 DON'T DROP THE **BATON**: LONG-HORIZON ROBOT MANIPULATION VIA AGENTIC SUBTASK EXPLORATION AND TRANSITION-AWARE MEMORY

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-horizon robot manipulation chains many contact-rich skills into one multi-stage task. Vision-language-action (VLA) and world-action models (WAMs) increasingly master the individual skills. However, the chain itself still fails: errors compound beyond the policy's ability to correct, and the execution of one subtask silently constrains the next. Coding agents offer a promising pathway: freeze the VLA and put an LLM agent in charge of it. The agent plans in language, moves in free space with analytic primitives, and invokes the VLA only for the contact-rich segments, writing all adaptation into language memory. Yet applied directly to long horizons, the agent-plus-VLA recipe breaks twice. ❶ Its competence is acquired through whole-task exploration at test time, whose cost is exponential in the number of stages: where a single stage requires T episodes on average, a K-stage task requires on the order of T^K, and a failed episode does not identify the stage responsible. ❷ It has no representation of the transitions: the VLA primitive carries an exit condition but no entry condition, and a subtask can succeed in a form its successor cannot use. We present **BATON**, which addresses the two failures in turn.[1] Against ❶, **BATON** makes the subtask the unit of exploration. Each subtask is explored in the inexpensive short-horizon regime, and its solution is stored in memory. A long-horizon trajectory is then composed from these solutions rather than discovered whole. Exploration cost thus becomes linear in the number of stages (KT), and every failure is attributed to a single stage. Against ❷, **BATON** equips this exploration with a transition-aware memory. Within a subtask, a verifier agent governs the *invocation transition*: the VLA is invoked only after the wrist view confirms that the scene is ready. Across subtasks, *handoff transition* restores an entry state disturbed by the predecessor's residue, and *lookahead transition* determines the execution strategy whose outcome the successor can inherit. On the long-horizon manipulation benchmark RoboMemArena, **BATON** improves task success rate by 37.7% and cumulative success rate by 29.7% over the current SoTA.

1 INTRODUCTION

What separates current manipulation policies from generalist ones is less about any single skill than about composition: interpreting an instruction, then holding it together across many contact-rich steps whose errors compound. The skills themselves are increasingly within reach: vision-language-action (VLA) models and world-action models (WAMs) map observations and language directly to low-level actions, generalizing broadly through web-scale pretraining [3, 15, 6, 2, 29, 1, 33, 24, 39]. What none of them yet delivers is the chain. Set $\pi_{0.5}$ [29] on a single skill (open a drawer, pick up a can, place it in the microwave) and it executes cleanly, at 97% success on LIBERO's single-task suites [28]; set it on the thousand-step tasks that chain such skills and it clears 38.7% of the intermediate stages while finishing only 21.5% of the full tasks [18]. The stages succeed, and the task dies between them.

Two things go wrong. First, errors compound and the policy cannot repair them. End-to-end VLAs overfit the demonstrated trajectories from their curated initial states [8, 9]. When each stage starts

[1] The recipe needs this interface, not a particular backend: a VLA, a WAM, or any contact-rich policy callable the same way, slots in unchanged. We write VLA throughout for simplicity.

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

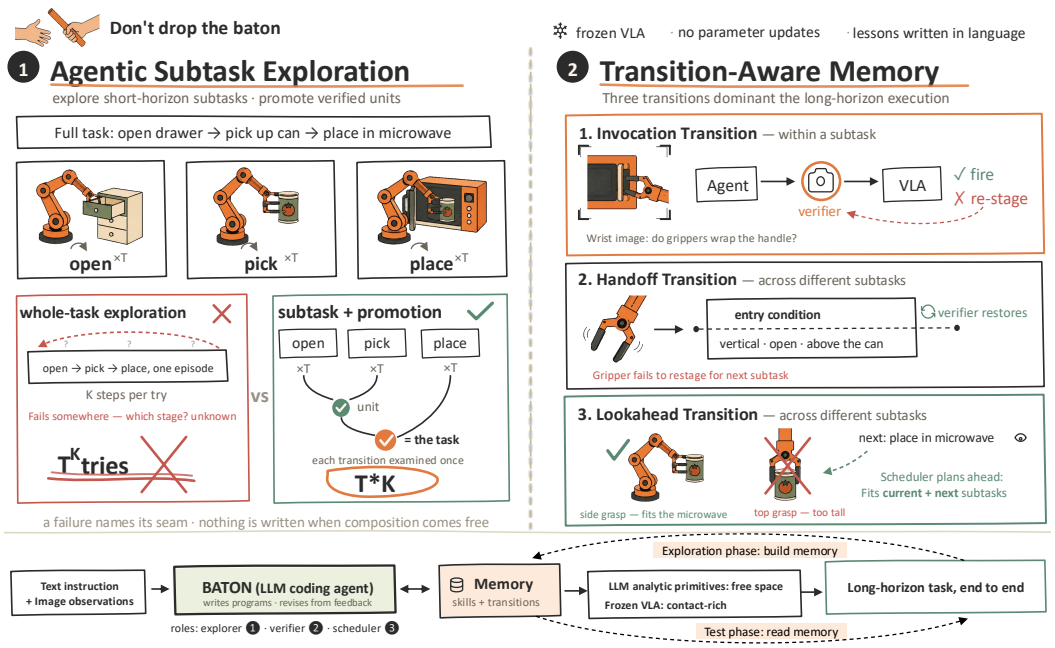

Figure 1: **BATON treats transitions as first-class objects.** *Bottom*: overall workflow (Sec. 3.1). *Left*: subtask exploration makes the cost linear in the number of stages and localizes each failure (Sec. 3.2). *Right*: the three proposed transition memories (Sec. 3.3).

wherever the previous one ended, it can only replay memorized motions from states it has never seen, and it can neither recognize nor correct the accumulating drift [30, 45]. Second, neighboring subtasks are coupled through execution: how one subtask is carried out fixes the state the next begins from, and an execution that succeeds locally can leave the next subtask impossible [17, 42, 10].

