LEARNING PHYSICAL PRINCIPLES FROM INTERACTION: SELF-EVOLVING EMBODIED PLANNING VIA TEST-TIME MEMORY

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

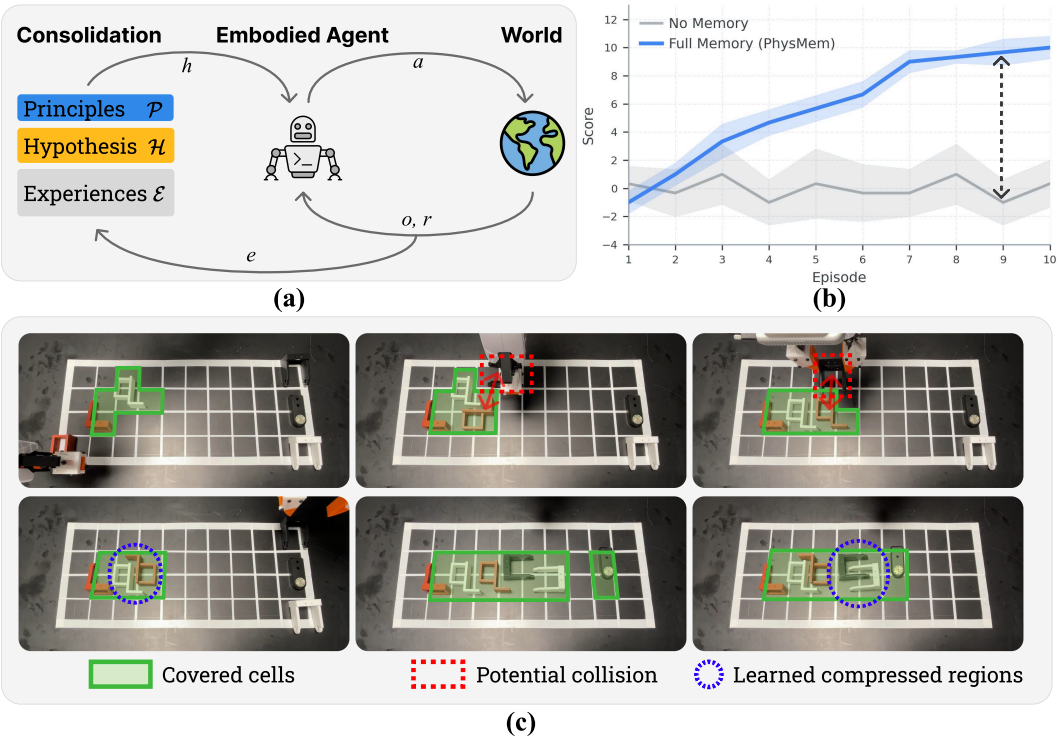


Figure 1: *PhysMem* learns physical principles through interaction. (a) The memory system maintains principles $\mathcal{P}$, hypotheses $\mathcal{H}$, and experiences $\mathcal{E}$; world feedback refines knowledge. **(b)** *PhysMem* (blue) improves continuously on *Parts Organization* while the no-memory baseline (gray) remains flat. **(c)** Qualitative results: green = covered cells, red dashed = collisions avoided, blue circles = learned strategies.

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; however, 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 enables VLM robot planners to learn physical principles from interaction at test time, without updating model parameters. The system records experiences, generates candidate hypotheses, and verifies them through targeted interaction before promoting validated knowledge to guide future decisions. A central design choice is verification before application: the system tests hypotheses against new observations rather than applying retrieved experience directly, reducing rigid reliance on prior experience when 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 compared to 23% for direct experience retrieval, and real-world experiments show consistent improvement over 30-minute deployment sessions. The project website:https://phys-mem.github.io/

1 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 (Driess et al., 2023; Brohan et al., 2023; Gemini Robotics Team et al., 2025). However, when deployed as robot planners, they often struggle to apply these principles in specific situations. A VLM planner may understand friction in the abstract but misjudge how far a ball will roll on a given surface, or recognize stability as a concept but fail to identify which irregular stone will provide a stable foundation. This gap between declarative knowledge and physical grounding is well documented (Zhang et al., 2026; Kawaharazuka et al., 2025), and its consequences compound in planning, where a single misjudgment about contact or dynamics can invalidate an entire action sequence. This paper studies whether a VLM robot planner can acquire useful physical understanding during deployment through interaction without updating model parameters. To isolate reasoning from execution, we separate high level planning from low level control, so improvements reflect better physical understanding rather than better motor policies.

We focus on settings where correct strategies cannot be inferred from vision alone. We consider three tasks: *Parts Organization* (learning spatial relationships between irregular shapes), *Ball Navigation* (discovering contact dynamics through trial), and *Balanced Stacking* (stability judgments about stones whose mass distribution only reveals itself upon contact). Initial attempts fail because the planner lacks physical intuition that only interaction can provide.

A natural approach is to add memory, storing experiences and retrieving them when similar situations arise (Blundell et al., 2016; Pritzel et al., 2017; Shi et al., 2025; Sridhar et al., 2025). However, embodied situations rarely repeat exactly, and retrieval-augmented approaches (Lewis et al., 2020; Gao et al., 2023; Li et al., 2025) apply past experience without verifying whether it still holds. In our experiments, direct episodic replay reaches only 23% success on a controlled benchmark, while principled abstraction reaches 76%. The issue is that unverified memory leads to rigid behavior: a small change in friction can turn a useful heuristic into a repeated error. Following Popper's falsificationism (Popper, 1959), we hypothesize that verified abstraction—forming compact principles and revising them when evidence contradicts expectation—outperforms increased recall.

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 (Liu et al., 2021). The resulting principle set is human-readable and can be inspected, edited, or transferred to new settings.

Across real-world runs, *PhysMem* enables clear learning curves on all three tasks, with principled abstraction outperforming direct retrieval by 53% in simulation across four VLM backbones. Transfer experiments show that prior principles help when physics are similar but test-time adaptation becomes essential when dynamics change. The resulting principles are both effective for decision-making and interpretable for human inspection.

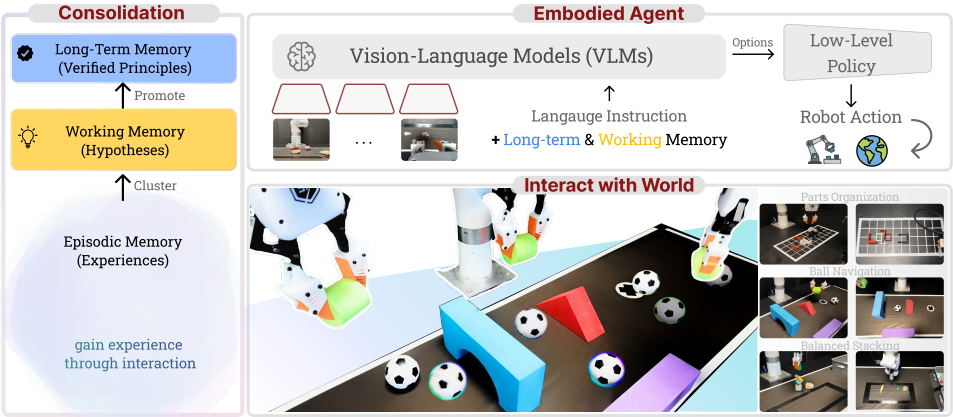


Figure 2: System overview of *PhysMem*. A three-tier memory (episodic memory/working memory/long-term memory) consolidates interaction into hypotheses and verified principles, which condition the VLM planner; outcomes feed back to refine memory.

2 Related Work

Vision-Language-Action Models. VLMs have enabled language-conditioned robot planning (Ahn et al., 2022; Driess et al., 2023; Huang et al., 2022; Liang et al., 2023), with web-scale pretraining transferring to robotic control (Brohan et al., 2022; 2023; Collaboration et al., 2023; Octo Model Team et al., 2024; Kim et al., 2024) and recent work advancing generalist policies through flow matching (Black et al.), embodied reasoning (Gemini Robotics Team et al., 2025; Nvidia et al., 2025; Figure AI Team, 2025), and chain-of-thought mechanisms (Zawalski et al., 2024; Zhao et al., 2025b). However, VLM general capabilities poorly predict downstream VLA performance (Kawaharazuka et al., 2025; Li et al., 2024b; Zhang et al., 2026), revealing a persistent gap between declarative knowledge and physical grounding. *PhysMem* fills this gap through test-time principle learning that grounds VLM reasoning in interaction experience.

Test-Time Learning. Meta-learning (Finn et al., 2017a;b), domain randomization (Tobin et al., 2017; Andrychowicz et al., 2020), and in-context learning (Chen et al., 2021; Laskin et al., 2023) enable adaptation without full retraining. Recent VLA approaches explore online adaptation through reinforcement learning (Luo et al., 2025; Xu et al.), language corrections (Shi et al., 2024), deployment experience (Physical Intelligence Team et al., 2025), and test-time training (Sun et al., 2020; Kim et al., 2025; Yoo et al., 2025). These methods focus on implicit policy adjustment; *PhysMem* instead generates human-readable hypotheses that can be inspected, edited, or transferred.

Memory in Embodied Agents. Memory systems enhance planning through experience accumulation (Blundell et al., 2016; Pritzel et al., 2017; Fang et al., 2019), with recent VLA integrations via hierarchical retrieval (Sridhar et al., 2025), dual memory banks (Shi et al., 2025; Li et al., 2024a), and visual foundation models (Fang et al., 2025). World models (Hafner et al., 2023; Bruce et al., 2024; Agarwal et al., 2025; SIMA Team et al., 2025) enable planning through imagination, while reflection (Shinn et al., 2023; Madaan et al., 2023), retrieval augmentation (Lewis et al., 2020; Gao et al., 2023; Li et al., 2025), and lifelong learning (Liu et al., 2021; Meng et al., 2025; Zhang et al., 2024) offer complementary strategies. However, failure-only reflection misses success patterns, and blind retrieval applies experience without verifying relevance. *PhysMem* addresses this through its scientific memory loop: generating hypotheses from both successes and failures, verifying through targeted interaction, and promoting only validated principles.

3 Method

We present *PhysMem*, a test-time memory framework that enables VLM robot planners to learn physical principles through interaction (Figure 2). The system consists of three components: (1) a VLM-based **Planner** that generates decisions given observations and principles, (2) a three-tier **Memory** (episodic, working, long-term) that evolves through a scientific memory loop, and (3) an

Algorithm 1 Scientific Memory Loop

Require: Experience buffer $\mathcal{E}$, Hypothesis store $\mathcal{H}$, Principle store $\mathcal{P}$
Require: Reflection model f_ϕ, thresholds $\tau_p, \tau_r, n_{\min}$

function RECORDEXPERIENCE($o, \omega, r, c, \mathbf{s}, \mathcal{P}_{\text{active}}$)
  $e \leftarrow (o, \omega, r, c, \mathbf{s})$
  $\rho \leftarrow \text{Resonance}(e, \mathcal{P}_{\text{active}})$ ▷ Eq. 3
  $\mathcal{E} \leftarrow \mathcal{E} \cup \{e\}$
  if $\rho < 1$ **then** ▷ Surprising experience
    TRIGGERCONSOLIDATION(e)
  else
    REINFORCEACTIVEPRINCIPLES($\mathcal{P}_{\text{active}}$)
  end if
end function

function CONSOLIDATE($\mathcal{E}$)
  $\{\mathcal{C}_k\} \leftarrow \text{ClusterBySymbolicState}(\mathcal{E})$
  for each cluster $\mathcal{C}_k$ with $|\mathcal{C}_k| \geq n_{\min}$ **do**
    $\mathcal{H}_k \leftarrow f_\phi(\mathcal{C}_k, \mathcal{P}, \mathcal{H})$ ▷ Eq. 4
    $\mathcal{H} \leftarrow \mathcal{H} \cup \mathcal{H}_k$
  end for
end function

function ATTRIBUTEANDPROMOTE($\mathcal{E}, \mathcal{H}, \mathcal{P}$)
  for each hypothesis $h \in \mathcal{H}$ **do**
    Update $\text{conf}(h)$ via action-level attribution ▷ Eq. 5
    if $\text{conf}(h) \geq \tau_p$ **and** $|\mathcal{E}_{\text{support}}| \geq 3$ **then**
      $\mathcal{P} \leftarrow \mathcal{P} \cup \{h\}$ ▷ Promote to principle
      $\mathcal{E} \leftarrow \mathcal{E} \setminus \mathcal{E}_{\text{folded}}$ ▷ Memory folding
      $\mathcal{H} \leftarrow \mathcal{H} \setminus \{h\}$
    else if $\text{conf}(h) \leq \tau_r$ **and** $|\mathcal{E}_{\text{contradict}}| \geq 2$ **then**
      $\mathcal{H} \leftarrow \mathcal{H} \setminus \{h\}$ ▷ Refute hypothesis
    end if
  end for
end function

Executor that carries out actions via low-level policies. Unlike retrieval-augmented approaches that blindly apply retrieved knowledge, *PhysMem* verifies hypotheses before promotion.

