pass through. The ball cannot go over or around this obstacle.
- **Obstacle 2 (Red)**: Rectangular cuboid in the MIDDLE of the workspace. Acts as a solid blocker that the ball must navigate around.
- **Obstacle 3 (Purple)**: Rectangular cuboid at the BOTTOM of the workspace. **CRITICAL**: If the ball lands on top of this obstacle ($y \approx 600\text{--}700$), the gripper cannot access the ball from below, resulting in a stuck state.
- **Target**: Circular region with cross pattern on the RIGHT side.

Spatial Zones. For reasoning about ball position and strategy selection, we define five zones:

- **Zone A (Initial)**: $y < 400, x > 300$ — Above archway level
- **Zone B (Pre-archway)**: $y \in [400, 480], x < 350$ — Aligned with archway
- **Zone C (Post-archway)**: $x > 350, y < 600$ — Transit area after archway
- **Zone D (Target approach)**: $x > 650, y < 550$ — Near target
- **Danger Zone**: $y > 550, x > 520$ — Robot arm motion limit exceeded

7) *Memory Content Examples*: We provide concrete examples of how spatial zones and obstacle references enable structured hypothesis formation and principle extraction.

Episodic Memory (Experience Records). Each experience captures ball position, zone, distances to obstacles, and outcome:

```
Experience #23:
- Step: 2/6, Ball Zone: ZONE B (pre-archway,
  aligned with archway)
- Ball Position: (y=440, x=280)
- Distance to OBS1 archway: ~50 pixels
- Distance to OBS3: ~280 pixels
- Action: push("left", "right", 440, 200, 440,
  450, "medium")
- Outcome: SUCCESS (+3, passed through archway)
- Ball Position (after): (y=430, x=480)
- Observation: MEDIUM speed successfully pushed
  ball through archway.
  Horizontal push (constant y) maintained
  archway height alignment.

Experience #15:
- Step: 1/6, Ball Zone: ZONE A (initial, close
  to OBS1)
- Ball Position: (y=350, x=420)
- Distance to OBS1 archway: ~150 pixels
- Action: push("top-right", "bottom-left", 280,
  480, 450, 300, "high")
- Outcome: OVERSHOOT (-1)
- Ball Position (after): (y=680, x=250)
- Observation: HIGH speed too strong from close
  distance. Ball passed
  through archway but landed ON TOP of OBS3 (y
  ~680). Now stuck.
```

Working Memory (Hypotheses Under Testing). Hypotheses reference obstacles and zones explicitly:

```
## Hypotheses (Under Testing)

1. [TESTING] Post-archway LOW speed requirement
  (from 3 experiences):
  After passing through OBS1, MUST use LOW
  speed for next push.
  MEDIUM/HIGH causes ball to roll onto OBS3
  top surface (y~600-700).
  Confidence: 67% (2/3)

2. [TESTING] 45-degree descent for archway
  alignment (from 4 experiences):
  When in ZONE A (above archway level),
  diagonal descent
  (end_dir="bottom-left") is more controllable
  than straight "bottom".
  Confidence: 75% (3/4)
```

Long-Term Memory (Verified Principles). Principles are promoted after sufficient verification:

```
## Verified Principles

1. [95%] [SPEED] Distance-based speed for OBS1
  approach:
  Ball >200px from archway: HIGH speed
  acceptable
  Ball 100-200px: MEDIUM preferred; Ball <100
  px: LOW REQUIRED

2. [88%] [AVOID] OBS3 danger zone (y>550, x
  >520):
  Never push toward target from DANGER zone.
  Robot arm exceeds
  motion limit. MUST push upward (end_dir
  includes "top") first.
```

Complete Specifications. The full obstacle layout, spatial zone definitions, and stage-specific decision rules incorporated into the VLM prompt for Ball Navigation are elided in this release.

D. Balanced Stacking

The balanced stacking task requires the robot to build a stable tower from irregularly-shaped “balance stones” with varying physical properties, testing physics reasoning about stability, friction, and center of gravity.