Coding agents are built for exactly this kind of composition [41, 40, 34]. In this line of work [20, 11], an LLM agent controls the robot by writing programs over perception and control primitives. The programs run, and the agent revises them from execution feedback. Nothing is trained; the agent improves by trying things at test time. Harness VLA [44] brings this recipe to contact-rich manipulation by adding a frozen VLA to the agent's toolbox. The agent plans in language and moves through free space with analytic, inverse-kinematics-solved primitives; where the task turns contact-rich, it hands over to the VLA, wrapped as a retryable primitive with a prompt and a stop predicate. For long-horizon control, this division of labor is attractive on three counts. First, each part does what it is best at: the VLA handles local contact, and the agent handles composition, semantic re-binding, and staging. Second, nothing is fine-tuned. One frozen policy serves every task. Third, everything the system learns is written in language, so it can be read, audited, and reused. On short-horizon tasks, this design already works: skills stay reliable under perturbations that break end-to-end VLAs [44]. Long-horizon manipulation is another matter: no coding agent yet earns the same competence over a thousand-step chain.

Two difficulties block the way. ❶ is the large cost of full task exploration. The agent earns its competence at test time. During exploration, the agent freely trials staging orders, pre-contact poses, VLA invocation timings, and termination thresholds until the task first succeeds, then consolidates the working solution into memory [44]. For a short-horizon task of a hundred-odd steps, first success arrives after T exploration episodes on average, with a matching bill of planner tokens: a real price, but one paid once per task. Stretching the same procedure over a task of K such stages breaks it. An episode succeeds only if every stage does, so whole-task exploration needs on the order of T^K episodes, beyond any budget (see Fig. 1, left). To make matters worse, the failures are uninformative: an episode reveals only that the task failed, not which stage broke. ❷ is the transitions the chain must cross. Beyond its cost, exploration is uninformed precisely where a chain is fragile: at the crossings. Within a subtask, no signal indicates when the scene is ready for the contact-rich policy, so the moment of handover must be found by repeated trial. Across subtasks, individually successful skills fail to compose, for two reasons. First, a completed subtask alters the state its successor starts from. A subsequent failure is therefore ambiguous: the successor's skill may be inadequate, or its

entry state is merely wrong, however exploration cannot distinguish the two. Second, even a correct entry is not sufficient: subtasks are interdependent, and the requirements of the next constrain how the previous one should be executed. This dependency spans subtasks, so a learner that acquires them one at a time can neither observe it nor correct for it.

Targeting ❶, we propose **BATON**, an LLM agent orchestrating a frozen VLA for long-horizon robot manipulation tasks. **BATON** makes the subtask the unit of exploration: it decomposes a long-horizon task and explores each subtask in the cheap short-horizon regime where the recipe can work. Each solved subtask's successful execution is extracted into abstract memory. A long-horizon trajectory is then composed from these pieces by memory rather than discovered whole. Exploration cost now scales linearly in the number of new subtasks and boundaries rather than exponentially (see Fig. 1, left). A failure names the stage that caused it instead of voiding a thousand-step episode. A relay team does not train by running ten thousand full races; it drills each exchange.

BATON meets ❷ with a *transition-aware* memory that consists of three parts. *Within* a subtask lies the *invocation transition*, where control passes from the agent to the contact-rich VLA after a *verifier agent* confirms. *Across* subtasks, two transitions govern each boundary. The *handoff transition* concerns the entry state of the successor: the boundary records an entry condition, and the verifier restores the scene to satisfy it before the next subtask begins. The *lookahead transition* operates in the reverse direction: the subtask requirements that follows determine how the current one is executed. A *scheduler agent* enforces this dependency, selecting from memory the strategy whose outcome the successor can inherit. Together these conditions turn each boundary into a *handoff contract* that is drafted in language, checked at execution, and stored with the coarse trajectories it connects. A good decomposition cuts a long task at interfaces that can be checked. Our core contributions are:

- We present **BATON**, the first coding agent to carry long-horizon robot manipulation end to end over a frozen VLA, learning by test-time exploration with no parameter updates.
- **BATON** rests on two mechanisms: *hierarchical subtask exploration* for ❶, which keeps exploration cost linear in the number of stages; *transition-aware memory* for ❷, which attaches checkable conditions to the invocation, handoff, and lookahead transitions.
- On RoboMemArena's thousand-step household tasks, **BATON** improves task success by 37.7% and cumulative success by 29.7% over the current SoTA.

2 Related Work

Long-horizon robot manipulation. The dominant response to long horizons is to train for them. Web-scale VLA pretraining keeps improving the individual skills [3, 15, 2, 29], but the resulting policies are reactive: they predict the next action from the current observation, overfit the curated initial states of their demonstrations [8, 9], and cannot correct their own compounding drift [30, 45]. Memory-augmented VLAs train history into the policy [32, 35], and memory-oriented benchmarks measure what remains: on RoboMemArena's thousand-step household tasks, every reported policy completes far more stages than full tasks [18, 7]. That skills fail when chained where each succeeds alone is documented across settings. Sequenced policies need terminal states aligned with what the next policy expects [17]; on BOSS, observation-space shift at subtask boundaries alone cuts the success of four baseline policies by 34–67%, a drop that thirty times more demonstrations do not remove [42]; VLA skills with isolated success rates of 77–100% still collapse under composition [31]; and nine grasping policies with near-identical first-subtask success (about 95%) spread from 0% to 78% on the subtask that follows [10]. The fixes, again, train for the boundary: HANDFUL selects among finger configurations by curriculum-based successive elimination [10], and Foresight Residual RL shapes terminal states with a residual policy rewarded by a learned downstream-success predictor [23]. Both bind the lesson to a fixed successor policy in a known sequence and pay a per-boundary training cost, so every new pairing of skills pays again. **BATON** encodes the same insight symbolically, in the manner of a coding agent. The handoff contract is drafted in language at plan time, checked at the boundary, and corrected when it fails. Because it attaches to the transition rather than to the task, it is paid for once and reused by every task that crosses the same boundary.