3.1 PROBLEM FORMULATION

We formulate manipulation using the options framework (Sutton et al., 1999), where a VLM high-level policy π_θ^H selects temporally-extended actions (options) based on observations $o_t \in \mathcal{O}$, task description τ, and learned principles:

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

where $\mathcal{P}_t \subseteq \mathcal{P}$ denotes active principles retrieved at time t, and a low-level policy π^L executes selected options. Since physical properties vary across environments and cannot be fully captured by pre-trained knowledge θ alone, our goal is to learn a principle set $\mathcal{P}^*$ at test time such that:

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

without modifying the VLM parameters θ.

3.2 SCIENTIFIC MEMORY LOOP

The scientific memory loop is the core contribution of *PhysMem*. Inspired by the scientific method, it transforms raw experiences into verified principles through four phases: experience collection, hypothesis generation, attribution, and principle promotion. Algorithm 1 provides the pseudocode.

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

A key 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 reso-

nance score:

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

where 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.

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:

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

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)

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:

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

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

Verification and Principle Promotion. Hypotheses achieving high confidence (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:

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

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

3.3 MEMORY ARCHITECTURE AND RETRIEVAL

PhysMem organizes memory in three tiers: (1) **Episodic memory** stores raw experiences bounded by capacity $N_{\max}$; (2) **Working memory** holds hypotheses under test with confidence scores; (3) **Long-term memory** contains verified principles with importance scores that decay ($\gamma = 0.995$) to forget outdated knowledge. At inference, relevant principles are retrieved via symbolic filtering and semantic ranking, with the top-k principles injected into the VLM prompt. When resonance is low, the system prioritizes fresh learning over applying potentially misleading prior knowledge. Full details are in Appendix A.

4 EXPERIMENTS

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. Our design separates high-level VLM reasoning from low-level control (motion planning), isolating reasoning improvements; we evaluate both task success and plan quality.

4.1 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,

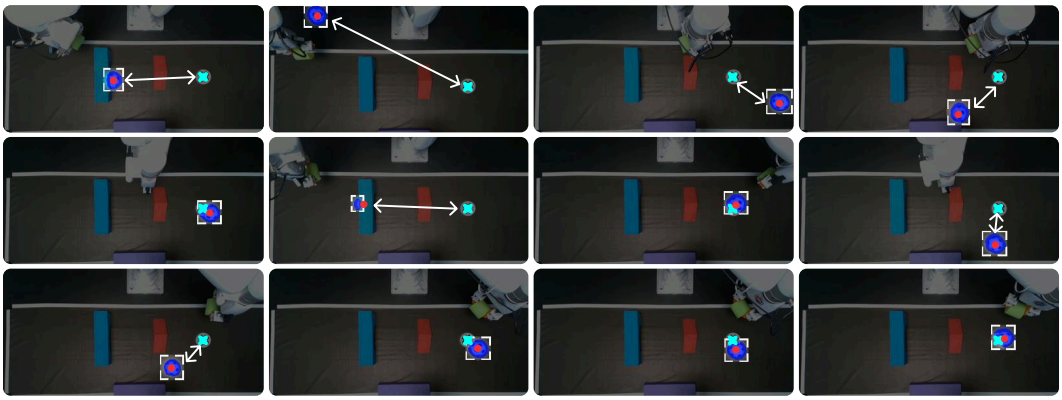


Figure 3: Qualitative progression on Ball Navigation. Twelve consecutive episodes showing the ball's trajectory through obstacles. Early episodes show failed attempts; later episodes demonstrate learned strategies for navigating obstacles and reaching the target.

we use Gemini-3.0-Flash (DeepMind, 2025a) with thinking mode. Hypothesis generation uses Qwen3-VL (Bai et al., 2025), 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. We provide a visual overview in Appendix E.

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.

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 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.

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 are in Appendix G.

4.2 Simulation Benchmark

We complement real-world evaluation with the Reflect-VLM brick insertion benchmark (Feng et al., 2025), 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 are in Appendix D.

Our experiments address four questions:

1. **Physics Understanding**: Can *PhysMem* improve the VLM's understanding of task-specific physical properties through interaction? (Section 4.3)
2. **Decision Quality**: Does improved physics understanding lead to better embodied decision-making over time? (Section 4.4)
3. **Memory Transfer**: Can experience-derived memory provide benefits in out-of-distribution scenarios, and when does it fail? (Section 4.5)
4. **Architecture Design**: Which components of *PhysMem* are essential, and why do we need memory compression and forgetting? (Section 4.8)

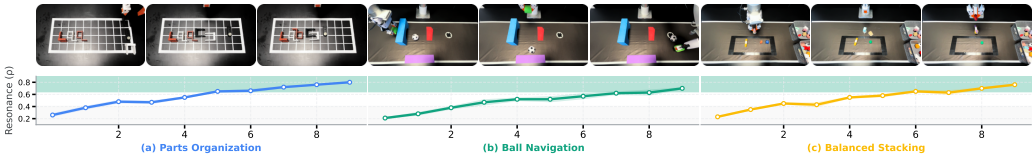


Figure 4: Resonance evolution across tasks. ρ rises as predictions match outcomes (reactive $\rightarrow$ principled).

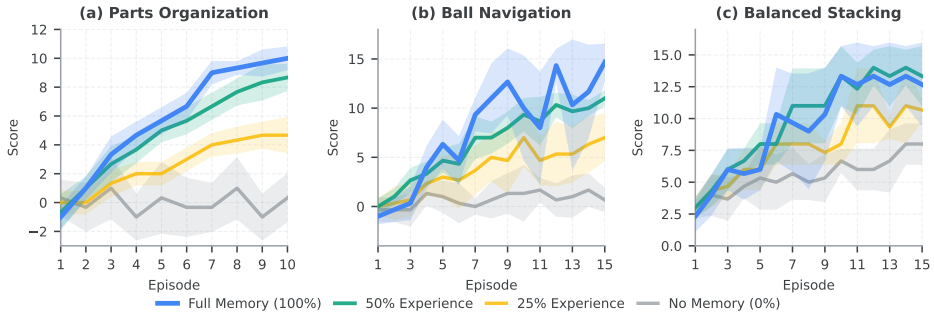


Figure 5: Test-time evolution across experience utilization levels (3 runs, shaded = std). Blue: 100%, Green: 50%, Yellow: 25%, Gray: 0%. Without memory, performance stays flat; complex dynamics (Ball Navigation) benefit most from full experience.

4.3 Learned Physical Principles

We examine whether *PhysMem* produces reasoning grounded in physical reality. Figure 4 shows the resonance score ρ, which measures prediction-outcome alignment. All three tasks exhibit a consistent pattern: ρ rises from $\approx$0.2 in early episodes to 0.9 by episode 10, crossing the $\rho = 0.7$ threshold around episode 5–6.

Episodes in the rational regime ($\rho > 0.7$) achieve 2.3$\times$ higher scores than episodes 1–3, and the monotonic increase in ρ confirms that *PhysMem* accumulates verified physical understanding rather than overfitting to narrow solutions.

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. Example principles are in Appendix E.5.

4.4 Test-Time Evolution

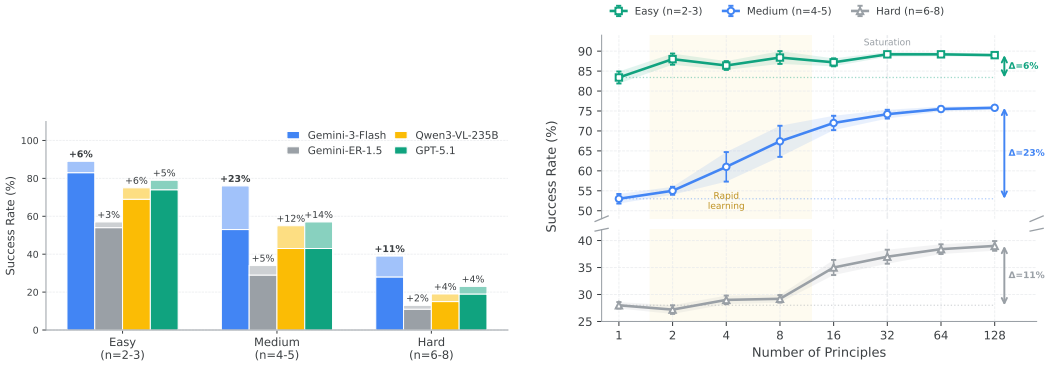
To evaluate how *PhysMem* actually improves through interaction and how much experience is needed, we evaluate performance under different experience utilization levels (0%, 25%, 50%, 100%), with 3 runs per condition. The test-time learning curve is shown in Figure 5.

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.

4.5 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 1 and details in Appendix E.6.

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$\rightarrow$6.9 by avoiding common packing errors. In contrast, Ball Navigation with new ball types shows that prior knowledge matches zero-shot perfor-



(a) Success rate vs. difficulty (across VLMs). **(b)** Success vs. # principles (Gemini-3-Flash).

Figure 6: Scaling results. Left: baseline success and gains from test-time learning across VLMs/difficulties. Right: success improves rapidly with the first few principles, then saturates.

mance: 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, where we report success rates (binary criterion: all bricks correctly placed) for large-scale comparison across VLMs and difficulty levels.

4.6 Scaling Across VLMs

We evaluate four VLMs with thinking mode: Gemini-3-Flash (DeepMind, 2025a), Gemini-ER-1.5 (DeepMind, 2025b), GPT-5.1 (OpenAI, 2025), and Qwen3-VL-235B (Bai et al., 2025) (100 episodes each). Figure 6 shows that test-time learning benefits *scale with VLM capability*: on medium difficulty, Gemini-3-Flash improves by +23% (53%→76%), GPT-5.1 by +14%, Qwen3-VL by +12%, and Gemini-ER-1.5 by only +5%. Test-time learning amplifies existing capabilities rather than compensating for fundamental limitations. All difficulty levels benefit, with hard tasks improving +11% through AVOID hypotheses that prevent repeated mistakes.

4.7 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). See Fig. 6 (right).

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% despite their difficulty; 68% of learned principles are avoidance constraints (vs. 41% on medium), confirming that learning what *not* to do improves success even when the VLM cannot fully solve the task.

Table 1: 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.

		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**

Table 2: Ablation study across difficulty levels. Success rates and token consumption for brick insertion (Gemini-3-Flash). 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×

4.8 Ablations

Which components of *PhysMem* are essential, and how does their importance vary with task difficulty? We ablate key modules across all difficulty levels (100 episodes each, Gemini-3-Flash), and report the results in Table 2.

Principle abstraction is essential. Direct retrieval degrades severely as task complexity increases: 48% on easy, 23% on medium, and only 8% on hard. This confirms that state matching becomes fragile with complex dependency graphs, while principled abstraction maintains robust performance.

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 presents an accuracy-efficiency trade-off. Without forgetting, retaining all experiences provides marginal gains on simpler tasks (+2% on easy and medium) but degrades performance on complex tasks (-3% on hard) as accumulated noise interferes with decision-making. Given the 3.4× token overhead, forgetting provides a favorable balance. Full ablations appear in Appendix B.

5 Discussion & Limitation

Sensing and Representation Our experiments use visual observations, yet richer modalities—tactile (Huang et al., 2025), audio (Liu & Chen, 2024), and active perception (Sripada et al., 2024)—could accelerate principle discovery. Similarly, text-based principles excel at discrete rules but struggle with continuous dynamics (Balazadeh et al., 2025; Tian et al., 2025). Visual chain-of-thought (Zhao et al., 2025a), continuous visual tokens (Qin et al., 2025), and latent world models (Bi et al., 2025; Hafner et al., 2023; Bruce et al., 2024; Ball et al., 2025) suggest future systems may bypass language for physics prediction. *PhysMem*'s architecture naturally extends to both richer inputs and non-textual hypothesis representations.

Limitations Our work focuses on high-level planning; integrating learned principles into VLA execution remains open (Physical Intelligence Team et al., 2025; Gemini Robotics Team et al., 2025). For continuous deployment, environment reset remains a practical challenge—our system requires human intervention when objects fall or parts break. Autonomous recovery would enable truly lifelong learning (Liu et al., 2021).

6 Conclusion

We presented *PhysMem*, a test-time memory framework that enables VLM robot planners to learn physical principles through interaction. Our scientific loop of hypothesis generation, verification, and promotion produces knowledge that is both effective and interpretable. Experiments across three real-world tasks and simulation benchmarks validate that principled abstraction outperforms raw retrieval, and that the ability to update beliefs is essential for robust adaptation. We hope this work inspires further research on robots that grow wiser through experience.

REFERENCES

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.

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gober, 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)*, pp. 287–318, 2022.

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.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Zesen Cheng, Lianghao 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.

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*, pp. 7318–7328, 2025.

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 Buttimore, 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 Gharamani, Raia Hadsell, Aäron van den Oord, Inbar Mosseri, Adrian Bolton, Satinder Singh, and Tim Rocktäschel. Genie 3: A new frontier for world models. 2025.

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.

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*.

Charles Blundell, Benigno Uria, 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.

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.

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.

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.

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.

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 Büchler, 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, João Silvério, Joey Hejna, Jonathan Booher, 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, Minho 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 Sundaresan, Qiuyu Chen, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Mart'in-Mart'in, Rohan Baijal, 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.

Google DeepMind. Gemini 3 flash: Frontier intelligence built for speed, 2025a. URL https://blog.google/products-and-platforms/products/gemini/gemini-3-flash/. Accessed: 2026-01-21.