1) *Task Description: Setup.* Five balance stones with diverse properties are scattered on the workspace:

- **Size variation:** Small, medium, and large stones
- **Shape variation:** Flat, rounded, angular geometries
- **Surface properties:** Different friction coefficients (smooth, textured, rough)
- **Weight distribution:** Varying centers of gravity

Objective. Stack all five stones into a stable tower. The tower must remain standing (without collapse) for a stability verification period after the final stone is placed.

Challenge. The task is challenging because: (1) optimal stacking order depends on stone properties that are not visually obvious, (2) stone shapes create complex contact geometries affecting stability, (3) friction between stone pairs varies and affects stackability, and (4) the VLM must learn which stone combinations are stable through trial and error.

2) *VLM Interface: Visual Input.* The VLM receives three images per decision step:

- 1) **Wrist Camera:** Close-up view of the stone being grasped or the current tower top surface, showing detailed texture and contact geometry
- 2) **Top-Down Camera:** Overhead view showing all available stones and the current tower state
- 3) **Base Camera:** Side-angle view of the stacking area providing depth perception for tower height and stone orientations

Action Output. The VLM outputs a visual grounding specification:

- `stack(point_y, point_x)`: Pixel coordinates identifying which stone to pick up next (note: y before x following Gemini convention)

Example: `stack(200, 150)` selects the stone at pixel position $y = 200$, $x = 150$. The system uses this point to identify the target stone via SAM segmentation, then autonomously determines grasp pose and placement location using adaptive height control.

3) *Scoring Mechanism:* The scoring system rewards successful stacking while penalizing collapses:

- $+i$: Successfully placing a stone on layer i (where $i = 1$ is the base, $i = 2$ is second layer, etc.)
- $-2j$: Penalty when j stones fall during or after placement

This progressive reward structure encourages building taller stable towers: placing the 5th stone on a 4-layer tower yields $+5$, but causing 3 stones to fall incurs -6 . The optimal strategy balances ambition with stability.

Episode success is defined as completing a 5-stone tower that remains stable.

4) *Safety Mechanism:* Tower stability is monitored continuously:

- 1) Visual tracking of stone positions during and after placement

- 2) Automatic episode termination if catastrophic collapse detected

- 3) Force/torque sensing during placement to detect instability
- 5) *Low-Level Executor:* Given the VLM’s target stone selection, the low-level controller performs:

Grasp Planning.

- Apply SAM segmentation to isolate the selected stone
- Analyze stone geometry to determine optimal grasp direction
- Compute antipodal grasp points based on stone shape
- Plan collision-free approach trajectory

Adaptive Placement.

- Move stone above current tower with clearance margin
- **Adaptive height adjustment:** Incrementally lower stone while monitoring force feedback
- Detect contact with tower surface via force threshold
- Release gripper and retract with minimal disturbance
- Verify tower stability before proceeding

The adaptive height adjustment is critical for handling the irregular stone geometries and varying tower heights.

6) Stone Naming System and Surface Properties:

To enable precise hypothesis formation about stone interactions, we use a systematic naming convention: `{Surface}_{Shape}_{Size}_{Orientation}`.

Surface Types (Friction Ranking). Five surface materials with different friction coefficients:

- 1) **3M** (3M Gripping Material TB641): Black velcro-like, *highest* friction
- 2) **BLK** (Black Duct Tape): Industrial grade, high friction
- 3) **WHT** (White PVC Tape): Insulation tape, medium friction
- 4) **WOOD** (Rough Wooden): Natural wood grain, medium-low friction
- 5) **PAINT** (Painted Surface): Smooth lacquer, *lowest* friction

Shape Types (Cross-Section Geometry).

- 1) **SQR** (Square): Flat 4-sided, excellent contact area
- 2) **HEX** (Hexagonal): 6-sided, smaller contact, best for top layers
- 3) **DMND** (Diamond): Can be horizontal (H) or vertical (V) bar
- 4) **PENT** (Pentagon): 5-sided irregular
- 5) **OVAL** (Egg): Rounded, challenging contact
- 6) **TREE** (Branch): Irregular, must always be placed last

Orientation Types.

- **H** (Horizontal): Long bar lying flat, width $\gg$ height
- **V** (Vertical): Long bar standing up, height $\gg$ width
- **C** (Compact): Roughly cubic, width $\approx$ height

Example: `3M_HEX_L_C` = 3M surface + Hexagonal cross-section + Large + Compact

7) *Memory Content Examples:* We provide concrete examples of how surface and shape properties enable structured hypothesis formation and principle extraction.