Memory for manipulation agents. Skill libraries since Voyager converge to one program per skill name [40]. Embodied-memory systems for manipulation key retrieval on the current instruction, recent observations, or execution history [38, 13]. Memory-augmented VLAs build episodic context

into the policy itself [32] or retrieve past keyframes from an experience buffer [35]; in either case the memory records what was seen, not which strategy was chosen. HiMe is the sharpest foil. Its hierarchy indexes control frequency, it deletes conflicting entries to keep memory consistent, and its subtask monitor records *that* a grasp succeeded rather than *which* grasp [13]. The rival strategies we preserve are, in that design, a defect to remove; yet which grasp was taken is exactly what the next subtask depends on. Task-oriented grasping selects a grasp for a named use, classically from a fixed semantic taxonomy [26] and more recently with open-vocabulary use descriptions grounded by foundation models [37]; either way, the condition names a use for one object, not the plan that remains. Task-and-motion planning derives grasp constraints by reasoning backward over symbolic operators [14], which requires engineered predicates and models [12]. To our knowledge, **BATON**'s memory is the first to keep rival strategies per subtask and to key retrieval on the remaining plan, through a scheduler that decides *how* each subtask is executed. The handoff contract, drafted and refined in language without training, plays the role of these taxonomies, predicates, and value functions.

Agentic control of frozen VLAs. An LLM agent can control a robot by writing programs over perception and control primitives, revising them from execution feedback with nothing trained [20, 11]; Harness VLA extends this recipe to contact-rich manipulation by wrapping a frozen VLA as a retryable primitive [44] (Sec. 3.1). ASPIRE runs agentic skill discovery in code, with no learned policy in the loop [25]. Adjacent lines place a reasoning agent above the motor level without writing programs. VLM planners orchestrate VLAs as callable tools [19, 5]; dual-system models couple a slow reasoner to a fast policy [33, 29, 27]; world-action models extend the motor level with predictive world modeling [24, 39]. All of these systems make individual skill invocations reliable, or discover new skills. None constrains what one invocation must leave behind for the next, and none addresses how exploration cost scales once a task spans many stages. **BATON** keeps the substrate of Harness VLA and treats the transition itself as the unit of both exploration and learning.

3 METHOD

3.1 PRELIMINARIES

We first review the paradigm of coding agents for robot manipulation [20, 11, 44], in which an LLM agent solves a task by writing programs and delegates only contact-rich motion to a learned policy. Within this paradigm, the agent acts through two kinds of primitive. Analytic primitives execute non-contact motions, such as transporting the end-effector, reorienting the wrist, and actuating the gripper, and are composed over symbolic perception queries. Contact-rich motion is instead handled by a single learned primitive, $\texttt{VLA_ACT}(\ell, \varphi)$, which executes a frozen VLA under instruction ℓ until the stop predicate φ is satisfied. Accordingly, each skill factors into two segments,

$$u = \underbrace{a}_{\text{non-contact: LLM agent}} \triangleright \underbrace{v}_{\text{contact-rich: } \texttt{VLA_ACT}}, \tag{1}$$

where a denotes the non-contact prefix staged by the agent and v denotes the contact-rich segment executed by $\texttt{VLA_ACT}$. Since the VLA is invoked only from a pre-contact pose staged by the agent, it operates near its training distribution. Moreover, no component of the system is trained, and the agent acquires its competence through test-time exploration, consolidating the first successful solution into a language memory $\mathcal{M}$ that is consulted in subsequent attempts. It is worth noting that Eq. (1) does not explicitly represent the transition between a and v, and the moment at which the VLA should be invoked is therefore left to be discovered by trial.

Challenges of long-horizon tasks. Although this paradigm is effective for short-horizon tasks, where test-time exploration is affordable and incurs only a one-time cost, it does not extend directly to long-horizon tasks. We define a long-horizon task $\mathcal{T}$ as an instruction whose execution decomposes into an ordered sequence of subtasks $\langle u_1, \ldots, u_K \rangle$, each of which is a skill of the form in Eq. (1). Compared with a single skill, such a task exhibits two distinctive properties (see Fig. 1). The first is conjunctive success, as $\mathcal{T}$ succeeds only if every subtask u_k succeeds. The second is the interdependence among subtasks. Since the initial state of u_{k+1} is determined by the terminal state of u_k, the execution of each subtask determines the starting condition of the next, while the requirements of the next subtask in turn constrain how the preceding one should be executed. As discussed below, these two properties give rise to two corresponding obstacles.

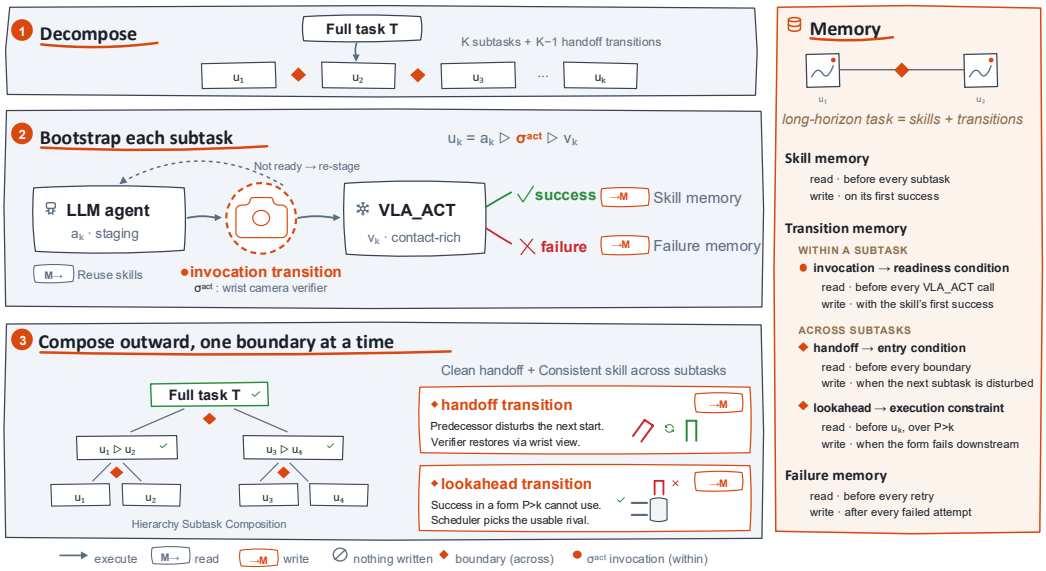

Figure 2: **Overview of BATON.** *Left*: **BATON** decomposes a long-horizon task into subtasks, explores each subtask individually (Sec. 3.2), and composes them into the full task through three transition memories (Sec. 3.3). *Right*: the skill, transition, and failure memories built during exploration.

