OMNIACT: A FRAMEWORK FOR LONG-HORIZON CYBER-PHYSICAL EMBODIED AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Embodied agents deployed in everyday environments must carry out extended tasks across physical and digital domains while respecting user requirements and responding to unexpected execution outcomes. This requires coordinating robotic skills and digital tools with context management and timely feedback. We present **OmniAct**, a framework for long-horizon cyber-physical embodied agents that integrates multimodal planning, hierarchical memory, and asynchronous visual monitoring. A shared skill interface supports task decomposition and execution across heterogeneous tools and robotic policies. Sensory, episodic, and reflective memory organize recent observations, task events, and reusable experience, while visual monitoring enables interruption and replanning during physical execution. Constraints derived from earlier user dialogue are retained in reflective memory and supplied to the monitor, allowing historical requirements to guide stopping decisions. Across 40 manipulation and navigation tasks on two robotic platforms, OmniAct achieves overall end-to-end success rates of 63.3% and 65.0%, respectively, exceeding the evaluated planning, agent-harness, and composed baselines. Ablations support the contributions of memory organization and timely intervention. Further evaluations show improved constraint satisfaction across six planner backbones and substantially reduced input-context growth over 270 interaction rounds. These results demonstrate the value of coordinating memory and execution feedback for long-horizon embodied tasks.

1 INTRODUCTION

Deploying embodied agents in everyday environments requires them to translate human intent into reliable action across extended interactions. In homes and workplaces, user requests are embedded in an evolving physical and social context: moving objects, actions produce unexpected outcomes, and preferences expressed earlier continue to constrain what the agent should do. Fulfilling these requests often involves both physical actions and digital services, from operating appliances and manipulating objects to retrieving information and navigating between locations. Reliable embodied assistance therefore requires maintaining relevant context and using execution feedback to coordinate these capabilities throughout a task.

Foundation models provide many of the capabilities needed for such systems. Language-based planners compose skills and incorporate execution feedback (Ahn et al., 2022; Liang et al., 2023; Huang et al., 2022), while vision-language-action models learn task-conditioned control from visual and language inputs (Kim et al., 2024; Black et al., 2024; Ma et al., 2024; Wang et al., 2026; Liu et al., 2025). General-purpose agent harnesses provide infrastructure for model-driven tool use (OpenClaw, 2026; Nous Research, 2026), and memory methods organize information beyond the immediate interaction (Packer et al., 2023; Xu et al., 2025). Robotic monitoring methods further generate and evaluate execution constraints to detect failures and trigger recovery (Guo et al., 2023; Zhou et al., 2025). Bringing these capabilities together requires a task-level runtime that coordinates their inputs, state, and execution feedback throughout a workflow.

Existing approaches nevertheless leave gaps in supporting the workflows and handling the execution failures illustrated in Fig. 1. In panel (a), a single request spans digital tools and robotic executors. Standardized interfaces simplify their invocation, but the planner still needs to relate tool outputs to physical progress and retain relevant requirements across successive operations. History concatenation

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.