Episodic Memory (Experience Records). Each experience captures stone properties, tower state, and placement outcome:

```
Experience #18:
- Step: 1/5, Tower State: Empty (first stone)
- Stone Selected: BLK_DMND_M_V (Black Diamond
  Medium Vertical)
- Contact Cross-Section: Diamond point, ~3cm2
- Aspect Ratio: 1:3.5 (vertical bar)
- Surface Friction: HIGH (Black Tape)
- Action: stack(280, 380)
- Outcome: COLLAPSE (-2), Stone fell
  immediately
- Observation: Vertical bar (V orientation) as
  base stone FAILS.
  Small contact area insufficient. AVOID V-
  orientation at Step 1.

Experience #31:
- Step: 3/5, Tower State: 2 layers
- Current Top: 3M_SQR_M_C (3M Square)
- Stone Selected: WOOD_SQR_M_C (Wooden Square)
- Contact Compatibility: SQR -> SQR (excellent)
- Friction Pairing: 3M -> WOOD (good)
- Action: stack(400, 380)
- Outcome: SUCCESS (+3)
- Observation: SQR on SQR with 3M->WOOD
  friction is stable.
```

Working Memory (Hypotheses Under Testing). Hypotheses reference surface and shape properties:

```
## Hypotheses (Under Testing)

1. [TESTING] Tree-shaped (TREE) stones must be
  LAST (Step 5):
  Irregular surface cannot support any stone
  above.
  Confidence: 100% (3/3)

2. [TESTING] SQR on DMND_H incompatibility:
  Square cross-section cannot stack on
  horizontal bar.
  Contact geometries do not align. 100%
  collapse rate.
  Confidence: 100% (3/3)
```

Long-Term Memory (Verified Principles). Principles use the naming convention for precise specification:

```
## Verified Principles

1. [95%] [REQUIRE] Step 1 base stone criteria:
  Select LARGEST stone + HIGHEST friction (3M>
  BLK>WHT>WOOD>PAINT)
  + LARGEST contact area. NEVER select V-
  orientation.

2. [90%] [SEQUENCE] HEX stones at Steps 4-5:
  Hexagonal contact too small for lower layers
  .
  Place after Step 3 (toward top of tower).
```

Complete Specifications. The full stone naming convention, friction compatibility matrix, and step-by-step selection rules incorporated into the VLM prompt for Balanced Stacking are elided in this release.

E. Example Learned Principles

This section presents example principles learned by PhysMem during real-world experiments. These demonstrate the system’s ability to discover task-specific physics knowledge through interaction.

Parts Organization.

- PREFER: “Place elongated parts along grid edges to maximize interior space for irregular shapes”
- AVOID: “Don’t place large multi-cell parts in the center first; they block flexible arrangements”
- SEQUENCE: “Place the most constrained (largest) parts first, then fill gaps with smaller parts”
- AVOID: “Avoid complete overlap between the [a,c] regions of two q-shaped parts; their internal geometries conflict”
- PREFER: “When placing white-U at 180 degrees, its [a,c] region can interlock with another U-shape’s [b,d] region”
- SEQUENCE: “Place I-shaped parts last; their narrow profile fits into remaining gaps more easily”

Ball Navigation.

- AVOID: “High speed pushes near obstacles; ball rebounds unpredictably”
- PREFER: “Use medium speed for long-distance pushes toward open areas”
- PREFER: “Aim for indirect paths that use obstacle edges to guide ball”
- AVOID: “Don’t push directly toward the bridge hole from far away; small angular errors cause the ball to miss”
- SEQUENCE: “First navigate past the archway, then reduce speed for the final approach to target”
- PREFER: “Use low speed (0.3–0.5) after passing through narrow gaps to maintain control”
- AVOID: “Avoid pushing at angles greater than 45 degrees; the ball tends to spin rather than roll straight”

Balanced Stacking.