Google DeepMind. Gemini embodied reasoning 1.5: Multi-step reasoning for robotic planning, 2025b. URL https://deepmind.google/technologies/gemini/. Accessed: 2026-01-21.

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.

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.

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)*, pp. 538–547, 2019.

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.

Figure AI Team. Helix: A vision-language-action model for generalist humanoid control. *Technical Report*, 2025.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning (ICML)*, pp. 1126–1135, 2017a.

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)*, pp. 357–368, 2017b.

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.

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.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

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.

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)*, pp. 1769–1782, 2022.

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.

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.

Sungjune 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.

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.

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.

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, pp. 9459–9474, 2020.

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)*, pp. 17913–17922, 2024a.

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.

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*, 2024b.

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)*, pp. 9493–9500, 2023.

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.

Jiaxun Liu and Boyuan Chen. SonicSense: Object perception from in-hand acoustic vibration. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2024.

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.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, 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.

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.

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.

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.

OpenAI. GPT-5.1: Advanced multimodal reasoning model, 2025. URL https://openai.com/gpt-5.

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.

Karl Popper. *The Logic of Scientific Discovery*. Routledge, 1959.

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)*, pp. 2827–2836, 2017.

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.

Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. Memoryvla: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2508.19236*, 2025.

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.

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.

SIMA Team, Adrian Bolton, Alexander Lerchner, et al. SIMA 2: A generalist embodied agent for virtual worlds. *arXiv preprint arXiv:2512.04797*, 2025.

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.

Venkatesh Sripada, Samuel Carter, Frank Guerin, and Amir Ghalamzan. AP-VLM: Active perception enabled by vision-language models. *arXiv preprint arXiv:2409.17641*, 2024.

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)*, pp. 9229–9248. PMLR, 2020.

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.

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.

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)*, pp. 23–30, 2017.

Shoukai Xu, Mingkui 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*.

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.

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.

Jesse Zhang, Minho 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.

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.

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*, pp. 1702–1713, 2025a.

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)*, 2025b.

A METHOD IMPLEMENTATION DETAILS

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

A.1 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

A.2 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

A.3 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

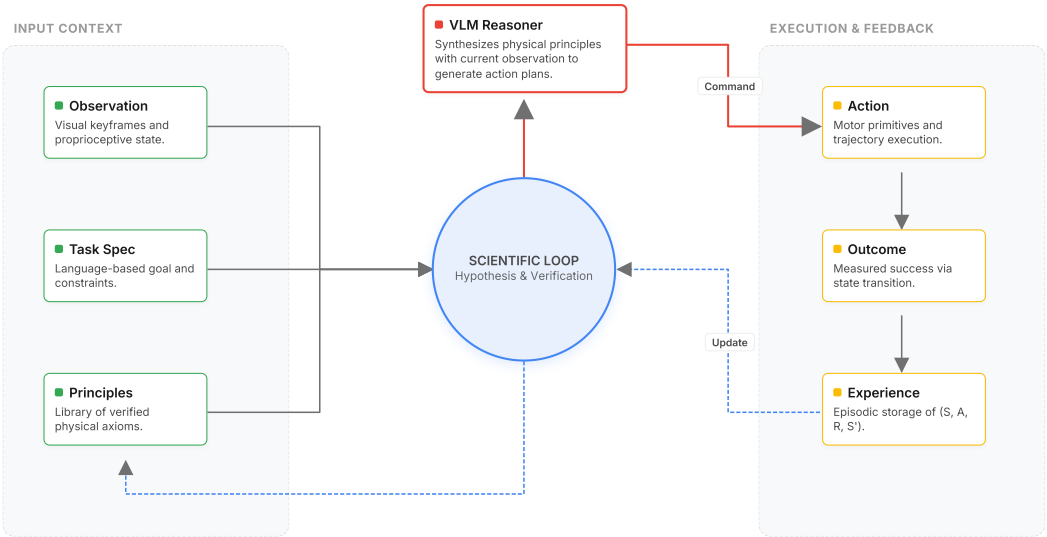


Figure 7: PhysMem overall pipeline. PhysMem enables VLM robot planners to learn physical principles through interaction. The scientific loop transforms raw experiences into verified knowledge: collecting observations, generating hypotheses, verifying through action-level attribution, and promoting validated principles to guide future decisions.

A.4 Confidence Update Rules

The action-level attribution updates hypothesis confidence as follows:

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

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

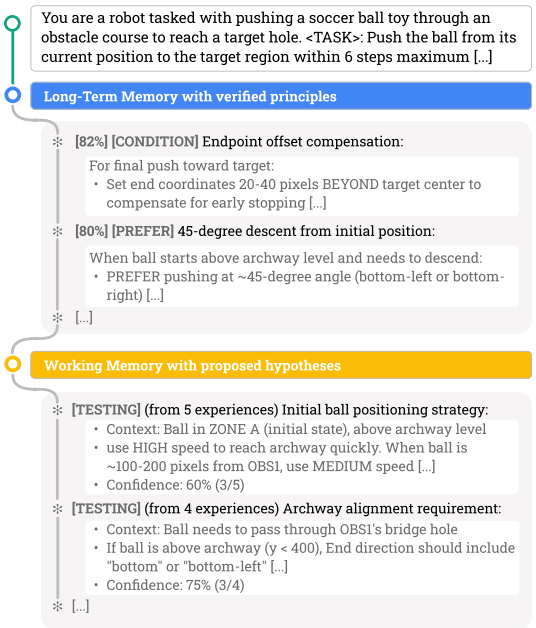


Figure 8: Memory injection into VLM prompts. Verified principles (blue) and active hypotheses (yellow) are injected into the VLM planner's context. Each entry includes a confidence score, typed constraint (AVOID, PREFER, SEQUENCE), and natural language description.

A.5 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:

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

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

A.6 HYPERPARAMETER SUMMARY

Table 3 lists all hyperparameters used in our experiments.

Table 3: Hyperparameter settings.

Parameter	Value	Description
Memory Capacity		
$N_{\max}$	3000	Max episodic experiences
Folded TTL	100	Episodes before folded exp. removal
Consolidation		
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
Principle Management		
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

Table 4: 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**.

	Easy (2–3 bricks)			Medium (4–5 bricks)			Hard (6–8 bricks)		
Configuration	Success	Δ	Tokens	Success	Δ	Tokens	Success	Δ	Tokens
PhysMem (Full)	89%	—	1.0×	76%	—	1.0×	**39%**	—	1.0×
Memory Architecture Ablations									
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×
Mechanism Ablations									
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×
Simplified Baselines									
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×

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).

B.1 Full Results

Table 4 presents complete ablation results with token consumption analysis.

B.2 Analysis by Component

B.2.1 Memory Architecture Ablations

Episodic Memory. Without episodic memory, the system cannot store raw experiences, eliminating the foundation for learning. Performance drops severely across all difficulties: 54% on easy (-35), 37% on medium (-39), and 14% on hard (-25). This ablation confirms that experience storage is fundamental to the system. Token consumption is minimal (0.2–0.3×) because there is nothing to process, but this "efficiency" is meaningless without learning capability.

Working Memory. Working memory stores unverified hypotheses for active testing. Its importance increases with task complexity:

- **Easy** (-5%): Simple tasks have obvious solutions. Hypothesis exploration adds moderate value.
- **Medium** (-7%): Benefit from hypothesis testing becomes more apparent as task structure requires exploration.
- **Hard** (-11%): Hypothesis exploration becomes crucial for discovering complex constraints. The system must actively test theories like "does brown block pink?" through deliberate action.

Long-term Memory. Without long-term memory, verified hypotheses cannot be consolidated into persistent principles. The system must re-learn patterns each session:

- **Easy** (-8%): Simple patterns can be re-discovered quickly, but consolidation still provides measurable benefit.
- **Medium** (-12%): Without principle retention, the system repeatedly makes the same mistakes before re-learning.

- **Hard** (-13%): Complex dependency chains require stable principles; losing them between episodes severely impacts performance.

Token consumption increases (1.15–1.35$\times$) because the system must repeatedly generate and test hypotheses that would otherwise be available as stable principles.

B.2.2 MECHANISM ABLATIONS

Resonance Filtering. Resonance filtering retrieves principles based on prediction-outcome alignment rather than raw similarity. Its importance scales with difficulty:

- **Easy** (-8%): Few principles accumulate, but filtering still provides noticeable benefit.
- **Medium** (-18%): Moderate principle count creates potential for confusion. Retrieving principles learned from different contexts misleads planning.
- **Hard** (-18%): Many principles accumulate (especially AVOID constraints from failures). Without proper filtering, the planner receives conflicting guidance.

Verification. Verification validates hypotheses against subsequent experiences before promoting them to principles:

- **Easy** (-4%): Simple hypotheses are rarely incorrect. Natural verification through task success mostly suffices.
- **Medium** (-12%): More complex hypotheses require explicit verification. Incorrect hypotheses about partial orderings mislead the planner.
- **Hard** (-12%): Complex dependency hypotheses are most error-prone and benefit significantly from explicit verification.

Forgetting. The Ebbinghaus-inspired forgetting mechanism decays old experiences and removes redundant principles. Removing forgetting shows an interesting accuracy-efficiency trade-off:

- **Easy** ($+2\%$): Additional experiences provide marginal benefit; 1.8$\times$ token overhead.
- **Medium** ($+2\%$): Similar marginal gain but 3.4$\times$ token overhead makes the trade-off unfavorable.
- **Hard** (-3%): Accumulated noise actually *hurts* performance while consuming 4.8$\times$ tokens.

This confirms that forgetting provides a favorable accuracy-efficiency balance, especially for complex tasks where noise accumulation degrades decision quality.

Folding. Experience folding compresses similar experiences into representative clusters. Without folding, the system retains all individual experiences:

- **Easy** (-1%): Minimal accuracy impact; 1.4$\times$ tokens.
- **Medium** (-2%): Slight accuracy drop; 2.1$\times$ tokens.
- **Hard** (-2%): Similar pattern; 2.8$\times$ tokens.

Folding primarily improves efficiency rather than accuracy, compressing redundant experiences while preserving essential patterns.

B.2.3 SIMPLIFIED BASELINES

Direct Retrieval. Direct retrieval stores raw experiences and retrieves the most similar one based on state embedding distance. This approach fails across all difficulty levels: 48% on easy (-41), 23% on medium (-53), and only 8% on hard (-31). The failure mechanism confirms that small state differences cause retrieved actions to be inappropriate, validating the need for principle abstraction.

Only Episodic. This baseline uses only episodic memory without any symbolic abstraction (no hypotheses or principles). Performance is slightly better than direct retrieval (52%, 28%, 10%) because it maintains richer experience representations, but still fails to generalize effectively. The gap from the full system (-37 to -48) confirms that symbolic abstraction is essential for robust learning.

Only Principles. This baseline uses pre-loaded static principles without adaptation capability. It achieves moderate performance (67%, 51%, 21%) on tasks where principles transfer well, but cannot adapt to novel situations. The -18 to -25 gap from the full system demonstrates that test-time adaptation is necessary even when good prior knowledge exists.

B.3 KEY FINDINGS

1. **Three-tier memory hierarchy is essential**: Removing any memory tier (episodic, working, or long-term) causes significant degradation. Episodic memory provides the foundation (-35 to -39), working memory enables exploration (-5 to -11), and long-term memory enables consolidation (-8 to -13).
2. **Principle abstraction outperforms experience matching**: Both Direct Retrieval (-41 to -53) and Only Episodic (-37 to -48) baselines fail severely, validating that abstract principles generalize better than raw experience retrieval.
3. **Component importance scales with task complexity**: Resonance filtering (-8 to -18), verification (-4 to -12), and working memory (-5 to -11) all show increasing importance on harder tasks.
4. **Forgetting balances accuracy and efficiency**: Without forgetting, simpler tasks gain marginally (+2%) but harder tasks degrade (-3%) while consuming 3–5$\times$ more tokens. Forgetting provides a favorable trade-off across difficulty levels.
5. **Static knowledge requires adaptation**: The Only Principles baseline achieves moderate performance but cannot match the full system, confirming that test-time adaptation complements prior knowledge.

C HYPOTHESIS AND PRINCIPLE QUANTITY ANALYSIS

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

Table 5: 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).

D 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.

D.1 ENVIRONMENT OVERVIEW

We use the Reflect-VLM simulation environment (Feng et al., 2025), 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

D.2 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.

D.2.1 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.

Difficulty Levels. We evaluate across three difficulty levels based on brick count and dependency complexity (Figure 9):

- **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.

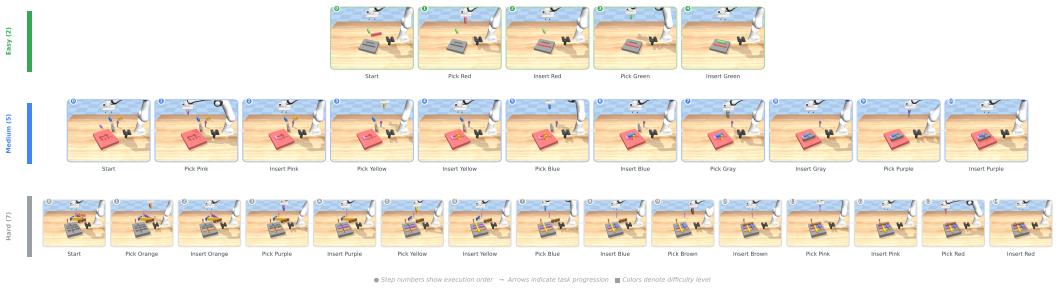


Figure 9: Assembly task demonstration across difficulty levels. Each row shows the oracle planner's execution sequence from initial state to completion. Step numbers indicate execution order. **Easy** (green, 2 bricks): 5 steps total, minimal dependencies. **Medium** (blue, 5 bricks): 11 steps with moderate 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.