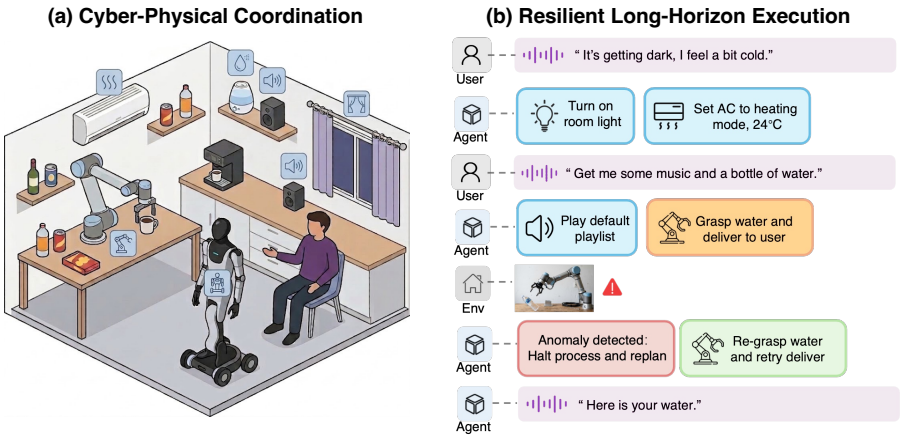


Figure 1: Two key challenges for real-world long-horizon embodied deployment. (a) A single user command may require joint orchestration of IoT devices, robotic manipulators, mobile platforms, and web APIs. (b) Over extended horizons, unexpected physical failures must be autonomously detected and recovered from without human intervention.

introduces growing amounts of irrelevant context, while compression and retrieval can omit constraints that remain applicable. Panel (b) highlights the consequences when physical execution deviates from the plan. Online monitoring methods detect constraint violations and trigger recovery (Guo et al., 2023; Zhou et al., 2025), but the appropriate intervention may also depend on earlier user dialogue. Preserving those requirements and making them available to subsequent monitoring decisions remains a separate context-management problem. Reliable execution therefore requires connecting these capabilities throughout the workflow.

To address these coordination challenges, we introduce **OmniAct**, a framework for long-horizon cyber-physical embodied agents. OmniAct integrates multimodal planning, hierarchical memory, and asynchronous visual monitoring to connect task context with execution feedback. A semantic planner decomposes user requests and dispatches heterogeneous executors through a shared skill interface. Sensory memory maintains recent visual context, episodic memory consolidates interactions at event boundaries, and reflective memory retains reusable experience, user preference and planner-recorded constraints. During physical execution, a visual monitor evaluates progress and can interrupt a skill to return control to the planner. Relevant rules derived from earlier user dialogue are stored in reflective memory and supplied alongside the current task to condition monitoring. This connects historical requirements to execution-time decisions within the same planning and recovery loop.

We evaluate OmniAct on 40 tasks spanning manipulation and navigation, with robotic skills, four IoT devices, and twelve digital tool APIs. Complete-system comparisons include embodied planners, general-purpose harnesses, and combinations of existing memory and monitoring methods (Sec. 4.2). System and component ablations examine memory organization, intervention timing, and adherence to historical stopping rules (Sec. 4.3). We further evaluate constraint satisfaction across six planner backbones and input-context growth over extended interaction traces (Secs. 4.4 and 4.5). These evaluations show improved task completion and constraint adherence, together with substantially slower growth of the planner's input context.

Our contributions are threefold:

1. We present OmniAct, a framework for embodied agents that coordinate robotic skills and digital tools across long-horizon cyber-physical tasks through a shared skill interface, while preserving executor-specific control representations.
2. We integrate sensory, episodic, and reflective memory with interruptible execution, enabling the planner to retain task context and supply historical user constraints to asynchronous visual monitoring.
3. We evaluate the system against planning baselines, agent harnesses, and composed alternatives, and analyze component contributions, planner-backbone transfer, and context efficiency through targeted experiments.

2 Related Work

2.1 Foundation Models for Embodied Planning and Control

Foundation models support embodied decision-making at both the task and action levels. SayCan (Ahn et al., 2022) combines language-based task relevance with skill affordances, while Code as Policies (Liang et al., 2023) generates programs that compose perception and control APIs. Inner Monologue (Huang et al., 2022) incorporates environmental feedback into language-based planning. Multimodal models such as PaLM-E (Driess et al., 2023) and RoboBrain2 (Team et al., 2025) further connect visual observations with embodied reasoning and planning. At the control level, vision-language-action models, including RT-2 (Brohan et al., 2023), OpenVLA (Kim et al., 2024), and π_0 (Black et al., 2024), learn mappings from visual and language inputs to robot actions. General-purpose agent runtimes, such as Hermes (Nous Research, 2026) and OpenClaw (OpenClaw, 2026), provide complementary infrastructure for model-driven tool use. Bringing these planning and control capabilities into broader real-world deployment requires a common interface for robotic skills and digital tools, together with persistent context management and execution feedback. OmniAct supports this integration through shared skill routing, hierarchical memory, and asynchronous visual monitoring.