- SEQUENCE: “Place the flattest, largest stone as base for maximum contact area”
- AVOID: “Don’t stack smooth-surfaced stones directly on each other; they slide easily”
- PREFER: “Alternate between rough and smooth stones for better grip”
- AVOID: “Don’t place heavy stones on small contact surfaces; high pressure causes instability”
- PREFER: “Select stones with natural concavities as middle layers; they cradle the stones above”
- AVOID: “Don’t place the largest stone on top; the high center of mass causes toppling”
- SEQUENCE: “Build towers with decreasing stone size from bottom to top for optimal stability”

F. Out-of-Distribution Experimental Settings

To evaluate memory transfer capabilities (Section V-D in main text), we designed out-of-distribution (OOD) variants for each task. Figure 14 shows the OOD settings and experimental props.

Parts Organization OOD Variant. We modify the initial grid configuration:

- Parts are pre-placed at different positions (occupying cells 14–16, 23–25)
- Initial component count is increased, reducing available placement space
- The constrained placement region (± 3 columns from occupied boundary) creates more challenging packing scenarios

These modifications test whether learned spatial reasoning principles transfer to novel initial configurations.

Ball Navigation OOD Variant. We introduce two new ball types with substantially different dynamics:

- TENNIS BALL: Higher friction, less predictable bounce behavior
- MOON-TEXTURED PLASTIC BALL: Lower friction, different rolling dynamics

Prior principles about soccer ball dynamics (speed selection, push angle effects) may not transfer directly, requiring test-time adaptation.

Balanced Stacking OOD Variant. We test with five unseen stone combinations:

- Novel weight distributions not encountered during training
- Surface textures created with different tape combinations
- Shape/material/size combinations that never appeared before

To ensure the fairness of the experiment, we used the same sequence of part combinations during the out-of-distribution (OOD) testing in different episodes. Stability principles (e.g., “flat base first”) may transfer, but specific friction pairing rules require re-learning.

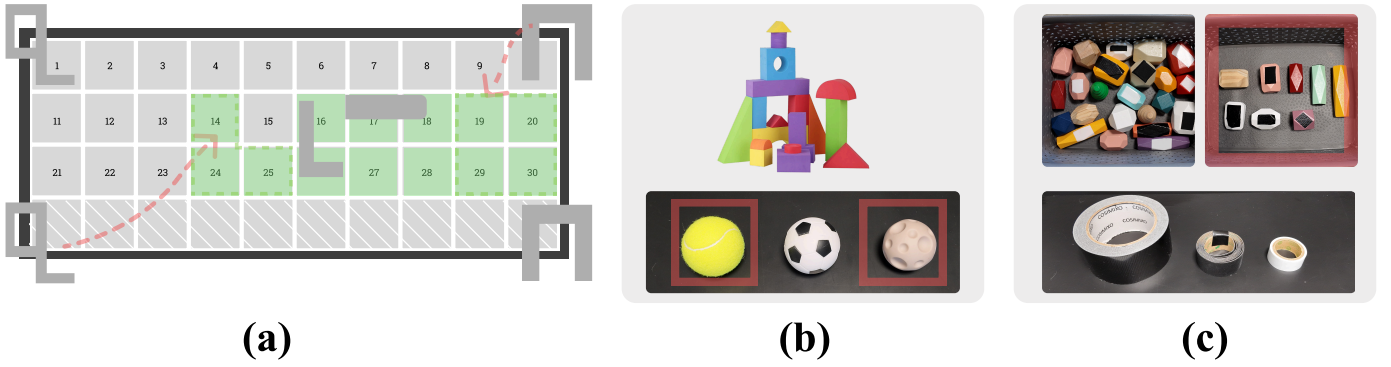


Fig. 14: **Out-of-distribution experimental settings and props.** (a) **Parts Organization OOD:** Initial grid configuration is modified with parts pre-placed at different positions. Green cells (14–16, 23–25) are pre-occupied; hatched area indicates the constrained placement region (± 3 columns from occupied boundary). Dashed trajectory shows example placement paths. (b) **Ball Navigation OOD:** Top shows stacking blocks (illustrative only, not used in ball task). Bottom shows all balls: soccer ball (center) is the in-distribution training ball; tennis ball (yellow) and moon-textured plastic ball (red boxes) are OOD items with different friction and elasticity. (c) **Balanced Stacking OOD:** Top rows show balance stones with varying surfaces—some treated with tapes (3M, black duct, white PVC; shown in bottom), others with natural WOOD or PAINT surfaces. Red-boxed stones (top-right) represent OOD combinations with novel shape/material/size configurations not seen during in-distribution training.