Obstacle ❶: the cost of exploration. The first obstacle follows from conjunctive success, which requires whole-task exploration to find a working realization of every subtask within a single episode. If a single subtask reaches its first success after T episodes on average, then K subtasks explored jointly require on the order of T^K episodes, since each episode must succeed on all of them. Moreover, failures are uninformative. An unsuccessful episode reveals only that the chain broke, and because the paradigm consolidates only whole-task solutions, the subtasks solved along the way leave nothing in memory. Consequently, exploration that is affordable and self-correcting on a single subtask becomes prohibitively expensive over a chain and retains no partial progress.

Obstacle ❷: the fragility of transitions. The second obstacle follows from the interdependence among subtasks. As noted above, the formulation in Eq. (1) specifies what each segment should accomplish but imposes no condition on the transition from a to v, and the same holds for the boundary between consecutive subtasks. Over a long chain, these unconstrained transitions become the points at which execution breaks down. Within a subtask, the paradigm has no notion of when the scene is ready for the VLA, so the handover is tied to the states encountered during exploration. Between subtasks, the missing conditions matter in both directions. In the forward direction, a subtask inherits the residue of its predecessor. A subsequent failure then becomes ambiguous, since it may stem from either the skill itself or its entry state. In the backward direction, a local success admits many realizations, not all of which the successor can build upon. However, the paradigm stores only the first solution that succeeds, without recording the conditions under which it holds. Whether consecutive subtasks fit together is therefore never checked.

Together, these two obstacles lead to **BATON**'s design. To address the first, exploration must be organized per subtask, so that its cost grows linearly with the chain, each failure is attributed to the subtask responsible, and solved subtasks are retained (Sec. 3.2). To address the second, the transitions both within and between subtasks must be stated as explicit conditions that the system verifies and remembers, rather than left implicit in whichever solution succeeds first (Sec. 3.3).

3.2 HIERARCHICAL SUBTASK EXPLORATION

Rather than exploring a long-horizon task as a whole, **BATON** represents it as a chain of subtasks joined by handoff boundaries,

$$\mathcal{T} = u_1 \triangleright \sigma_{1,2} \triangleright u_2 \triangleright \sigma_{2,3} \triangleright \cdots \triangleright \sigma_{K-1,K} \triangleright u_K, \tag{2}$$

where each u_k is a short-horizon skill of the form in Eq. (1) and $\sigma_{k,k+1}$ is the boundary between u_k and u_{k+1}. We refer to the ordered subtasks $\mathcal{P} = \langle u_1, \ldots, u_K \rangle$ as the plan. Exploration proceeds bottom-up over this chain. Each subtask is first explored in isolation, which keeps the search within

the short-horizon regime where the paradigm of Sec. 3.1 is effective. The agent refines its solution from execution feedback until the subtask first succeeds. Each failed attempt is recorded in memory, so that later attempts avoid repeating it. The successful solution is stored in an abstract, symbolic form, so that it can be reused across instances and tasks. The solved subtasks are then composed hierarchically. Adjacent subtasks are chained into pairs, and each successful pair becomes a verified unit. These units are in turn chained with their neighbors until the full task runs end to end. When a composition fails, both of its parts are already known to succeed on their own, so the search is directed to the boundary between them. A boundary is regarded as crossed once the successor succeeds from the state left by its predecessor.

Compared with whole-task exploration, this bottom-up scheme resolves the first obstacle in three respects. First, each round of search targets a single new element, either a subtask or a boundary between two verified units. Each element is therefore explored only once, and the total cost scales on the order of KT rather than T^K. Second, a failure is confined to the element under exploration and never invalidates a verified unit. Solved subtasks and compositions are thus retained as reusable progress. Third, composition introduces no new skills, since each unit simply executes its parts in order. Memory therefore grows only at boundaries where composition fails. These boundaries correspond to the transitions identified in the second obstacle, and they are the focus of Sec. 3.3.

3.3 Transition-Aware Memory

Hierarchical exploration (Sec. 3.2) makes the subtasks and their boundaries explicit, but it leaves the transitions within and between subtasks unconstrained. **Baton** addresses this obstacle with a transition-aware memory $\mathcal{M}$. We identify three transitions, each with its own entry in $\mathcal{M}$. One lies within a subtask. The *invocation transition* governs when control passes from the agent to the VLA. The other two lie at the boundary $\sigma_{k,k+1}$ between consecutive subtasks. The *handoff transition* governs the state that a subtask delivers to its successor, while the *lookahead transition* governs how the requirements of the successor shape the execution of the current subtask. We describe each transition in turn, and then present the unified store that organizes them and shares them across tasks.

Within a subtask: the invocation transition. To represent the transition left implicit in Eq. (1), **Baton** makes it explicit in the factorization of each subtask,

$$u_k \;=\; \underbrace{a_k}_{\text{non-contact: LLM agent}} \;\triangleright\, \sigma_k^{\text{act}} \,\triangleright\; \underbrace{v_k}_{\text{contact-rich: VLA_ACT}}, \tag{3}$$

where σ_k^{act} denotes the invocation transition at which control passes from the agent to the VLA. Rather than searching for when to invoke the VLA, **Baton** conditions this transition on a *readiness condition* stated in language. Before each invocation, an agent termed verifier checks the condition against the wrist-camera observation, and the agent re-stages the scene whenever it does not hold. When placing an object into a basket, for instance, the condition may require the gripper to lie within the rim of the basket. The readiness condition serves two purposes. During exploration, it guides the search directly, replacing trial over invocation timings. During execution, it keeps the VLA from being invoked in an unsuitable state.