2.2 Memory-Augmented Agents

Memory allows agents to retain experience beyond the immediate interaction and use it in subsequent decisions. Reflexion (Shinn et al., 2023) stores verbal reflections on task feedback to improve later trials, while MemGPT (Packer et al., 2023) manages information across memory tiers to extend the effective context of language models. Embodied memory methods support experience reuse through language-program retrieval (Sarch et al., 2023) and environmental grounding through hierarchical semantic representations (Xie et al., 2024) or dynamic spatial memory (Liu et al., 2024). A-MEM (Xu et al., 2025) explores adaptive memory organization through linked notes and incremental updates. More recently, general-purpose agent harnesses such as OpenClaw (OpenClaw, 2026) and Hermes (Nous Research, 2026) bring tool execution, context management, and persistent memory into a shared runtime, opening opportunities to integrate memory more closely with embodied agent workflows. Prior memory methods mainly address general-purpose interactions or specific embodied tasks, with limited coordination across skills and tools. OmniAct combines sensory, episodic, and reflective memory to organize observations, task events, and experience for long-horizon embodied execution.

2.3 Robotic Failure Detection and Recovery

Execution feedback is essential when physical outcomes diverge from a robot's plan. DoReMi (Guo et al., 2023) detects violations of language-generated constraints using vision-language feedback and initiates recovery from plan-execution misalignment. Code-as-Monitor (Zhou et al., 2025) expresses spatiotemporal constraints as executable monitoring programs over tracked geometric elements, supporting both reactive and proactive failure detection. Complementary work studies how to explain failures and select corrective actions. REFLECT (Liu et al., 2023) constructs hierarchical summaries of multisensory robot experiences for failure explanation and correction planning. AHA (Duan et al., 2024) trains a vision-language model to detect and reason about manipulation failures, providing feedback that can be incorporated into planning systems. Existing approaches emphasize detecting execution failures, while intervention and recovery may also depend on constraints established in earlier interactions. OmniAct combines visual monitoring with hierarchical memory to guide whether execution should continue, resume, or terminate under these historical constraints.

3 Methodology: The OmniAct Framework

3.1 Problem Formulation

A key challenge in building persistent embodied agents is to enable a single planning system to coordinate diverse executors spanning both cyber (APIs, IoT devices) and physical (robotic manipulation, navigation) domains under partial observability. We consider long-horizon tasks

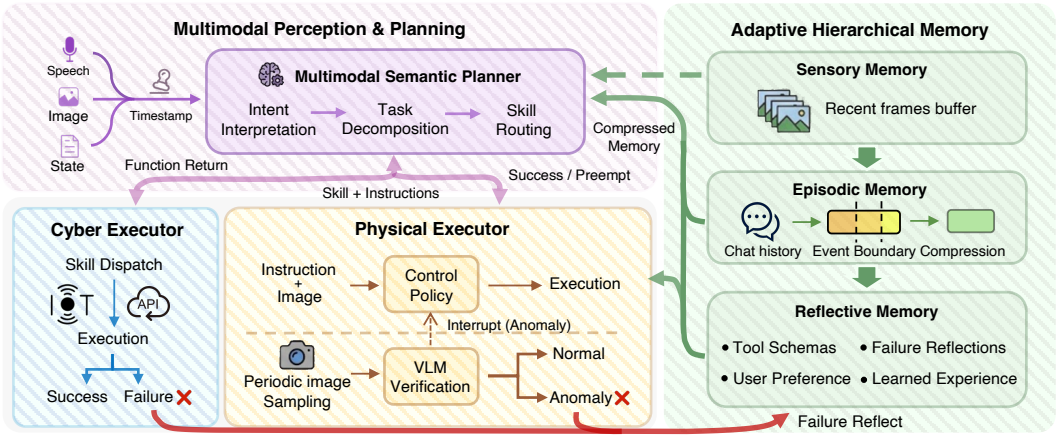


Figure 2: **Overview of the OmniAct framework.** The system comprises three modules: (1) *Multimodal Semantic Planner* fuses speech, image, and state inputs for intent interpretation, task decomposition, and skill routing; (2) *Closed-Loop Execution Engine* with asynchronous visual preemption that periodically verifies physical progress via a VLM and triggers interruption upon anomaly; and (3) *Adaptive Hierarchical Memory* (sensory, episodic, reflective) employing event-boundary compression for bounded context growth and storing failure reflections for cumulative adaptation.