APPENDIX G
CROSS-MODEL GENERALIZATION DETAILS

We tested `PhysMem` across multiple VLM architectures to validate model-agnostic benefits.

TABLE VIII: **Cross-model results on medium difficulty.** Success rates with and without `PhysMem` across different VLM planners (100 episodes each). All models use thinking mode. Consolidation model fixed to Qwen3-VL.

Model	Baseline	+PhysMem	Δ
Gemini-3-Flash	53%	76%	+23%
GPT-5.1	43%	57%	+14%
Qwen3-VL-235B	38%	50%	+12%
Gemini-ER-1.5	29%	34%	+5%

Analysis. Test-time learning benefits scale with VLM capability. Gemini-3-Flash achieves the largest improvement (+23%), while weaker models show diminishing returns. This suggests that effective hypothesis generation and verification require sufficient base reasoning capability. The consolidation model (Qwen3-VL) remains fixed across all conditions, isolating the effect of planner capability on learning efficiency.

APPENDIX H

DETAILED TASK SPECIFICATIONS FOR VLM PLANNING

This section originally contained the complete task-specific reference materials directly incorporated into VLM planner prompts (coordinate systems, naming conventions, spatial reasoning frameworks, and decision rules) for Parts Organization, Ball Navigation, and Balanced Stacking. We have elided these task-specific specifications in this release. The general structure of how such specifications integrate with the memory system is described in the eight-section prompt outline below (Section I); examples of the principles and hypotheses that emerge for each task are reported in Section F-B8, Section F-C7, and Section F-D7.

APPENDIX I

PROMPT TEMPLATES: STRUCTURE AND GENERAL COMPONENTS

This section describes the structure of the prompt templates used in our real-world experiments and shows the task-agnostic components (hypothesis generation and memory injection). Each task prompt follows an **eight-section structure** that integrates the memory system:

- 1) **Task Description:** Core task specification and naming conventions
- 2) **Action Definitions:** Available actions and their semantics
- 3) **Strategic Guidance:** Planning heuristics and decision rules
- 4) **Current State:** Robot’s current situation
- 5) **Repeated Action Warning:** Loop detection (if applicable)
- 6) **Long-term Memory:** Verified principles (Tier 3)
- 7) **Working Memory:** Hypotheses under test (Tier 2)
- 8) **Action Request:** Output format specification

Sections 6–7 are dynamically populated by the memory system at run time. The complete per-task prompt instances for Brick Insertion, Parts Organization, Ball Navigation, and Balanced Stacking are elided in this release; we instead show the task-agnostic hypothesis generation template and the memory injection format used across all tasks.

A. Hypothesis Generation Prompt Template

The consolidation engine uses the following **general template** to generate hypotheses from clustered experiences. This template is task-agnostic and can be instantiated for any manipulation task by providing task-specific descriptions and examples.

General template for hypothesis generation (task-agnostic).

```
=====
HYPOTHESIS GENERATION FROM EXPERIENCE CLUSTERS
=====

You are analyzing robot manipulation experiences to identify
patterns and generate actionable hypotheses for improving
task performance.

=====
TASK CONTEXT
=====

Task Name: {task_name}
Task Description: {task_description}

Available Actions: {available_actions}
(e.g., "pick, insert, reorient, put down" for assembly;
  "push" for ball navigation;
  "stack" for balanced stacking;
  "place" for parts organization)

Key Object Properties: {key_properties}
(e.g., "brick color, shape, orientation" for assembly;
  "ball position, obstacle locations, speed" for navigation;
  "stone size, texture, weight distribution" for stacking;
  "part shape, cell occupancy, rotation" for organization)

=====
CLUSTER SUMMARY
=====

This cluster contains {n_experiences} similar experiences:

Action Type Distribution:
{action_type_distribution}

Outcome Statistics:
- Success rate: {success_rate}%
- Total successes: {n_successes}
- Total failures: {n_failures}
- Common failure reasons: {failure_tags}

Shared Context/Properties:
{shared_properties}

=====
SAMPLE EXPERIENCES FROM CLUSTER
=====

{formatted_experiences}

[Note: The system automatically formats 3-5 representative
experiences from the cluster, showing action, outcome, and
relevant context for each.]

=====
EXISTING KNOWLEDGE (avoid duplicating)
=====

Current Verified Principles (Long-term Memory):
{existing_principles}

Current Hypotheses Under Testing (Working Memory):
{existing_hypotheses}
```


IMPORTANT: Do NOT generate hypotheses that duplicate or closely overlap with existing knowledge.