D.2.2 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").

D.2.3 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.

D.2.4 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.

D.2.5 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.

D.3 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.4 SIMULATION SCALING EXPERIMENTS

The simulation scaling experiments (VLM difficulty scaling and principle scaling) are presented in the main text (Section 4.6 and Section 4.7). 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.

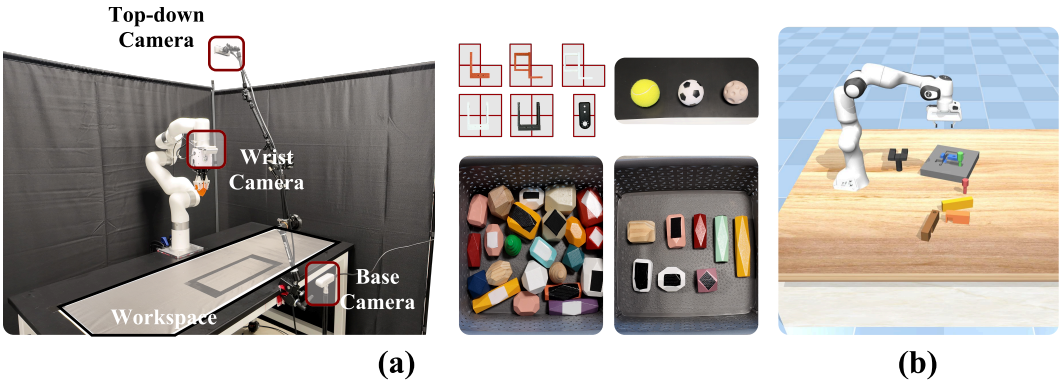


Figure 10: Experimental environment. (a) **Real-world setup**: An xArm6 robot arm with TPU-printed fin-ray soft grippers operates in a tabletop workspace. The multi-camera perception system includes top-down, wrist-mounted, and base cameras (Intel RealSense D435). The workspace is enclosed with a backdrop for consistent lighting. (b) **Simulation environment**: The Reflect-VLM benchmark (Feng et al., 2025) uses a Franka Panda robot (different from our real-world xArm6) in MuJoCo for brick insertion tasks. We use this controlled environment for large-scale scaling experiments (see Section 4.6 and Section 4.7 in main text).

E 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.

E.1 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

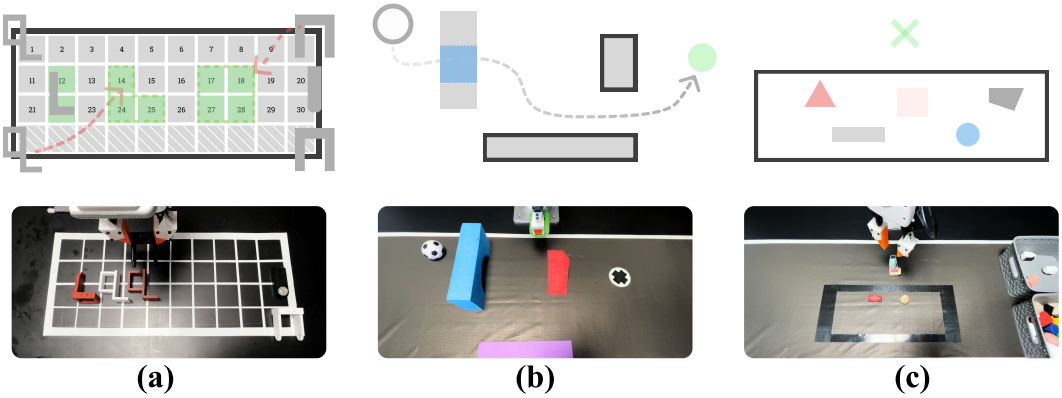


Figure 11: Real-world task overview. Top row shows abstract symbolic representations; the bottom row shows actual task setups. (a) **Parts Organization**: Place 6 irregularly-shaped parts (occupying 2–4 cells each) onto a 3×10 grid. The symbolic diagram shows grid numbering (1–30), occupied cells (green), and part placement trajectories (dashed arrows). (b) **Ball Navigation**: Push a soccer ball through an obstacle course including a "bridge hole" to reach the target within 6 steps. The diagram shows obstacles (blue archway, red/purple blockers), target region (circle with cross), and example trajectory (dashed line). (c) **Balanced Stacking**: Stack 5 balance stones into a stable tower. The diagram shows abstract stone shapes with varying sizes and the target stacking position.

E.2 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.

E.2.1 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.

E.2.2 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.

E.2.3 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 {`color`}-{`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]]` $\rightarrow$ `[[b,d],[a,c]]` (original $a \rightarrow c$, $b \rightarrow a$, $c \rightarrow d$, $d \rightarrow b$)
- **180° CCW**: `[[a,b],[c,d]]` $\rightarrow$ `[[d,c],[b,a]]` (original $a \rightarrow d$, $b \rightarrow c$, $c \rightarrow b$, $d \rightarrow a$)
- **270° CCW**: `[[a,b],[c,d]]` $\rightarrow$ `[[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]`.

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

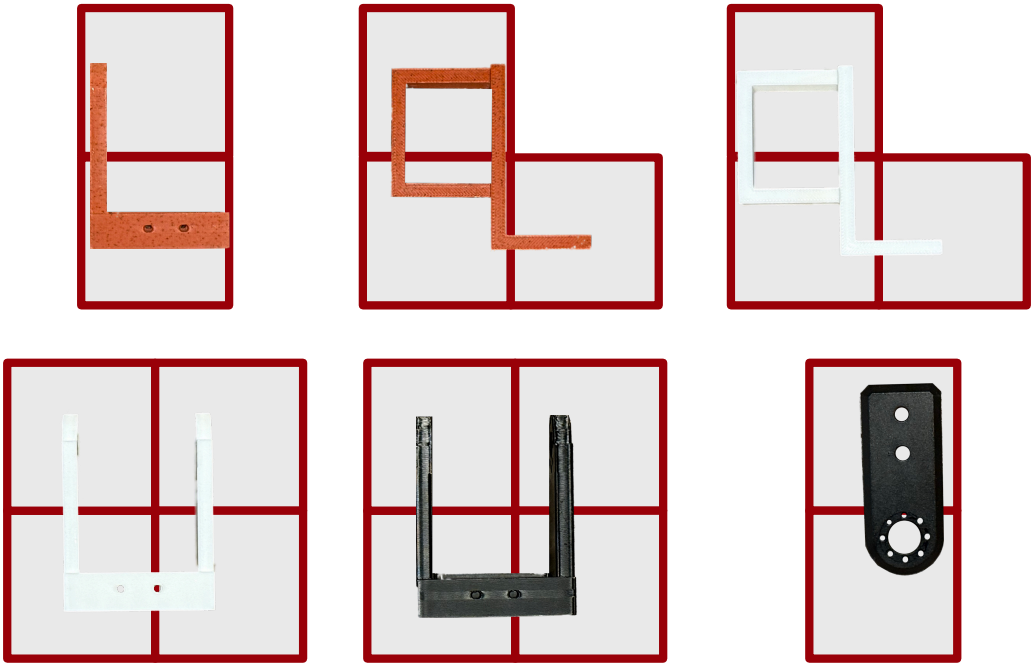


Figure 12: Parts Organization: Component shapes and grid occupancy. Each panel shows a 3D-printed part overlaid on its 2×2 template grid, illustrating which cells (`[a,b,c,d]`) each part occupies. **Top row**: `red-L` (2 cells, occupies `[a,c]`), `red-q` (3 cells, `[a,c,d]`), `white-q` with `red-L` interlocking 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).

E.2.4 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:

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

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.

E.2.5 SAFETY MECHANISM

Before executing each placement, the system performs collision checking using Segment Anything Model (SAM) (Kirillov et al., 2023):

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.

E.2.6 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.

E.2.7 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.

E.2.8 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. For the full coordinate system reference, spatial reasoning guidelines, and decision rules that can be directly incorporated into VLM prompts, see Section G.1 (Parts Organization Detailed Specifications).

E.3 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.

E.3.1 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.

E.3.2 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.

E.3.3 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.

E.3.4 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.

E.3.5 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 $\rightarrow$ 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

E.3.6 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 ($y \approx 600$–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

E.3.7 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 <100px: 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. For the full obstacle layout, spatial zone definitions, and stage-specific decision rules that can be directly incorporated into VLM prompts, see Section G.2 (Ball Navigation Detailed Specifications).

E.4 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.

E.4.1 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.

E.4.2 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.

E.4.3 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.

E.4.4 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

E.4.5 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.

E.4.6 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

E.4.7 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. For the full stone naming convention, friction compatibility matrix, and step-by-step selection rules that can be directly incorporated into VLM prompts, see Section G.3 (Balanced Stacking Detailed Specifications).

E.5 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"

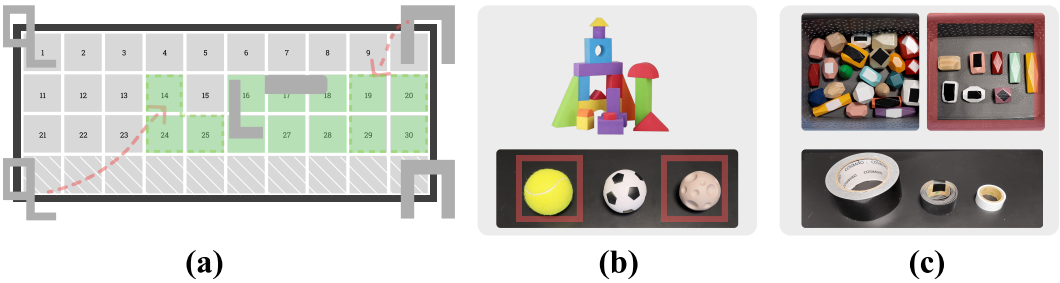


Figure 13: Out-of-distribution experimental settings and props. (a) **Parts Organization OOD**: Initial grid configuration is modified with parts pre-placed at different positions. Green cells (14–16, 23–25) are pre-occupied; hatched area indicates the constrained placement region (± 3 columns from occupied 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.

E.6 Out-of-Distribution Experimental Settings

To evaluate memory transfer capabilities (Section 4.5 in main text), we designed out-of-distribution (OOD) variants for each task. Figure 13 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.

F CROSS-MODEL GENERALIZATION DETAILS

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

Table 6: 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.

G DETAILED TASK SPECIFICATIONS FOR VLM PLANNING

This section provides comprehensive task-specific reference materials designed to be directly incorporated into VLM planner prompts. These specifications include coordinate systems, naming conventions, spatial reasoning frameworks, and decision rules that enable more precise hypothesis formation and principle extraction. Each specification is formatted as a `promptlisting` block that can be directly inserted into the planner's context. We provide these detailed specifications for the three real-world manipulation tasks: Parts Organization (Tangram), Ball Navigation, and Balanced Stacking.

G.1 PARTS ORGANIZATION: DETAILED SPECIFICATIONS

The following specifications enable precise spatial reasoning about tangram piece interactions using a template-based coordinate system.