Across subtasks (i): the handoff transition. The handoff transition addresses the forward direction, in which a predecessor disturbs the entry state of its successor, such as a gripper left rotated after opening a door. Since neither skill is at fault in this case, re-exploring either one would not resolve the failure. The boundary $\sigma_{k,k+1}$ instead records an *entry condition*, which specifies what the scene must satisfy for u_{k+1} to begin. This condition belongs to the boundary rather than to either subtask, since the residue that arrives depends on the predecessor while the residue that matters depends on the successor. The verifier checks the condition at the boundary and, when it is violated, restores the scene with analytic primitives before the successor begins. Unlike the initiation sets of classical skill formalisms [36, 16], which are learned offline for each skill, the entry condition is expressed in language and verified on live observations.

Across subtasks (ii): the lookahead transition. The lookahead transition addresses the backward direction, in which a subtask succeeds in a form that its successor cannot use. An object may be grasped in several ways, for instance, but only some of them leave enough clearance for placing it into a confined space. Which execution is appropriate therefore depends not on the subtask alone but on what follows it. The boundary $\sigma_{k,k+1}$ accordingly records an *execution constraint*, which

Table 1: RoboMemArena [18]: per-category and average TSR and CSR, in %. We compare with previous methods and with Harness VLA, which shares all components with **BATON**. The ground-truth oracle executes with perfect memory of the past and bounds what retrospection alone can achieve; it is excluded from bolding. Best per column in bold.

	Transferring		Occlusion		Counting		Sequence		Average	
Method	TSR	CSR	TSR	CSR	TSR	CSR	TSR	CSR	TSR	CSR
GPT-5.4 (VLM only)	13.8	32.9	1.8	9.2	12.9	50.7	15.0	47.3	8.7	30.5
$\pi_{0.5}$ [29]	20.0	42.8	12.7	17.2	14.3	50.9	60.0	71.6	21.5	38.7
HiF-VLA [21]	17.5	38.9	12.7	27.1	8.6	45.9	42.5	70.2	16.9	39.8
MemoryVLA [32]	15.0	37.2	7.3	13.1	14.3	55.1	37.5	65.2	15.0	35.3
MemER [35]	20.0	36.1	16.4	33.2	27.1	65.1	65.0	79.1	27.3	49.1
PrediMem [18]	22.5	45.2	27.3	38.4	45.7	69.3	72.5	89.5	38.5	55.2
PrediMem + GT memory [18]	*32.5*	*54.8*	*33.6*	*49.8*	*51.4*	*75.6*	*85.0*	*92.3*	*46.1*	*64.8*
FrameSamp+Modul [7]	63.8	72.1	39.1	56.5	31.4	57.8	73.8	86.9	46.1	63.9
Harness VLA [44]	50.0	58.3	0.0	2.3	14.3	50.8	75.0	93.1	23.1	37.9
BATON (ours)	**75.0**	**83.3**	**50.0**	**78.6**	**64.3**	**83.3**	**87.5**	**93.8**	**63.5**	**82.9**

specifies the form in which u_k must end for u_{k+1} to proceed. Before executing u_k, **BATON** looks ahead to the remaining plan $\mathcal{P}_{>k} = \langle u_{k+1}, \ldots, u_K \rangle$ and carries out u_k in a form that satisfies the constraints it induces. The same subtask is thus executed differently depending on its successor.

In summary, hierarchical subtask exploration acquires a long-horizon solution piece by piece, confining each search to a single subtask or boundary. The transition-aware memory complements it at the transitions that prove fragile, recording the conditions under which adjacent pieces fit together and verifying them during execution. By keeping exploration cost linear in stage count and making fragile transitions explicit, **BATON** carries a long-horizon task reliably from one subtask to the next.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Benchmark. RoboMemArena [18] consists of 26 household tasks in partially observable kitchen scenes, each chaining 3–9 verification stages over more than 1,000 control steps on average. The tasks fall into four categories by what must be carried across stages: *transferring* (4 tasks), *occlusion* (11), *counting* (7), and *sequence* (4). Each task is instantiated by a scene seed that resamples object placement. All exploration runs on a single reference seed and evaluation runs on ten held-out seeds. We also evaluate **BATON** on a real robot, a PiPER-X 6-DoF research arm, on two long-horizon tasks with 20 trials each. We report two metrics: task success rate (TSR), the percentage of episodes that complete the task, and cumulative success rate (CSR), the percentage of stages completed. A high CSR with a low TSR means that the stages succeed and the chain does not.

Models. For a controlled comparison, **BATON** and the Harness VLA baseline [44] are built on identical components. Both use the π_{RLinf} checkpoint [43] as the frozen VLA policy without any fine-tuning, SAM-3 [4] for segmentation, and GPT-5.6-sol as the backbone of the coding agent, and both follow the coding-agent loop, analytic primitives, and skill memory of Harness VLA. Both systems start from the skill memory that Harness VLA acquires on LIBERO [22], whose skills closely resemble the subtasks of RoboMemArena, and both explore on the reference seed with a budget of 20 episodes per task. Harness VLA spends this budget exploring each task *as a whole*, whereas **BATON** spends it exploring the transitions between subtasks. Performance differences between the two can therefore be attributed to these components rather than to the surrounding infrastructure. We further compare **BATON** with previous methods evaluated on RoboMemArena [18], including a VLM-only agent, the reactive VLA $\pi_{0.5}$ [29], and memory-augmented VLAs [32, 35, 7, 18, 21].