requiring coordination of robotic skills and digital tools under partial observability. OmniAct provides a shared skill interface while preserving executor-specific control representations.

At each planning timestep t, the system receives a multimodal observation $o_t = (v_t, s_t)$, where v_t is a visual frame and s_t denotes discrete state information from sensors or tool APIs. The multimodal context c_t integrates o_t with the current user intent (e.g., transcribed speech) and system state. Since the system must dispatch to fundamentally different executors ranging from stateless API calls to continuous robotic control, a single low-level action representation is insufficient. We therefore define a skill-routing action space: each action is a tuple $a_t = (\tau_t, g_t)$, where $\tau_t \in \mathcal{T}$ selects the target skill and g_t specifies skill-specific conditioning parameters. Memory context $\mathcal{H}_t$, retrieved from hierarchical memory (Sec. 3.4), provides compressed prior experience. Given the planning model π, the high-level planning policy is:

$$a_t \sim \pi(\cdot \mid c_t, \mathcal{H}_t, \mathcal{T}). \tag{1}$$

3.2 Multimodal Semantic Planner

The semantic planner instantiates π to interpret user intent, decompose tasks, and route sub-goals to available skills. Given multimodal context c_t and memory context $\mathcal{H}_t$, it grounds instructions in visual observations and tool states, selects skills $\tau_t \in \mathcal{T}$, and specifies their conditioning parameters g_t. Retrieved history provides task outcomes, relevant experience, and user preferences to inform planning. When needed, the planner also records monitoring constraints from prior interactions to supplement the monitor's default prompt and current task description. It outputs structured JSON specifications supporting parallel skill dispatch where applicable. Output formats and monitoring prompts are provided in the Appendix.

3.3 Closed-Loop Execution with Asynchronous Visual Preemption

Visual feedback for low-level control does not by itself ensure compliance with task-level requirements and constraints established during earlier interactions. OmniAct therefore runs an asynchronous visual monitor alongside physical execution. Digital tools provide return values and execution status, while robotic skills are additionally monitored through periodic visual observations.

For the skill dispatched at planning step t, the monitor prompt is constructed as:

$$p_t^{\text{mon}} = p_{\text{default}} \oplus d(a_t) \oplus C_t^{\text{mon}}, \tag{2}$$

where p_{default} specifies the default monitoring instructions, $d(a_t)$ describes the current task, and C_t^{mon} contains additional constraints previously recorded by the planner. The operator $\oplus$ denotes prompt concatenation; when no additional constraints are present, C_t^{mon} is empty. During execution, the monitor evaluates recent visual observations against this prompt to assess task progress and identify execution anomalies or constraint violations. Consequently, similar visual states can warrant different interventions depending on requirements established in earlier interactions.

When execution remains consistent with the task and monitoring constraints, the active skill continues under periodic observation. Otherwise, the execution engine requests interruption and returns the monitoring feedback to the planner, which uses the current observations and relevant history to revise subsequent actions or terminate the task. If execution proceeds, the selected skill is dispatched and monitored with its task description and applicable constraints, closing the loop between planning, physical execution, and visual feedback. User-specified stopping conditions may instead end the task without further attempts. This task-level loop complements the executor's low-level control and protection mechanisms.

3.4 Adaptive Hierarchical Memory

Long-horizon execution requires retaining information with different temporal roles: recent observations for tracking physical changes, task history for maintaining continuity, and persistent rules and experience for guiding subsequent decisions. OmniAct organizes these functions into sensory, episodic, and reflective memory.