Parts Organization: Coordinate System and Part Properties

```
================================================================
PART NAMING AND COORDINATE SYSTEM REFERENCE
================================================================

**PART NAMING CONVENTION:**
Each part is identified by {color}-{shape}:
- Colors: black, white, red
- Shapes: L-shape, q-shape, I-shape, U-shape
- Parts: red-L, white-q, red-q, white-U, black-U, black-I

**TEMPLATE-BASED COORDINATE SYSTEM (2x2 Grid):**
Each part's local geometry uses a template grid for spatial reasoning
    :

    +---+---+
    | a | b |   <- top row [a, b]
    +---+---+
    | c | d |   <- bottom row [c, d]
    +---+---+

This template enables consistent reference to part sub-regions
regardless of rotation. For example, "the left vertical portion"
of a U-shape is always [a,c], whether at 0 deg or rotated.

**DEFAULT CELL OCCUPANCY (at 0 degrees):**

| Shape   | Cells Occupied | Description                          |
|---------|----------------|--------------------------------------|
| L-shape | [a, c, d]      | Vertical bar left, extension bottom-
    right |
| q-shape | [a, c, d]      | Similar to L, different internal
    geometry |
| I-shape | [a, c]         | Vertical bar, left column only       |
| U-shape | [a, b, c, d]   | All four cells (full template)       |

**ROTATION TRANSFORMATIONS (Counterclockwise):**

| Rotation | Template Becomes | Cell Mapping                    |
|----------|------------------|---------------------------------|
| 0 deg    | [[a,b],[c,d]]    | Original                        |
| 90 deg   | [[b,d],[a,c]]    | a->c, b->a, c->d, d->b          |
| 180 deg  | [[d,c],[b,a]]    | a->d, b->c, c->b, d->a          |
| 270 deg  | [[c,a],[d,b]]    | a->b, b->d, c->a, d->c          |
```

```
Example: After 180 deg rotation, the original [a] region is now
at position [d], and the original [d] region is now at position [a].

**PART-SPECIFIC INTERNAL BIASES:**
Even when occupying the same template cells, parts have internal
geometric biases that determine overlap compatibility:

| Part     | Internal Bias Description                      |
|----------|------------------------------------------------|
| black-U  | Physical mass biased toward [b,d];             |
| white-U  | [a,c] contains mostly empty U-opening space    |
| white-q  | Left edge aligns with left boundary of [a,c]   |
| red-q    | Right edge aligns with right boundary of [a,c] |
| black-I  | Top/right edges align with top-right of [a,c]  |

These biases determine whether adjacent parts can physically
    interlock when sharing grid cells.
```

Parts Organization: Spatial Reasoning Guidelines

```
================================================================
SPATIAL REASONING AND PLACEMENT RULES
================================================================

**USING COORDINATES FOR SPATIAL RELATIONSHIPS:**

When describing or reasoning about part interactions, use the
template regions [a,b,c,d]:

Examples:
- "The [c] region of white-q can overlap with [d] region of red-q"
- "When U-shape rotates 180 deg, its [a,c] can interlock with
   another U's [b,d]"
- "Avoid complete overlap of [a,c] regions between two q-shapes"

**KEY SPATIAL RULES:**

1. U-SHAPE INTERLOCKING:
   When two U-shaped parts must share adjacent cells, place the
   second at 180 deg rotation so its [a,c] region (empty opening)
   interlocks with the first U's [b,d] region (physical mass).

2. Q-SHAPE [a,c] CONFLICT:
   NEVER place two q-shaped parts with their [a,c] regions on
   overlapping grid cells. Despite identical template occupancy,
   their internal geometries (left-aligned vs right-aligned)
   create physical collisions.

3. Q-SHAPE COMPLEMENTARY PAIRING:
   White-q's [c] region CAN share cells with red-q's [d] region.
   Their opposite internal biases create physical clearance.

4. Q-SHAPE 180-DEGREE INTERLOCK:
   When a q-shape is rotated 180 deg, its [a] and [d] regions
   swap effective positions. This enables the rotated part's [a]
   to overlap with another q-shape's [d], and vice versa.

5. 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.

**CONTACT SURFACE COMPATIBILITY:**

| Base Shape | Can Overlap With          | Avoid Overlap With
    |
|------------|---------------------------|---------------------------|

| U [b,d]    | U [a,c] at 180 deg        | U [b,d] (both have mass)
    |
| q [a,c]    | q [d] of opposite color   | q [a,c] (geometry clash)
    |
| q [d]      | q [c], I [a,c]            | -
    |
| I [a,c]    | U [a,c] opening           | U [b,d] (blocked)
    |
```

G.2 BALL NAVIGATION: DETAILED SPECIFICATIONS

The following specifications enable spatial reasoning about ball navigation through the obstacle course using obstacle IDs and spatial zones.

Ball Navigation: Obstacle Layout and Coordinate System

```
================================================================
OBSTACLE LAYOUT AND COORDINATE REFERENCE
================================================================

**COORDINATE SYSTEM:**
- Format: (y, x) following Gemini visual grounding convention
- Range: 0-1000 (normalized)
- Origin: (0, 0) is TOP-LEFT
- Max: (1000, 1000) is BOTTOM-RIGHT

**WORKSPACE LAYOUT (Top-Down View):**

    x: 0                    500                    1000
    +--------------------------------------------------+
y:0 |                                                  |
    |  ========== (white boundary line) ==========     |
    |                                                  |
    |     +-------+                                    |
    |     |       |                                    |
200 |     | OBS 1 |        +-----+                     |
    |     | BLUE  |        | OBS |      +------+       |
    |     | arch  |        |  2  |      |TARGET|       |
400 |     |  [  ] |   O    | RED |      |  (+) |       |
    |     |       |  ball  |     |      +------+       |
    |     +-------+        +-----+                     |
600 |                                                  |
    |           +------------------+                   |
    |           |     OBS 3        |                   |
800 |           |     PURPLE       |                   |
    |           +------------------+                   |
    |  ========== (white boundary line) ==========     |
1000+--------------------------------------------------+

**OBSTACLE DEFINITIONS:**

| ID  | Color  | Type            | Position (y, x)     | Key Feature
              |
|-----|--------|-----------------|---------------------|-----------------------|

| OBS1| Blue   | Archway block   | (200-550, 100-300)  | Bridge hole
   y~400-480 |
| OBS2| Red    | Rectangular     | (200-500, 450-550)  | Solid
   blocker         |
| OBS3| Purple | Rectangular     | (650-850, 280-520)  | DANGER: ball
    stuck    |
| TGT | White  | Circle w/ cross | (350-450, 720-800)  | Goal
   position         |

**CRITICAL OBSTACLE NOTES:**

OBS1 (Blue Archway):
- Ball MUST pass through the "bridge hole" at y~400-480
- Cannot go over or around - passage through is mandatory
- Archway opening is approximately 80 pixels tall
```

```
OBS3 (Purple Block) - DANGER ZONE:
- If ball lands on TOP of OBS3 (y~600-700), it becomes STUCK
- Gripper cannot access ball from below when on OBS3 surface
- This is the most common failure mode - avoid at all costs!
```

Ball Navigation: Spatial Zones and Navigation Strategy

```
================================================================
SPATIAL ZONES FOR NAVIGATION PLANNING
================================================================

**ZONE DEFINITIONS:**

+--------------------------------------------------+
|                 ZONE A: Initial Area              |
|   (Ball starts here, need to align with archway) |
|     +-------+                                    |
|     | OBS 1 |  ZONE B: Pre-archway staging      |
|     | [   ] |  (Must be at archway height y~440)|
|     +-------+        +-----+                     |
|                      | OBS |   ZONE D: Target    |
|  ZONE C: Post-       |  2  |   approach area     |
|  archway transit     +-----+                     |
|                                                  |
|           +------------------+                   |
|           |     OBS 3        | ZONE E: DANGER!   |
|           +------------------+ (Robot arm limit) |
+--------------------------------------------------+

| Zone   | Condition           | Description                    |
|--------|---------------------|--------------------------------|
| ZONE A | y<400, x>300        | Initial area, above archway    |
| ZONE B | y in [400,480], x<350| Pre-archway, aligned with hole|
| ZONE C | x>350, y<600        | Post-archway transit area      |
| ZONE D | x>650, y<550        | Target approach area           |
| DANGER | y>550, x>520        | Robot arm limit exceeded!      |

**DIRECTION DEFINITIONS:**

              top (y decreases)
                    ^
                    |
   top-left    top-right
         \      /
          \    /
left  <----  O  ----> right  (x changes)
          /    \
         /      \
bottom-left  bottom-right
                    |
                    v
            bottom (y increases)

**SPEED TIERS:**

| Speed  | Velocity  | Typical Distance | Best Use Case
     |
|--------|-----------|------------------|----------------------------|

| low    | ~150 mm/s | ~50-80 pixels    | Fine positioning, near TGT
     |
| medium | ~300 mm/s | ~100-180 pixels  | Through archway, general
    nav|
| high   | ~450 mm/s | ~200-350 pixels  | Long distances, initial pos
     |
```

Ball Navigation: Stage-Specific Decision Rules

```
================================================================
NAVIGATION DECISION RULES BY STAGE
================================================================

**STAGE 1: ZONE A -> ZONE B (Initial to Pre-Archway)**

Distance to OBS1 archway determines speed:
- >200 pixels from archway: HIGH speed acceptable
- 100-200 pixels from archway: MEDIUM speed preferred
- <100 pixels from archway: LOW speed REQUIRED

Descent alignment rules:
- If ball above archway (y < 400): Must descend first
- PREFER diagonal descent (end_dir="bottom-left")
- AVOID straight down ("bottom") - ball overshoots!
- Goal: Position ball at y~400-480 aligned with archway

**STAGE 2: ZONE B -> ZONE C (Through Archway)**

Prerequisites:
- Ball MUST be at archway height (y in [400, 480])
- Use MEDIUM speed for passage
- Horizontal push (same y for start/end) maintains alignment

**STAGE 3: ZONE C (Post-Archway Transit)**

CRITICAL - Most common failure point:
- Use LOW speed immediately after archway!
- HIGH/MEDIUM speed causes ball to land on OBS3 surface
- If ball lands on OBS3: STUCK - no gripper access
- Goal: Position ball at y~500-550 (above OBS3)

**STAGE 4: ZONE D (Target Approach)**

Danger zone avoidance (y>550, x>520):
- NEVER push toward target from DANGER zone
- Robot arm exceeds motion limit
- MUST push UPWARD first (end_dir includes "top")

Target approach direction:
- PREFER approach from LEFT (push rightward)
- PREFER approach from BOTTOM (push upward)
- AVOID approach from RIGHT (arm collision risk)

Final approach:
- Within 100 pixels of target: LOW speed MANDATORY
- Endpoint offset: Set end 20-40 pixels BEYOND target center
  (compensates for ball momentum stopping early)

**SPEED SELECTION DECISION TREE:**

[Ball Location] --> [Speed Decision]

ZONE A (>200px from OBS1)     --> HIGH OK
ZONE A (100-200px from OBS1)  --> MEDIUM preferred
ZONE A (<100px from OBS1)     --> LOW required
ZONE B (at archway height)    --> MEDIUM for passage
ZONE C (post-archway)         --> LOW required!
Near OBS3 (y>550)             --> LOW, push UP first
ZONE D (>100px from target)   --> MEDIUM with caution
```

```
ZONE D (<100px from target)   --> LOW mandatory
```

G.3 BALANCED STACKING: DETAILED SPECIFICATIONS

The following specifications enable reasoning about stone selection using a systematic naming convention for surfaces, shapes, and orientations.

Balanced Stacking: Stone Naming Convention and Properties

```
================================================================
STONE NAMING SYSTEM AND PROPERTY REFERENCE
================================================================

**NAMING CONVENTION:**
Each stone is identified by: {Surface}_{Shape}_{Size}_{Orientation}

Example: 3M_HEX_L_C = 3M surface + Hexagonal + Large + Compact

**SURFACE TYPES (Friction Ranking: High to Low):**

| ID    | Surface              | Description              | Friction
     |
|-------|----------------------|--------------------------|-----------|

| 3M    | 3M Gripping Material | TB641 black velcro-like  | HIGHEST
     |
| BLK   | Black Duct Tape      | Industrial grade 9mil    | HIGH
     |
| WHT   | White PVC Tape       | Insulation tape          | MEDIUM
     |
| WOOD  | Rough Wooden         | Natural wood grain       | MED-LOW
     |
| PAINT | Painted Surface      | Smooth lacquer finish    | LOWEST
     |

Friction ordering for stacking: 3M > BLK > WHT > WOOD > PAINT
Higher friction surfaces should generally be placed LOWER.

**SHAPE TYPES (Cross-Section Geometry):**

| ID   | Shape      | Cross-Section | Stackability Notes
       |
|------|------------|---------------|-----------------------------------|

| HEX  | Hexagonal  | 6-sided       | Best for TOP layers (step 4-5)
       |
| SQR  | Square     | 4-sided flat  | Good base, should be below HEX
       |
| TRI  | Triangular | 3-sided       | Stable with flat edge down
       |
| DMND | Diamond    | 4-sided angle | Can be horizontal (H) or
   vertical |
| OVAL | Oval/Egg   | Rounded       | Challenging contact surface
       |
| PENT | Pentagon   | 5-sided       | Good intermediate layer
       |
| TREE | Tree/Branch| Irregular     | MUST be LAST (step 5) always
       |

**SIZE CATEGORIES:**

| ID | Size   | Approx Volume | Contact Area | Best Position    |
|----|--------|---------------|--------------|------------------|
| L  | Large  | >50 cm^3      | >15 cm^2     | BASE (step 1)    |
```

```
| M  | Medium | 20-50 cm^3    | 8-15 cm^2    | MIDDLE (step 2-3)|
| S  | Small  | <20 cm^3      | <8 cm^2      | TOP (step 4-5)   |

**ORIENTATION / ASPECT RATIO:**

| ID | Orientation | Aspect Ratio      | Description               |
|----|-------------|-------------------|---------------------------|
| H  | Horizontal  | width >> height   | Long bar lying flat       |
| V  | Vertical    | height >> width   | Long bar standing up      |
| C  | Compact     | width ~ height    | Roughly cubic/spherical   |

CRITICAL: Vertical bars (V) have very small contact area and
should NEVER be used as base stone. Only acceptable at steps 4-5.
```