4.2 MAIN RESULTS

As reported in Tab. 1, **BATON** establishes a new state of the art on RoboMemArena, attaining a TSR of 63.5% and a CSR of 82.9%. These results exceed the strongest prior method by 37.7% and 29.7%, and Harness VLA by 40.4 and 45.0 points, respectively. The superiority of **BATON** further persists across all task categories, with TSR gains of 11.2, 10.9, 18.6, and 12.5 points over the best competing

Table 3: Ablation of the three transition memories on RoboMemArena: per-category and average TSR and CSR, in %. Each variant removes one transition of Sec. 3.3 and keeps the hierarchical subtask exploration of Sec. 3.2. Best per column in bold.

	Transferring		Occlusion		Counting		Sequence		Average	
Variant	TSR	CSR	TSR	CSR	TSR	CSR	TSR	CSR	TSR	CSR
w/o invocation	52.5	63.5	36.4	66.5	42.9	61.8	80.0	93.4	47.3	68.9
w/o handoff	57.5	67.1	9.1	44.4	21.4	78.5	77.5	93.3	30.4	64.6
w/o lookahead	62.5	70.8	18.2	47.8	28.6	**83.3**	82.5	**93.8**	37.7	68.0
BATON (full)	**75.0**	**83.3**	**50.0**	**78.6**	**64.3**	**83.3**	**87.5**	**93.8**	**63.5**	**82.9**

method on transferring, occlusion, counting, and sequence, respectively. It is worth emphasizing that these results are achieved with a frozen VLA that receives no fine-tuning on the benchmark, in contrast to the reported policies, which are trained on its data.

A closer examination of the results reveals two insights into the source of these gains. First, Harness VLA fails almost entirely on the occlusion category. Within the 20-episode budget, whole-task exploration rarely reaches a first success on these many-stage tasks and thus writes nothing to memory, whereas BATON reaches a first success on every subtask and boundary within the same budget. More fundamentally, occlusion tasks compose a heterogeneous set of operations, such as opening and closing drawers and microwaves, and thus exhibit a substantially stronger dependence on inter-subtask transitions than the remaining categories. The pronounced improvement of **BATON** on this category therefore directly reflects the contribution of the transition-aware memory. Second, Harness VLA already attains strong performance on the sequence category, in which subtasks are only loosely coupled and transitions are correspondingly less consequential. The comparatively modest gain of **BATON** on this category is consistent with this interpretation. The benefit of **BATON** scales with the extent to which a task depends on the transitions between its subtasks.

Table 2: Real-robot long-horizon experiment results. TSR/ CSR in %, 20 trials per task.

	Cans→plate		Can+bottle→plate		Average	
Method	TSR	CSR	TSR	CSR	TSR	CSR
$\pi_{0.5}$	10.0	31.7	5.0	21.7	7.5	26.7
GPT-5.6-sol (VLM)	15.0	38.3	5.0	25.0	10.0	31.7
Harness VLA [44]	35.0	61.7	20.0	43.3	27.5	52.5
BATON (ours)	**55.0**	**73.3**	**40.0**	**63.3**	**47.5**	**68.3**

As shown in Tab. 2, **BATON** remains effective on a physical robot. We evaluate two long-horizon tasks, namely placing two cans onto a plate in sequence and placing a can and a bottle onto a plate, and compare **BATON** with $\pi_{0.5}$, a vanilla coding agent, and Harness VLA, where the latter two share the GPT-5.6-sol backbone of **BATON**. The baselines exhibit distinct failure modes. $\pi_{0.5}$ fails in a largely consistent manner, as it tends to follow a fixed trajectory regardless of the state left by the preceding subtask. The vanilla agent adapts more flexibly to such states, yet lacks the precision required for contact-rich execution. Harness VLA executes contact precisely whenever the preceding transition is appropriate, but remains vulnerable to failures at the transitions themselves. By additionally securing these transitions, **BATON** achieves the best performance.

4.3 Ablation: Which Transition Matters

To isolate the contribution of each transition memory, we conduct an ablation study on RoboMemArena. All variants retain the hierarchical subtask exploration of Sec. 3.2 and follow the evaluation protocol of Sec. 4.1, while the three transition memories of Sec. 3.3 are removed one at a time. Differences in performance can therefore be attributed to the memory rather than to the exploration. In the *w/o invocation* variant, the VLA is invoked without checking the readiness condition, as in Harness VLA. In *w/o handoff*, the boundary records no entry condition, and each subtask starts from whatever state its predecessor leaves. In *w/o lookahead*, the scheduler is disabled, and memory keeps only the first strategy that succeeded for each subtask, as in prior skill libraries.

The results in Tab. 3 yield three key observations. First, each transition memory is indispensable to **BATON**. Ablating any single transition substantially degrades the overall performance, reducing TSR and CSR by 39.4% and 19.0% on average, respectively. Among the three, the two inter-subtask transitions incur a markedly larger degradation than the invocation transition, underscoring that the coupling between consecutive subtasks constitutes the primary determinant of long-horizon success. Second, the ablations exert a considerably stronger impact on TSR than on CSR. Given that TSR requires the successful completion of every subtask, whereas CSR credits partial progress, this

discrepancy corroborates our central premise that long-horizon failures stem predominantly from the transitions between subtasks rather than from the subtasks themselves. Third, the degradation is most pronounced on the occlusion and counting categories. Compared with transferring and sequence, tasks in these categories entail more frequent gripper reconfiguration and more stringent constraints across subtask boundaries, and thus benefit most substantially from the transition-aware memory. Transferring is the one category where removing the invocation transition hurts most, since its grasps on the cabinet top depend almost entirely on the wrist-view alignment of the readiness condition.

4.4 Qualitative Analysis

Fig. 3 presents a successful evaluation rollout of **Baton** on an occlusion task under a held-out seed. The task requires the robot to open and close three drawers in turn, then place the butter in the empty one. In comparison, Harness VLA, which explores the task as a whole, fails to reach a single success on this task within its exploration budget. Although it already possesses a skill for opening a drawer, it stalls at this stage, repeatedly adjusting its approach without securing the handle until the episode is exhausted. **Baton** avoids this failure by solving each subtask once and composing the solutions, and the resulting rollout exercises all three transitions of Sec. 3.3. At each drawer, the invocation transition defers the VLA until the wrist view shows the handle between the fingers. At each boundary, the handoff transition restores the wrist from the rotated pose left by the previous opening to the pose expected by the next reach. At the first drawer, the lookahead transition selects a grasp that keeps the handle reachable for the subsequent closing motion. Each transition thus intervenes precisely where the chain requires it.