Sensory Memory. A lightweight FIFO buffer retains recent visual frames for the control policy and visual monitor. It provides short-term perceptual context for assessing ongoing execution without accumulating raw visual observations in the planner's context.

Episodic Memory. Interaction history is compressed at semantic event boundaries, such as sub-task completion or relevant state changes. Timestamped records preserve task events, state transitions, and action outcomes while reducing redundant observations. This representation supports temporal ordering and task continuity, with storage growth determined by the number and size of retained event summaries rather than the full observation stream.

Reflective Memory. Alongside static tool and skill descriptions, this level retains persistent user requirements, planner-generated monitoring rules, and reusable execution experience. During interaction, the planner can summarize requirements into explicit rules and store them for subsequent monitoring. Relevant rules are incorporated into C_t^{mon}, allowing earlier instructions to govern intervention during later execution. Execution difficulties can also produce records of hypothesized causes and candidate corrective strategies for future planning. Reflective memory thus supplies persistent guidance for both skill selection and execution monitoring.

4 Experiments

We evaluate OmniAct's task performance, component contributions, planner-backbone transfer, and context efficiency. Following the experimental setup in Sec. 4.1, we compare complete systems on manipulation and navigation tasks requiring cyber-physical coordination (Sec. 4.2). We then examine memory and monitoring through system and component ablations (Sec. 4.3), assess constraint satisfaction across planner backbones (Sec. 4.4), and measure input-context growth over extended interactions (Sec. 4.5).

4.1 Experimental Setup

Platforms and Tasks. The robot evaluation uses a UR5e manipulator and a Keenon mobile robot, together with four household IoT devices and twelve virtual tool APIs. The suite contains 40 tasks, with 20 tasks per platform. Each domain includes four physical-only tasks (L1), eight tasks combining physical execution with digital tools (L2), and eight tasks additionally requiring multimodal interaction and broader tool coordination (L3). Each task is repeated three times, giving 60 task trials per platform and method, with completion assessed by human evaluation. Task definitions are provided in the Appendix.

Table 1: End-to-end (E2E) and sub-task (Sub) success rates (%) across task complexity levels. All reports trial-weighted E2E across L1–L3. CaM denotes adapted Code-as-Monitor, and Both combines A-MEM and CaM. Bold indicates the best result in each column, including ties.

Method	Manipulation (UR5e)						Navigation (Keenon)					
	L1 E2E	L2 Sub	L2 E2E	L3 Sub	L3 E2E	All E2E	L1 E2E	L2 Sub	L2 E2E	L3 Sub	L3 E2E	All E2E
Embodied planning baselines												
SayCan	58.3	62.5	45.8	60.9	37.5	45.0	75.0	62.5	37.5	57.9	20.8	38.3
Code as Policies	66.7	31.3	25.0	20.8	12.5	28.3	58.3	62.5	50.0	45.8	37.5	46.7
RoboBrain2	66.7	39.6	4.2	4.4	0.0	15.0	66.7	41.6	12.5	25.0	12.5	23.3
General-purpose agent harnesses												
OpenClaw	66.7	64.1	50.0	69.4	37.5	48.3	66.7	69.4	54.2	65.3	37.5	50.0
Hermes	75.0	67.5	54.2	73.9	41.7	53.4	75.0	66.7	50.0	72.2	45.8	53.3
Composed baselines												
Hermes + A-MEM	75.0	67.5	50.0	73.9	41.7	51.7	75.0	75.0	58.3	76.4	50.0	58.3
Hermes + CaM	**91.7**	72.5	58.3	77.8	45.8	60.0	75.0	79.2	62.5	77.8	50.0	60.0
Hermes + Both	83.3	72.5	58.3	77.8	45.8	58.3	**83.3**	80.6	62.5	79.2	50.0	61.7
OmniAct	**91.7**	**75.0**	**62.5**	**80.5**	**50.0**	**63.3**	**83.3**	**83.3**	**66.7**	**80.6**	**54.2**	**65.0**