Balanced Stacking: Friction and Shape Compatibility

```
================================================================
FRICTION COMPATIBILITY AND SHAPE STACKING RULES
================================================================

**FRICTION COMPATIBILITY MATRIX:**
Higher friction = better grip. Stack HIGH friction below LOW.

                    TOP STONE
              3M    BLK   WHT   WOOD  PAINT
         3M   ***   ***   **    **    **
BASE     BLK  ***   **    **    **    *
STONE    WHT  **    **    *     *     *
         WOOD **    **    *     **    *
         PAINT**    *     *     *     X

*** = Excellent grip | ** = Good | * = Fair | X = Poor (slip risk)

Key insight: Painted surfaces (PAINT) on painted surfaces create
high slip risk. Avoid stacking PAINT on PAINT at lower layers.

**SHAPE STACKING COMPATIBILITY:**

| Base Shape    | Can Support              | Cannot Support Well
    |
|---------------|--------------------------|-------------------------|

| SQR (Square)  | HEX, SQR, TRI, PENT      | TREE (unstable)         |
| HEX (Hexagon) | HEX, TREE, small SQR     | Large SQR (overhang)    |
| DMND_H (H-bar)| DMND_V, small compact    | SQR (incompatible!)     |
| DMND_V (V-bar)| Small compact, TREE      | Heavy stones            |
| TREE          | Nothing (always TOP)     | Everything              |

**CRITICAL INCOMPATIBILITY:**
SQR on DMND_H is FORBIDDEN - square cross-section cannot stack
on horizontal bar. Contact geometries fundamentally incompatible.
Observed 100% collapse rate in all attempts.

**SHAPE ORDERING RULES:**

1. SQR below HEX:
   Square cross-section should always be placed BEFORE hexagonal.
   Square provides flat platform for hexagonal contact.

2. DMND_H before DMND_V:
   Horizontal bars should be placed BEFORE vertical bars of same size
    .
   Horizontal provides wider stable platform for middle layers.
```

Balanced Stacking: Step-Specific Selection Rules

```
================================================================
STACKING DECISION RULES BY STEP
================================================================

**STEP 1 (BASE STONE) - CRITICAL REQUIREMENTS:**

ALL of the following MUST be satisfied:
[x] Select LARGEST available stone
[x] Select HIGHEST friction surface (3M > BLK > WHT > WOOD > PAINT)
[x] Select LARGEST contact surface area
[ ] NEVER select slender/vertical bar (V) orientation
[ ] NEVER select small stones

Base stone selection priority:
1. Any 3M surface, LARGEST size, NOT V-orientation
2. Any BLK surface, LARGEST size, NOT V-orientation
3. Any WOOD surface, LARGEST size, NOT V-orientation
4. LARGEST remaining stone, NOT V-orientation

**STEPS 2-3 (MIDDLE LAYERS):**

Preferences:
- PREFER: 3M Gripping or Black Tape surfaces
- PREFER: Square (SQR) cross-sections for stable platform
- AVOID: White-tape hexagonal (WHT_HEX) at Step 2 - collapse risk!
- AVOID: Vertical bars (V) - save for later steps

**STEPS 4-5 (UPPER LAYERS):**

Now acceptable:
- Hexagonal (HEX) stones belong HERE, not earlier
- Vertical bars (V) acceptable at these steps
- Small stones and light weight preferred

**STEP 5 (FINAL STONE) - MANDATORY RULE:**

If TREE-shaped stone exists: MUST place it at Step 5
- Tree cannot support any stone above due to irregular surface
- Always the final piece, no exceptions

**SAME SIZE/SHAPE FRICTION ORDERING:**

When two stones have IDENTICAL size and shape but different surfaces:
- Place WOODEN surface stone BELOW (lower layer)
- Place PAINTED surface stone ABOVE (upper layer)
- Rationale: Wood provides better friction than painted lacquer

**STACKING ORDER DECISION TREE:**

Step 1: LARGEST + HIGHEST friction + NOT V-orientation
Step 2: 3M/BLK surface preferred, avoid WHT_HEX
Step 3: Continue middle strategy, SQR for platform
Step 4: HEX acceptable, V-bar acceptable
Step 5: TREE (if exists), else smallest remaining
```

H COMPLETE PROMPT TEMPLATES

This section provides the complete prompt templates used in our real-world experiments. Each prompt follows our **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 during execution.

Incorporating Detailed Specifications. The prompts below can be augmented with the **detailed task specifications** from Section G. These specifications provide comprehensive reference materials including coordinate systems, naming conventions, and decision rules that significantly improve the VLM planner's task understanding. For each task, the corresponding specifications can be inserted after SECTION 1 (Task Description) or as a separate reference section. See:

- Parts Organization: Section G.1 (coordinate system, part properties)
- Ball Navigation: Section G.2 (obstacle layout, spatial zones)
- Balanced Stacking: Section G.3 (stone naming, stacking rules)

Below we present the base prompt template for each task.

H.1 BRICK INSERTION: COMPLETE PLANNING PROMPT (SIMULATION)

Complete prompt template for the Brick Assembly task (simulation).

```
============================================================
SECTION 1: TASK DESCRIPTION
============================================================

There is a puzzle consisting of a board and several pieces with
different colors on the table. The goal is to assemble the puzzle
with the robot arm.

In each step, one of the following four actions can be taken:
pick up [obj], put down [obj], reorient [obj], and insert [obj],
where [obj] refers to the piece to be manipulated.

The image of the goal state is: <image>.
The image of the current state is: <image>.
The most recently executed actions are: {action_history}

What action should be taken next?
Note that [obj] should be a color chosen from: {available_colors}

============================================================
SECTION 2: ACTION DEFINITIONS
============================================================

**pick up [color]**: Pick up a brick. Can pick up bricks from
the TABLE or bricks already INSERTED in the board.

**put down [color]**: Put down the brick you're holding ONTO
THE TABLE (not onto the board).
```

```
**reorient [color]**: Rotate the brick you're holding to align
with the board slot. WARNING: The resulting orientation may not
be perfect - after reorienting, check if the brick can be
correctly inserted. If not, you may need to reorient again.

**insert [color]**: Insert the brick you're holding INTO THE
BOARD. This will only succeed if: (1) you're holding the brick,
(2) it's properly oriented, and (3) the target slot is not
blocked by other bricks.

**done**: Declare the task complete. Use this ONLY when the
current observation matches the goal image.

---
## CRITICAL: BEFORE EVERY ACTION, FIRST CHECK IF DONE

**STEP 0 (MANDATORY)**: Compare current observation with goal:
- If current state MATCHES goal state, output `done` IMMEDIATELY
- Do NOT pick up, remove, or re-insert bricks after goal achieved
- Unnecessary actions after goal completion waste steps

============================================================
SECTION 3: STRATEGIC GUIDANCE
============================================================

Before each action, think through these questions:

1. **Check for blockers**: Are there bricks currently on the
   board that will BLOCK me from inserting the remaining bricks?

2. **If blockers exist**: Which brick should I REMOVE FIRST?
   Pick it up from the board and put it down on the table.

3. **If no blockers**: Which brick on the table should I INSERT
   FIRST to make progress toward the goal?

4. **Handle blocked insertions**: If insert fails because another
   brick is blocking the slot:
   - Put down the brick you're holding onto the table
   - Pick up the blocking brick from the board
   - Put down the blocking brick onto the table
   - Then decide your next move

============================================================
SECTION 4: CURRENT STATE
============================================================

{if_holding_object}
You are currently HOLDING: **{holding_color}**

IMPORTANT RULES:
- You CANNOT 'pick up' another object while holding {holding_color}
- You CAN ONLY: 'insert {holding_color}' or 'put down {holding_color
    }'
  or 'reorient {holding_color}'
{else}
Your gripper is EMPTY. You can 'pick up' any object.
{endif}

============================================================
SECTION 5: REPEATED ACTION WARNING
============================================================
```

```
{if_repeated_action}
WARNING: You have repeated '{repeated_action}' {repeat_count} times.

THIS IS NOT WORKING! You MUST try a DIFFERENT action:
- If reorienting: Try 'insert' directly or 'put down' and switch
- If inserting: Slot might be BLOCKED - remove blocker first
- Otherwise: Try a completely different action
{endif}

============================================================
SECTION 6: LONG-TERM MEMORY (Learned Principles)
============================================================

Apply these verified principles from past experience:

{formatted_principles}

Example format:
1. [92%] [SEQUENCE] Remove blocking bricks before attempting
   to insert dependent pieces.
2. [87%] [AVOID] Repeated reorient actions on the same brick -
   the issue is likely slot blocking, not orientation.
3. [85%] [PREFER] When insert fails, put down current brick
   and investigate which piece is blocking the slot.

============================================================
SECTION 7: WORKING MEMORY (Hypotheses Under Testing)
============================================================

Consider these hypotheses (not yet fully verified):

{formatted_hypotheses}

Example format:
1. [TESTING] Elongated blocks often need adjacent pieces
   removed before insertion is possible.
2. [TESTING] If a brick lands tilted, one reorient action
   is usually sufficient to correct orientation.

============================================================
SECTION 8: ACTION REQUEST
============================================================

Based on the goal image and current image, determine the next action.
Valid actions: pick up [color], put down [color], reorient [color],
insert [color], done

REQUIREMENTS:
1. Output exactly ONE action
2. You MUST provide an action - not providing one is NOT allowed
3. Your response MUST end with: Action: <your chosen action>

Examples:
- Action: pick up blue
- Action: insert red
- Action: reorient green
- Action: done

YOUR ACTION:
```

H.2 PARTS ORGANIZATION: COMPLETE PLANNING PROMPT

Complete prompt template for the Parts Organization task.

```
============================================================
SECTION 1: TASK DESCRIPTION
============================================================

You are a robot tasked with efficiently packing irregularly-shaped
3D-printed parts onto a grid workspace.

TASK: Place all 6 parts onto the 3x10 grid (30 cells total) while
minimizing the total number of occupied grid cells.

VISUAL INPUT:
- Image 1 (Wrist Camera): Close-up view of the current part to place
- Image 2 (Third-View Camera): Top-down view of the grid with:
  * Grid cells numbered 1-30 (3 rows x 10 columns)
  * Currently occupied cells highlighted in red
  * Available cells shown in green
- Image 3 (Part Template): Reference showing all 6 part types with
  their cell occupancy patterns at each rotation angle

CURRENT PART: {current_part_id} ({part_description})
PARTS REMAINING: {remaining_parts}
CELLS OCCUPIED: {occupied_cells} / 30

============================================================
SECTION 2: ACTION DEFINITIONS
============================================================

You must output ONE action specifying ALL cells the part will occupy.

IMPORTANT: All parts in this task occupy 2-4 cells (no single-cell
    parts).
You must specify the COMPLETE list of cells the part will cover.

**Action Format:**
  place({cell_1}, {cell_2}, ..., {cell_n})
  - Each cell_i is a grid cell index (1-30)
  - List ALL cells the part will occupy after placement
  - Cells must be adjacent and match the part's shape

**Grid Layout Reference:**
  Row 1: [1]  [2]  [3]  [4]  [5]  [6]  [7]  [8]  [9]  [10]
  Row 2: [11] [12] [13] [14] [15] [16] [17] [18] [19] [20]
  Row 3: [21] [22] [23] [24] [25] [26] [27] [28] [29] [30]

**Examples by Part Shape:**

  L-shaped part (3 cells):
    place(3, 13, 14)      - L pointing right-down
    place(4, 14, 15)      - L pointing right-down (shifted)
    place(11, 12, 2)      - L pointing right-up

  U-shaped part (4 cells):
    place(6, 7, 16, 17)   - U shape spanning 2x2
    place(1, 2, 11, 12)   - U shape in top-left corner

  I-shaped part (2 cells, elongated):
    place(5, 6)           - Horizontal bar
    place(3, 13)          - Vertical bar
```

```
  T-shaped part (3 cells):
    place(4, 5, 6, 15)    - T pointing down (actually 4 cells)
    place(2, 11, 12, 13)  - T pointing right

============================================================
SECTION 2.5: PART NAMING AND COORDINATE SYSTEM
============================================================

**Part Naming Convention:**
Each part is identified by {color}-{shape}:
- Colors: black, white, red
- Shapes: L-shape (3 cells), q-shape (3 cells), I-shape (2 cells), U-
    shape (4 cells)
- Parts: red-L, white-q, red-q, white-U, black-U, black-I

**Template-Based Coordinate System (2x2 Grid):**
Each part's local geometry uses a template grid:

    +---+---+
    | a | b |   <- top row [a, b]
    +---+---+
    | c | d |   <- bottom row [c, d]
    +---+---+

Default (0 degree) cell occupancy:
- L-shape: [a, c, d] - vertical bar on left, extension at bottom-
    right
- q-shape: [a, c, d] - similar to L but different internal geometry
- I-shape: [a, c]    - vertical bar on left column
- U-shape: [a, b, c, d] - all four cells

**Rotation Transformations (counterclockwise):**
- 90 deg:  a->c, b->a, c->d, d->b  (template becomes [[b,d],[a,c]])
- 180 deg: a->d, b->c, c->b, d->a  (template becomes [[d,c],[b,a]])
- 270 deg: a->b, b->d, c->a, d->c  (template becomes [[c,a],[d,b]])

**Part-Specific Internal Biases:**
- black-U / white-U: physical mass biased toward [b,d]; [a,c] has
    empty space
- white-q: left edge aligns with left boundary of [a,c]
- red-q: right edge aligns with right boundary of [a,c]
- black-I: top/right edges align with top-right of [a,c]

**Why This Matters:**
Use template regions [a,b,c,d] to describe spatial relationships:
- "The [c] region of white-q can overlap with [d] region of red-q"
- "When U-shape rotates 180 deg, its [a,c] can interlock with another
     U's [b,d]"

============================================================
SECTION 3: STRATEGIC GUIDANCE
============================================================

Before selecting a placement, consider these strategies:

1. **Analyze Part Shape**: Look at the current part's geometry.
   - How many cells does it occupy?
   - What rotation minimizes wasted space?

2. **Check Remaining Space**: Examine the grid for:
   - Contiguous empty regions that match the part shape
```

```
   - Corner and edge positions (often more efficient)
   - Gaps that could be filled by remaining parts

3. **Plan Ahead**: Consider future parts:
   - Will this placement block efficient positions for larger parts?
   - Are you leaving usable gaps or creating dead space?

4. **Rotation Strategy**:
   - L-shaped parts: Try rotations that fit into corners
   - Elongated parts: Align with grid edges when possible
   - Irregular parts: Test all 4 rotations mentally

5. **Packing Principles**:
   - Place larger/more constrained parts first
   - Fill corners and edges before center
   - Avoid creating isolated single-cell gaps

============================================================
SECTION 4: CURRENT STATE
============================================================

PART TO PLACE: {current_part_description}
  - Shape: {shape_type} (occupies {n_cells} cells)
  - Template Position: See Image 3, Part #{part_index}

GRID STATUS:
  - Occupied cells: {occupied_cell_list}
  - Available cells: {available_cell_list}
  - Largest contiguous region: {largest_region_size} cells

{if_multi_cell_part}
NOTE: This part requires {n_cells} adjacent cells. Verify that
your chosen position and rotation fit within the available space.
{endif}

============================================================
SECTION 5: REPEATED ACTION WARNING
============================================================

{if_repeated_action}
WARNING: You have attempted placement at position {last_position}
{repeat_count} times. Each attempt resulted in: {failure_reason}

REQUIRED: Choose a DIFFERENT position or rotation.
- If collision: The cells are occupied. Try another region.
- If out-of-bounds: The part extends beyond grid. Try different
    rotation.
- If invalid: Check the action format matches the examples.
{endif}

============================================================
SECTION 6: LONG-TERM MEMORY (Learned Principles)
============================================================

Apply these verified principles from past experience:

{formatted_principles}

Example format (using template coordinates [a,b,c,d]):
1. [92%] [PREFER] When placing white-U at 180 deg, its [a,c] region
   can interlock with black-U's [b,d] region for compact packing.
```

```
2. [88%] [AVOID] Complete overlap between [a,c] regions of two q-
    shaped
   parts - their internal geometries conflict and waste space.
3. [85%] [PREFER] Place the [c] region of white-q toward the grid's
   right side to leave room for I-shaped parts on the left.
4. [82%] [SEQUENCE] Place U-shaped parts before q-shaped parts;
   U-shapes constrain more cells and need strategic positioning.

============================================================
SECTION 7: WORKING MEMORY (Hypotheses Under Testing)
============================================================

Consider these hypotheses (not yet fully verified):

{formatted_hypotheses}

Example format (using template coordinates [a,b,c,d]):
1. [TESTING] When red-q is rotated 180 deg, its [a] region may
    overlap
   with white-q's [d] region without physical collision.
2. [TESTING] The [d] region of white-q can share grid cells with
   the [c] region of red-q due to complementary internal biases.
3. [TESTING] Black-I's internal bias (top-right of [a,c]) allows it
   to fit in narrow gaps left by U-shaped parts' [a,c] openings.

============================================================
SECTION 8: ACTION REQUEST
============================================================

Based on the images and information above, output your placement
    action.

REQUIREMENTS:
1. Output exactly ONE action
2. Format: place(cell_1, cell_2, ..., cell_n) listing ALL cells the
    part will occupy
3. Cells must be adjacent and match the part's shape from the
    template
4. Ensure no collision with already occupied cells
5. Ensure all cells are within grid boundaries (1-30)

EXAMPLES:
- place(3, 13, 14)        - L-shaped part occupying 3 cells
- place(6, 7, 16, 17)     - U-shaped part occupying 4 cells
- place(5, 6)             - I-shaped part occupying 2 cells

YOUR ACTION:
```