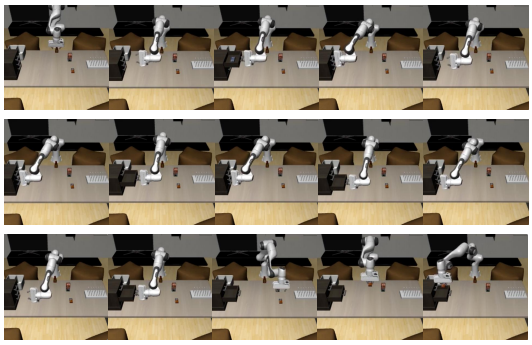

Figure 3: **A successful evaluation rollout of Baton on a 1,800-step RoboMemArena task.** The robot opens and closes three drawers in turn, then places the butter in the empty one.

Baton also transfers to the real robot, as illustrated by the successful long-horizon rollout in Fig. 4. The real-world setting exhibits the same challenge observed in simulation. Chaining subtasks is straightforward when they leave the gripper configuration unchanged. When a subtask instead requires the gripper to be reoriented or realigned with an object, the state left by its predecessor no longer matches what it expects, and chaining fails without transition memory. **Baton** resolves such cases at the transitions, restoring the entry state before each subtask begins and invoking the VLA only once the scene is ready.



Figure 4: Real-robot rollout of **Baton**. The arm transfers two cans onto the plate in turn.

5 Conclusion

Long-horizon manipulation is limited less by individual skills than by their composition, since stages that succeed in isolation still fail once chained. Coding agents over a frozen VLA can compose skills without training, yet none has carried a thousand-step chain, since whole-task exploration is exponentially costly and subtask transitions go unrepresented. We presented **Baton**, to our knowledge the first coding agent to accomplish long-horizon manipulation over a frozen VLA with no parameter updates. It explores one subtask at a time, so that exploration cost grows linearly with the number of stages, and it states the invocation, handoff, and lookahead transitions as conditions verified during execution. On RoboMemArena, **Baton** improves task success by 37.7% and cumulative success by 29.7% over the prior state of the art, with the largest gains where tasks depend most on transitions. More broadly, our results suggest that the bottleneck of long-horizon manipulation lies in the transitions rather than the skills, and that a policy is made reliable over long horizons not by retraining but by governing when it acts and what it inherits.

AI Disclosure. In this work, we used generative AI tools to polish paper writing and image illustrations. All authors reviewed the polished text against the original draft to ensure that no claim or result was altered, and independently verified that the edited figures accurately depict our method. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Reproducibility Statement. We are committed to ensuring the reproducibility of our results. The source code will be publicly released upon acceptance. Detailed descriptions of our experimental setup and implementation details are provided in Sec. 4.

Ethics Statement. This work focuses on robot manipulation in simulated and controlled laboratory environments, using only publicly available benchmarks and datasets. It does not involve human subjects, personal data, or sensitive information. We do not foresee any direct negative societal impact from this research.

REFERENCES

[1] Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[2] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[3] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[4] Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris Coll-Vinent, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, et al. SAM 3: Segment anything with concepts. In *International conference on learning representations*, 2026.

[5] Siyi Chen, Hugo Hadfield, Alex Zook, Mikaela Angelina Uy, Chan Hee Song, Erwin Coumans, Xuning Yang, Faisal Ladhak, Qing Qu, Stan Birchfield, et al. Volo: A physical orchestrator for open-vocabulary long-horizon manipulation. *arXiv preprint arXiv:2606.07723*, 2026.

[6] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems (RSS)*, 2023. arXiv:2303.04137.

[7] Yinpei Dai, Hongze Fu, Jayjun Lee, Yuejiang Liu, Haoran Zhang, Jianing Yang, Chelsea Finn, Nima Fazeli, and Joyce Chai. RoboMME: Benchmarking and understanding memory for robotic generalist policies. *arXiv preprint arXiv:2603.04639*, 2026.

[8] Yu Fang, Yuchun Feng, Dong Jing, Jiaqi Liu, Yue Yang, Zhenyu Wei, Daniel Szafir, and Mingyu Ding. When vision overrides language: Evaluating and mitigating counterfactual failures in VLAs. *arXiv preprint arXiv:2602.17659*, 2026.

[9] Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, et al. LIBERO-plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.

[10] Ethan Foong, Yunshuang Li, Hao Jiang, Gaurav S. Sukhatme, and Daniel Seita. Handful: Sequential grasp-conditioned dexterous manipulation with resource awareness. *arXiv preprint arXiv:2604.25126*, 2026.

[11] Letian Fu, Justin Yu, Karim El-Refai, Ethan Kou, Haoru Xue, Huang Huang, Wenli Xiao, Guanzhi Wang, Dantong Niu, Fei-Fei Li, Guanya Shi, Jiajun Wu, Shankar Sastry, Yuke Zhu, Ken Goldberg, and Linxi Fan. CaP-X: A framework for benchmarking and improving coding agents for robot manipulation. *arXiv preprint arXiv:2603.22435*, 2026.

[12] Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Integrated task and motion planning. *Annual Review of Control, Robotics, and Autonomous Systems*, 2021.

[13] Li Ji, Siyin Wang, Pengfang Qian, Xiaopeng Yu, Yihai Tian, Zhaoye Fei, Jingjing Gong, and Xipeng Qiu. Hime: Hierarchical embodied memory for long-horizon vision-language-action control. *arXiv preprint arXiv:2607.03449*, 2026.

[14] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical task and motion planning in the now. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[15] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. OpenVLA: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[16] George Konidaris and Andrew Barto. Skill discovery in continuous reinforcement learning domains using skill chaining. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2009.