=====

YOUR TASK: GENERATE HYPOTHESES

=====

Analyze the patterns in this experience cluster and generate 1-3 NEW hypotheses that could explain the observed outcomes.

HYPOTHESIS TYPES:

- AVOID: Identifies actions/conditions leading to failure
"Don't do X when condition Y is true"
- PREFER: Identifies actions/strategies leading to success
"Do X when condition Y is true"
- SEQUENCE: Identifies order-dependent rules
"Do X before Y" or "After X, do Y"
- COMPARE: Identifies comparative preferences
"X works better than Y when condition Z"
- GENERAL: General observations about the task domain

REQUIREMENTS FOR EACH HYPOTHESIS:

1. Be SPECIFIC - include concrete conditions and actions
2. Be ACTIONABLE - the agent can directly apply the rule
3. Be GROUNDED - based on patterns in the experience cluster
4. Reference relevant properties (objects, conditions, states)

=====

OUTPUT FORMAT

=====

For each hypothesis, provide:

HYPOTHESIS {N}:

Type: {AVOID | PREFER | SEQUENCE | COMPARE | GENERAL}
Statement: [Clear, actionable rule in natural language]
Applicable_Actions: [List of action types this applies to]
Trigger_Conditions: [When/where this hypothesis applies]
Evidence: [Brief explanation of supporting experiences]

=====

EXAMPLE OUTPUTS BY TASK TYPE

=====

[Parts Organization Task]

HYPOTHESIS 1:

Type: PREFER
Statement: Place L-shaped parts in corner positions with the corner facing inward to maximize space utilization.
Applicable_Actions: [place]
Trigger_Conditions: When placing L-shaped parts and corner cells (1, 10, 21, 30) are available.
Evidence: 4/5 successful placements used corner positions.

[Ball Navigation Task]

HYPOTHESIS 1:

Type: AVOID
Statement: Do not use high speed when the ball is within 50 pixels of any obstacle.
Applicable_Actions: [push]
Trigger_Conditions: When ball-to-obstacle distance < 50 pixels.
Evidence: 6/7 failures with high speed occurred near obstacles.

[Balanced Stacking Task]

HYPOTHESIS 1:

Type: SEQUENCE
Statement: Place the largest, flattest stone as the base before stacking any other stones.
Applicable_Actions: [stack]
Trigger_Conditions: When tower is empty (first placement).
Evidence: All 5 successful towers started with large flat base.

[Brick Assembly Task]

HYPOTHESIS 1:

Type: AVOID

Statement: Do not attempt insert when adjacent slots are occupied by blocking bricks.

Applicable_Actions: [insert]

Trigger_Conditions: When dependencies are not satisfied.

Evidence: 8/8 blocked inserts had adjacent occupied slots.

B. Memory Injection Formatting

Principle Formatting (Long-term Memory). Principles are formatted with confidence levels using a three-tier system:

```
\# Learned Principles (Apply these!)
```

1. [95%] [SEQUENCE] Always place the largest stone as the base.
2. [88%] [AVOID] High-speed pushes near obstacles cause rebounds.
3. [82%] [PREFER] Place L-shaped parts in corners facing inward.

Confidence display thresholds:

- **HIGH** ($\geq 85\%$): Strong evidence, should be followed
- **MEDIUM** (60%–84%): Moderate evidence, consider carefully
- **LOW** ($< 60\%$): Weak evidence, use with caution

Hypothesis Formatting (Working Memory). Hypotheses are marked as under testing to indicate lower confidence:

```
\# Hypotheses (Consider but verify)
```

1. [TESTING] Rough stones grip smooth stones better than vice versa.
2. [TESTING] Medium speed is optimal for most ball navigation.