H.3 BALL NAVIGATION: COMPLETE PLANNING PROMPT

Complete prompt template for the Ball Navigation task.

```
============================================================
SECTION 1: TASK DESCRIPTION
============================================================

You are a robot tasked with pushing a soccer ball toy through an
obstacle course to reach a target hole.

TASK: Push the ball from its current position to the target region
within 6 steps maximum. The ball must pass through THREE STAGES:
  1. Pass through Obstacle 1's "bridge hole" (archway)
  2. Navigate around Obstacles 2 and 3 toward target area
  3. Fine-tune position and reach the final target

OBSTACLE LAYOUT (fixed for all episodes):
  Obstacle 1 (BLUE): Archway block on LEFT side
    - Position: y~200-550, x~100-300
    - Has "bridge hole" at y~400-480 (ball MUST pass through here)
    - Ball cannot go over/around - MUST go through archway

  Obstacle 2 (RED): Rectangular cuboid in MIDDLE
    - Position: y~200-500, x~450-550
    - Solid blocker - ball bounces off or goes around

  Obstacle 3 (PURPLE): Rectangular cuboid at BOTTOM
    - Position: y~650-850, x~280-520
    - DANGER ZONE: If ball lands on top (y~600-700), gripper
      cannot access ball from below

  Target (WHITE circle with cross): RIGHT side
    - Position: y~350-450, x~720-800
    - Ball must stop within this region

SPATIAL ZONES:
  ZONE A (y<400, x>300): Initial area, above archway level
  ZONE B (y~400-480, x<350): Pre-archway staging area
  ZONE C (x>350, y<600): Post-archway transit area
  ZONE D (x>650, y<550): Target approach area
  DANGER (y>550, x>520): Robot arm limit zone near OBS3

VISUAL INPUT:
- Image 1 (Wrist Camera): Close-up view of the ball and nearby
  obstacles for detailed local context
- Image 2 (Top-Down Camera): Bird's-eye view showing:
  * Current ball position relative to obstacles
  * Target region (goal area with cross pattern)
  * Obstacle colors: BLUE(OBS1), RED(OBS2), PURPLE(OBS3)

CURRENT STEP: {current_step} / 6
BALL POSITION: Approximately ({ball_y}, {ball_x}) in normalized (y,x)
     coordinates
TARGET POSITION: Approximately ({target_y}, {target_x})
CURRENT ZONE: {zone_name}

============================================================
SECTION 2: ACTION DEFINITIONS
============================================================

You must output ONE push action in the following format:
```

```
  push({start_direction}, {end_direction}, {start_y}, {start_x}, {
    end_y}, {end_x}, {speed})

Parameters:
  - start_direction: Direction from ball center to gripper START
    position
    Options: "left", "right", "top", "bottom",
             "top-left", "top-right", "bottom-left", "bottom-right"
  - end_direction: Direction from ball center to gripper END position
    Options: same 8 directions as start_direction
  - start_y, start_x: Pixel coordinates where gripper starts the push
    (NOTE: y comes before x to match Gemini visual grounding
    convention)
  - end_y, end_x: Pixel coordinates defining push endpoint
  - speed: Push velocity level in {"low", "medium", "high"}

COORDINATE SYSTEM:
  Pixel values use a NORMALIZED 0-1000 range:
  - Top-left corner is (0, 0)
  - Bottom-right corner is (1000, 1000)
  - This is independent of actual image resolution

DIRECTION SPECIFICATION:
  The start_direction and end_direction explicitly encode the
  positional relationship between gripper points and the ball:
  - "left": Gripper is to the LEFT of the ball center
  - "right": Gripper is to the RIGHT of the ball center
  - "top": Gripper is ABOVE the ball center
  - "bottom": Gripper is BELOW the ball center
  - "top-left", "top-right", "bottom-left", "bottom-right": Diagonal

  These directions help you reason about the push geometry.

ENDPOINT CONSTRAINT (IMPORTANT):
  The end point MUST be within a circle of radius ~150mm centered
  at the start point. If your planned push is too long, the system
  will clip the end point to this constraint. Plan accordingly!

SPEED GUIDELINES:
  - "low" (150 mm/s): Short, precise pushes
    Best for: Fine adjustments near target, tight spaces
  - "medium" (300 mm/s): Moderate pushes
    Best for: General navigation, passing through obstacles
  - "high" (450 mm/s): Long, powerful pushes
    Best for: Long straight paths with no obstacles

PUSH EXAMPLES:
  push("left", "right", 300, 150, 300, 350, "medium")
    - Gripper starts on LEFT of ball, pushes RIGHT
    - Ball moves rightward at medium speed

  push("bottom", "top", 400, 250, 200, 250, "high")
    - Gripper starts BELOW ball, pushes UPWARD
    - Ball moves upward at high speed

  push("bottom-left", "top-right", 350, 100, 200, 300, "medium")
    - Gripper starts at BOTTOM-LEFT of ball
    - Pushes diagonally toward TOP-RIGHT

============================================================
SECTION 3: STRATEGIC GUIDANCE
```

```
============================================================

**STAGE-SPECIFIC STRATEGIES:**

1. **STAGE 1 - Reaching OBS1 Archway (ZONE A → ZONE B)**:
   - If ball is ABOVE archway level (y < 400): Must descend first
   - Use LOW speed for descent to avoid overshooting archway height
   - PREFER diagonal descent (end_dir="bottom-left") over straight
    down
   - Straight "bottom" push often overshoots - ball goes too low
   - Goal: Position ball at y~400-480, aligned with archway opening

2. **STAGE 2 - Passing Through Archway (ZONE B → ZONE C)**:
   - Ball MUST be at archway height (y~400-480) before attempting
    pass
   - Use MEDIUM speed to push through archway (end_dir="right")
   - Horizontal push maintains archway alignment during transit

3. **STAGE 3 - Post-Archway Transit (ZONE C)**:
   - CRITICAL: Use LOW speed immediately after archway!
   - HIGH/MEDIUM speed causes ball to roll onto OBS3 top surface
   - If ball lands on OBS3 (y~600-700): No gripper access, STUCK!
   - Goal: Keep ball at y~500-550 (above OBS3 danger zone)

4. **STAGE 4 - Target Approach (ZONE D)**:
   - AVOID the DANGER ZONE (y>550, x>520) - robot arm limit!
   - If ball is near OBS3: Push UPWARD first (end_dir includes "top")
   - Approach target from LEFT or BOTTOM (safer for robot arm)
   - Use LOW speed when within 100 pixels of target
   - Set endpoint 20-40 pixels BEYOND target (momentum compensation)

**SPEED SELECTION BY DISTANCE TO OBS1:**
- Ball >200 pixels from archway: HIGH speed OK
- Ball 100-200 pixels from archway: MEDIUM speed preferred
- Ball <100 pixels from archway: LOW speed REQUIRED

**COMMON FATAL MISTAKES:**
- Using HIGH speed when close to OBS1 → ball overshoots to OBS3
- Pushing straight down from initial position → ball overshoots
    archway level
- Using MEDIUM/HIGH after archway → ball lands on OBS3 top
- Pushing toward target from DANGER zone → robot arm collision
- Using MEDIUM speed in final approach → overshoots target

============================================================
SECTION 4: CURRENT STATE
============================================================

STEP: {current_step} of 6 maximum
REMAINING STEPS: {remaining_steps}

BALL STATUS:
  - Position: ({ball_x}, {ball_y}) pixels
  - Distance to target: approximately {distance_to_target} pixels

OBSTACLES DETECTED:
{obstacle_list}
  Example:
  - Obstacle 1: Rectangle at (150, 200) to (180, 280)
  - Obstacle 2: Circle centered at (300, 250), radius ~30 pixels

{if_last_push_info}
```

```
LAST PUSH RESULT:
  - Action: push({last_start}, {last_end}, {last_speed})
  - Ball moved from ({prev_pos}) to ({curr_pos})
  - Outcome: {push_outcome}
{endif}

============================================================
SECTION 5: REPEATED ACTION WARNING
============================================================

{if_stuck_pattern}
WARNING: The ball has not made progress toward the target for
{stuck_count} consecutive pushes.

Analysis of stuck pattern:
- Ball oscillating between positions: {oscillation_positions}
- Likely cause: {stuck_cause}

REQUIRED: Try a fundamentally different approach:
- If hitting obstacle: Push at an angle to go AROUND it
- If overshooting: Reduce speed level
- If undershooting: Increase speed or push distance
{endif}

============================================================
SECTION 6: LONG-TERM MEMORY (Learned Principles)
============================================================

Apply these verified principles from past experience:

{formatted_principles}

Example format (using obstacle IDs and spatial zones):
1. [95%] [SPEED] Distance-based speed for OBS1 approach:
   - Ball >200px from archway: HIGH speed acceptable
   - Ball 100-200px from archway: MEDIUM speed preferred
   - Ball <100px from archway: LOW speed REQUIRED
2. [92%] [CONDITION] Archway height prerequisite:
   Ball MUST be at y~400-480 before attempting OBS1 archway pass.
   If above (y<400), use LOW speed with end_dir="bottom-left".
3. [90%] [SPEED] Post-archway LOW speed requirement:
   Immediately after OBS1, MUST use LOW speed. MEDIUM/HIGH causes
   ball to land on OBS3 top surface - creates stuck position.
4. [88%] [AVOID] OBS3 danger zone (y>550, x>520):
   Never push toward target from this zone. Robot arm exceeds
   motion limit. MUST push upward (end_dir includes "top") first.

============================================================
SECTION 7: WORKING MEMORY (Hypotheses Under Testing)
============================================================

Consider these hypotheses (not yet fully verified):

{formatted_hypotheses}

Example format (using obstacle IDs and spatial zones):
1. [TESTING] 45-degree descent from ZONE A:
   When descending to archway level, diagonal push ("bottom-left")
   is more controllable than straight "bottom". Ball overshoots
   less with diagonal push due to horizontal velocity component.
2. [TESTING] Target approach direction preference:
   Approaching target from LEFT (push rightward) or BOTTOM
```

```
   (push upward) is safer than from RIGHT - avoids arm collision.
3. [TESTING] Endpoint offset compensation:
   Final push should set end coordinates 20-40px BEYOND target
   center (in push direction) to account for early momentum decay.

============================================================
SECTION 8: ACTION REQUEST
============================================================

Based on the images and information above, output your push action.

REQUIREMENTS:
1. Output exactly ONE push action
2. Format: push(start_direction, end_direction, start_y, start_x,
    end_y, end_x, speed)
3. start_direction and end_direction must be one of:
   "left", "right", "top", "bottom",
   "top-left", "top-right", "bottom-left", "bottom-right"
4. Coordinates use NORMALIZED (y, x) order in range 0-1000
   - (0, 0) is top-left, (1000, 1000) is bottom-right
5. Speed must be one of: "low", "medium", "high"
6. End point must be within ~150mm radius of start point

SCORING REMINDER:
  +1: Ball successfully moves
  +3: Ball passes through obstacle's bridge hole
  +5: Ball reaches target
  +2*(6-i): Early completion bonus at step i
  -2: Collision with obstacle OR invalid plan

EXAMPLES:
- push("left", "right", 300, 150, 300, 350, "medium")
- push("bottom", "top", 400, 250, 200, 250, "high")
- push("top-left", "bottom-right", 150, 100, 350, 400, "low")

YOUR ACTION:
```