[17] Youngwoon Lee, Joseph J. Lim, Anima Anandkumar, and Yuke Zhu. Adversarial skill chaining for long-horizon robot manipulation via terminal state regularization. In *Conference on Robot Learning (CoRL)*, 2021. arXiv:2111.07999.

[18] Huashuo Lei, Wenxuan Song, Huarui Zhang, Jieyuan Pei, Jiayi Chen, Haodong Yan, Han Zhao, Pengxiang Ding, Zhipeng Zhang, Lida Huang, Donglin Wang, Yan Wang, and Haoang Li. Robomemarena: A comprehensive and challenging robotic memory benchmark. *arXiv preprint arXiv:2605.10921*, 2026.

[19] Zixing Lei, Changxing Liu, Yichen Xiong, Minhao Xiong, Yuanzhuo Ding, Zhipeng Zhang, Weixin Li, and Siheng Chen. Towards long-horizon embodied agents with tool-aligned vision-language-action models. *arXiv preprint arXiv:2605.13119*, 2026.

[20] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[21] Minghui Lin, Pengxiang Ding, Shu Wang, Zifeng Zhuang, Yang Liu, Xinyang Tong, Wenxuan Song, Shangke Lyu, Siteng Huang, and Donglin Wang. Hif-vla: Hindsight, insight and foresight through motion representation for vision-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

[22] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, volume 36, 2023.

[23] Yuhan Liu, Xinyu Zhang, Litao Liu, and Abdeslam Boularias. Foresight residual rl for long-horizon robot manipulation with vision-language-action models. *arXiv preprint arXiv:2607.16506*, 2026.

[24] Yushan Liu, Peibo Sun, Shoujie Li, Yifan Xie, Lingfeng Zhang, Xintao Chao, Shiyuan Dong, Fang Chen, Xiao-Ping Zhang, and Wenbo Ding. OA-WAM: Object-addressable world action model for robust robot manipulation. *arXiv preprint arXiv:2605.06481*, 2026.

[25] Runyu Lu, Yubo Wu, Ethan Kou, Letian Fu, Wenli Xiao, Ajay Mandlekar, Yinzhen Xu, Guanya Shi, Ken Goldberg, Ang Chen, et al. Aspire: Agentic skills discovery for robotics. *arXiv preprint arXiv:2607.00272*, 2026.

[26] Adithyavairavan Murali, Weiyu Liu, Kenneth Marino, Sonia Chernova, and Abhinav Gupta. Same object, different grasps: Data and semantic knowledge for task-oriented grasping. In *Proceedings of the 4th Conference on Robot Learning (CoRL)*, 2021.

[27] Jiaqi Peng, Xiqian Yu, Delin Feng, Yuqiang Yang, Wenzhe Cai, Jing Xiong, Ganlin Yang, Jinliang Zheng, Jiafei Cao, Xueyuan Wei, et al. Cortex: A bidirectionally aligned embodied agent framework for long-horizon manipulation. *arXiv preprint arXiv:2607.05377*, 2026.

[28] Physical Intelligence. openpi: Open-source robot foundation models. https://github.com/Physical-Intelligence/openpi, 2025. Accessed 2026-09-25.

[29] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[30] Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*, 2011.

[31] Ke Rui, Yushen Zuo, Jiawei Wang, Haoran Jia, Jinming Ma, Weitao Zhou, and Minglei Li. Diagnosing semantic handoff failures in agent-orchestrated vision-language-action skill composition. *arXiv preprint arXiv:2607.06256*, 2026.

[32] 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. In *International Conference on Learning Representations (ICLR)*, 2026.

[33] Lucy Xiaoyang Shi, Brian Ichter, Michael Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, et al. Hi robot: Open-ended instruction following with hierarchical vision-language-action models. *arXiv preprint arXiv:2502.19417*, 2025.

[34] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023.

[35] Ajay Sridhar, Jennifer Pan, Satvik Sharma, and Chelsea Finn. Scaling up memory for robotic control via experience retrieval. In *International Conference on Learning Representations (ICLR)*, 2026.

[36] Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 1999.

[37] Chao Tang, Dehao Huang, Wenlong Dong, Ruinian Xu, and Hong Zhang. Foundationgrasp: Generalizable task-oriented grasping with foundation models. *IEEE Transactions on Automation Science and Engineering*, 2025.

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

[39] Fangyuan Wang, Ziyuan Wang, Guorui Pei, Mengshi Zhang, Canxi Liang, Jun Hu, Zhongxuan Li, Jinsong Wu, Ning Han, Zeqing Zhang, et al. World models for robotic manipulation: A survey. *arXiv preprint arXiv:2606.00113*, 2026.

[40] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

[41] John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, 2024.

[42] Yue Yang, Linfeng Zhao, Mingyu Ding, Gedas Bertasius, and Daniel Szafir. Boss: Benchmark for observation space shift in long-horizon task. *IEEE Robotics and Automation Letters*, 2025.

[43] Chao Yu, Yuanqing Wang, Zhen Guo, Hao Lin, Si Xu, Hongzhi Zang, Quanlu Zhang, Yongji Wu, Chunyang Zhu, Junhao Hu, et al. RLinf: Flexible and efficient large-scale reinforcement learning via macro-to-micro flow transformation. In *20th USENIX Symposium on Operating Systems Design and Implementation (OSDI 26)*, 2026.

[44] Yixian Zhang, Huanming Zhang, Feng Gao, Xiao Li, Zhihao Liu, Chunyang Zhu, Jiaxing Qiu, Yuchen Yan, Jiyuan Liu, Wenhao Tang, et al. Harness vla: Steering frozen vlas into reliable manipulation primitives via memory-guided agents. *arXiv preprint arXiv:2607.08448*, 2026.

[45] Meng Zheng, Samhita Marri, Anwesa Choudhuri, Benjamin Planche, Zhongpai Gao, Van Nguyen Nguyen, Terrence Chen, Girish Chowdhary, and Ziyan Wu. Failing forward: Adaptive failure-informed learning for vision-language-action models. *arXiv preprint arXiv:2605.08434*, 2026.