H.4 BALANCED STACKING: COMPLETE PLANNING PROMPT

Complete prompt template for the Balanced Stacking task.

```
============================================================
SECTION 1: TASK DESCRIPTION
============================================================

You are a robot tasked with building a stable tower by stacking
irregularly-shaped balance stones with varying friction surfaces.

TASK: Stack all 5 balance stones into a stable tower. The tower
must remain standing without collapse after the final stone is placed
    .

STONE SURFACE TYPES (Friction: High to Low):
  1. 3M Gripping (black velcro-like) - HIGHEST friction
  2. Black Duct Tape - HIGH friction
  3. White PVC Tape - MEDIUM friction
  4. Rough Wooden - MEDIUM-LOW friction
  5. Painted (colorful) - LOWEST friction

STONE SHAPE TYPES (Cross-Section):
  - SQR (Square): Flat 4-sided, good contact
  - HEX (Hexagonal): 6-sided, smaller contact area
  - DMND (Diamond): Can be horizontal (H) or vertical (V) bar
  - PENT (Pentagon): 5-sided irregular
  - OVAL (Egg): Rounded, challenging contact
  - TREE (Branch): Irregular - MUST be placed LAST

STONE PROPERTIES TO CONSIDER:
  - Contact Area: Larger = more stable base
  - Aspect Ratio: Vertical bars (V) have small contact
  - Friction Surface: High friction better for lower layers

VISUAL INPUT:
- Image 1 (Wrist Camera): Close-up view showing:
  * Detailed texture of stone surface (friction material)
  * Current tower top surface and contact geometry
  * Stone shape and cross-section details
- Image 2 (Top-Down Camera): Overhead view showing:
  * All available stones with visible surface materials
  * Current tower state (if stones already stacked)
  * Stone positions for coordinate selection
- Image 3 (Base/Side Camera): Side-angle view showing:
  * Tower height and structure
  * Stone orientations and aspect ratios (H/V/C)
  * Contact points and stability indicators

STONES AVAILABLE: {available_stones}
STONES STACKED: {stacked_count} / 5
CURRENT TOWER HEIGHT: {tower_height} layers

============================================================
SECTION 2: ACTION DEFINITIONS
============================================================

You must output ONE action to select which stone to stack next:

  stack({point_y}, {point_x})

Parameters:
```

```
  - point_y, point_x: Pixel coordinates pointing to the CENTER of
    the stone you want to pick up and stack
    (NOTE: y comes before x to match Gemini visual grounding
    convention)

The robot will:
1. Use SAM segmentation to identify the stone at that point
2. Calculate optimal grasp pose based on stone geometry
3. Pick up the stone and place it on top of the current tower
4. Use adaptive height control for safe placement

IMPORTANT: Point to the stone's CENTER, not its edge.
Remember: Coordinates are (y, x) order, not (x, y).

============================================================
SECTION 3: STRATEGIC GUIDANCE
============================================================

**STEP-SPECIFIC RULES:**

STEP 1 (Base Stone) - CRITICAL REQUIREMENTS:
  - MUST select: LARGEST stone available
  - MUST select: HIGHEST friction surface (3M > BLK > WHT > WOOD >
    PAINT)
  - MUST select: LARGEST contact surface area
  - FORBIDDEN: Slender/vertical bar (V) orientation
  - FORBIDDEN: Small stones

STEPS 2-3 (Middle Layers):
  - PREFER: 3M Gripping or Black Tape surfaces
  - PREFER: Square (SQR) cross-sections for stable platform
  - AVOID: White-tape hexagonal (WHT_HEX) at Step 2 - causes collapse
  - AVOID: Vertical bars (V) - save for later

STEPS 4-5 (Upper Layers):
  - Hexagonal (HEX) stones should be placed HERE, not earlier
  - Vertical bars (V) acceptable at these steps
  - Small stones and light weight are preferred

STEP 5 (Final Stone) - MANDATORY:
  - Tree-shaped (TREE) stones MUST be placed LAST
  - Tree cannot support any stone above it

**SHAPE COMPATIBILITY RULES:**

  SQR below HEX: Square cross-section should be placed BELOW
    hexagonal
  DMND_H before DMND_V: Horizontal bars before vertical bars of same
    shape
  SQR on DMND_H: FORBIDDEN - square cannot stack on horizontal bar
  Same shape different surface: WOOD below, PAINT above

**SURFACE FRICTION ORDERING:**

  Lower layers: HIGH friction (3M, Black tape)
  Upper layers: Can use lower friction (White tape, Wood, Paint)
  Same size/shape: Wood below, Painted above

============================================================
SECTION 4: CURRENT STATE
============================================================
```

```
TOWER STATUS:
  - Stones stacked: {stacked_count} / 5
  - Current height: {tower_height} layers
  - Stability assessment: {stability_status}

STACKED STONES (bottom to top):
{stacked_stone_list}
  Example:
  1. [BASE] Large flat gray stone - stable foundation
  2. [LAYER 2] Medium rough brown stone - good friction
  3. [LAYER 3] Small angular white stone - slight overhang left

AVAILABLE STONES:
{available_stone_list}
  Example:
  - Stone A: Small, smooth, oval-shaped (at y~200, x~150)
  - Stone B: Medium, rough, triangular (at y~180, x~300)

TOP SURFACE ANALYSIS:
  - Current top stone: {top_stone_description}
  - Flat region available: approximately {flat_area} cm^2
  - Recommended placement zone: center of current top

============================================================
SECTION 5: REPEATED ACTION WARNING
============================================================

{if_previous_collapse}
WARNING: The previous stacking attempt caused {n_fallen} stones
to fall from the tower!

Collapse analysis:
- Stone attempted: {failed_stone}
- Likely cause: {collapse_cause}
- Stones that fell: {fallen_stones}

PENALTY INCURRED: -{penalty_points} points

REQUIRED: Choose a DIFFERENT, more stable option:
- If top-heavy: Select a lighter/smaller stone
- If poor contact: Select a stone with flatter surface
- If center of gravity issue: Select a stone that balances better
{endif}

============================================================
SECTION 6: LONG-TERM MEMORY (Learned Principles)
============================================================

Apply these verified principles from past experience:

{formatted_principles}

Example format (using surface/shape/orientation naming):
1. [95%] [REQUIRE] Step 1 base: Select LARGEST stone with HIGHEST
   friction (3M > BLK > WHT > WOOD > PAINT), LARGEST contact area.
   NEVER select vertical bar (V) orientation for base.
2. [92%] [AVOID] Vertical bar (V) as base stone:
   Small contact area (~3-5cm2) causes immediate collapse.
   V-orientation only acceptable at Steps 4-5.
3. [90%] [SEQUENCE] HEX cross-section stones at Steps 4-5:
   Hexagonal contact area too small for lower layers.
   Place after Step 3 (toward top).
```

```
4. [88%] [REQUIRE] TREE shape MUST be Step 5 (last):
   Cannot support any stone above due to irregular surface.
5. [85%] [AVOID] SQR on DMND_H (horizontal bar):
   Square cannot stack on horizontal bar - 100% collapse rate.

============================================================
SECTION 7: WORKING MEMORY (Hypotheses Under Testing)
============================================================

Consider these hypotheses (not yet fully verified):

{formatted_hypotheses}

Example format (using surface/shape/orientation naming):
1. [TESTING] White-tape hexagonal (WHT_HEX) at Step 2:
   Causes collapse due to insufficient friction. White tape on
   hexagonal contact too slippery for early layers.
2. [TESTING] Same shape different surface ordering:
   When two stones have identical shape, place WOOD below PAINT.
   Wood provides better friction than painted for lower layers.
3. [TESTING] Horizontal before vertical bar ordering:
   DMND_H should be placed before DMND_V of same size.
   Horizontal bars provide wider platform for middle layers.

============================================================
SECTION 8: ACTION REQUEST
============================================================

Based on the images and information above, select the next stone to
    stack.

REQUIREMENTS:
1. Output exactly ONE stack action
2. Format: stack(point_y, point_x)
   (NOTE: y coordinate comes BEFORE x coordinate)
3. Point coordinates must indicate the CENTER of your chosen stone
4. Consider: Will this stone create a STABLE addition to the tower?

EXAMPLES:
- stack(200, 150)    - Select stone at y=200, x=150
- stack(300, 180)    - Select stone at y=300, x=180

SCORING REMINDER:
- Successfully placing stone on layer i: +i points
- Causing j stones to fall: -2j points
- Goal: Maximize total score by building stable 5-stone tower

YOUR ACTION:
```

H.5 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.
```

H.6 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.
```

REFERENCES

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.

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gober, 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)*, pp. 287–318, 2022.

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.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Zesen Cheng, Lianghao 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.

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*, pp. 7318–7328, 2025.

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 Buttimore, 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 Gharamani, Raia Hadsell, Aäron van den Oord, Inbar Mosseri, Adrian Bolton, Satinder Singh, and Tim Rocktäschel. Genie 3: A new frontier for world models. 2025.

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.

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*.

Charles Blundell, Benigno Uria, 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.

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.

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.

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.

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.

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 Büchler, 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, João Silvério, Joey Hejna, Jonathan Booher, 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, Minho 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 Sundaresan, Qiuyu Chen, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Mart'in-Mart'in, Rohan Baijal, 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.

Google DeepMind. Gemini 3 flash: Frontier intelligence built for speed, 2025a. URL https://blog.google/products-and-platforms/products/gemini/gemini-3-flash/. Accessed: 2026-01-21.

Google DeepMind. Gemini embodied reasoning 1.5: Multi-step reasoning for robotic planning, 2025b. URL https://deepmind.google/technologies/gemini/. Accessed: 2026-01-21.

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.

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.

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)*, pp. 538–547, 2019.

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.

Figure AI Team. Helix: A vision-language-action model for generalist humanoid control. *Technical Report*, 2025.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning (ICML)*, pp. 1126–1135, 2017a.

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)*, pp. 357–368, 2017b.

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.

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.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

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.

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)*, pp. 1769–1782, 2022.

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.

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.

Sungjune 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.

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.

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.

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, pp. 9459–9474, 2020.

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)*, pp. 17913–17922, 2024a.

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.

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*, 2024b.

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)*, pp. 9493–9500, 2023.

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.

Jiaxun Liu and Boyuan Chen. SonicSense: Object perception from in-hand acoustic vibration. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2024.

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.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, 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.

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.

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.

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.

OpenAI. GPT-5.1: Advanced multimodal reasoning model, 2025. URL https://openai.com/gpt-5.

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.

Karl Popper. *The Logic of Scientific Discovery*. Routledge, 1959.

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)*, pp. 2827–2836, 2017.

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.

Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. Memoryvla: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2508.19236*, 2025.

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.

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.

SIMA Team, Adrian Bolton, Alexander Lerchner, et al. SIMA 2: A generalist embodied agent for virtual worlds. *arXiv preprint arXiv:2512.04797*, 2025.

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.

Venkatesh Sripada, Samuel Carter, Frank Guerin, and Amir Ghalamzan. AP-VLM: Active perception enabled by vision-language models. *arXiv preprint arXiv:2409.17641*, 2024.

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)*, pp. 9229–9248. PMLR, 2020.

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.

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.

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)*, pp. 23–30, 2017.

Shoukai Xu, Mingkui 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*.

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.

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.

Jesse Zhang, Minho 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.

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.

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*, pp. 1702–1713, 2025a.

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)*, 2025b.