Models and Compared Systems. We compare OmniAct with three groups of baselines: embodied planning methods, including SayCan (Ahn et al., 2022), Code as Policies (Liang et al., 2023), and RoboBrain2 (Team et al., 2025); general-purpose agent harnesses, including OpenClaw (OpenClaw, 2026) and Hermes (Nous Research, 2026); and composed systems that augment Hermes with A-MEM (Xu et al., 2025), adapted Code-as-Monitor (Zhou et al., 2025), or both. The composed baselines assess whether existing memory and monitoring methods can provide comparable system-level performance. All systems use Gemini-3.1-Pro for high-level planning, except RoboBrain2, which retains its own 32B backbone and has demonstrated competitive performance on embodied planning benchmarks (Team et al., 2025).The comparison uses a common skill and tool interface across systems.

Metrics. We report end-to-end success (E2E) and sub-task success (Sub). E2E measures the fraction of episodes that satisfy all required task objectives, while Sub measures the average fraction of prescribed sub-tasks completed within each episode. Overall E2E is aggregated across task trials, corresponding to weights of 0.2, 0.4, and 0.4 for L1, L2, and L3 respectively. These metrics capture complete task execution and partial progress.

Low-Level Executor Reference. To characterize the underlying execution capability, we report Direct Policy at the skill-call level. The manipulation and navigation policies achieve average success rates of 65% and 75% under 20 trials. The imperfect reliability of individual skills motivates evaluating how each system handles execution failures over multi-step tasks.

4.2 Long-Horizon Cyber-Physical Orchestration

Table 1 shows that OmniAct achieves the highest overall E2E success in both task domains, reaching 63.3% in manipulation and 65.0% in navigation. It exceeds the strongest baseline in each domain—Hermes + CaM and Hermes + Both, respectively—by 3.3 percentage points, maintaining an advantage over systems equipped with additional memory or monitoring components.

The planning-oriented baselines become less effective as tasks require broader cyber-physical coordination. SayCan's E2E success decreases from 58.3% at L1 to 37.5% at L3 in manipulation, and from 75.0% to 20.8% in navigation. RoboBrain2 achieves 66.7% at L1 on both platforms, but reaches only 0.0% and 12.5% at L3. These results expose a substantial gap between performance on physical-only tasks and integrated workflows involving interaction, digital tools, and robotic execution. They

suggest that the planning capabilities provided by these configurations are insufficient to reliably complete the more complex tasks in our suite.

General-purpose agent harnesses provide a stronger starting point for these workflows. Hermes achieves 53.3% overall E2E in both domains, exceeding the strongest planning-oriented baselines at 45.0% in manipulation and 46.7% in navigation; OpenClaw also improves over these baselines overall. However, their performance still declines on L3 tasks: OpenClaw and Hermes achieve 37.5% and 41.7% in manipulation, and 37.5% and 45.8% in navigation. Partial progress remains substantially higher than full completion: for example, Hermes obtains 73.9% Sub but 41.7% E2E on L3 manipulation. Thus, substantial portions of a workflow can be completed while the full task remains unsuccessful.

The composed baselines further improve task performance, although the gains depend on the domain and component combination. Hermes + CaM reaches 60.0% overall E2E in both domains. Adding A-MEM improves native Hermes in navigation but not manipulation, and Hermes + Both performs below Hermes + CaM in manipulation while exceeding it in navigation. These differences show that adding memory and monitoring does not produce uniformly additive gains in the evaluated configurations, motivating a closer examination of how the components are organized and used during execution.

OmniAct ties the best L1 performance on both platforms and achieves the highest E2E success at L2 and L3. On L3 tasks, it reaches 50.0% in manipulation and 54.2% in navigation, compared with the strongest baseline scores of 45.8% and 50.0%, respectively. It also obtains the highest Sub scores across all L2/L3 conditions, showing improvements in both partial progress and complete task execution. The following ablations examine the contributions of memory organization, execution monitoring, and their interaction to these system-level results.

4.3 System and Component Ablations

We first examine the contributions of memory and execution monitoring within the same OmniAct runtime, then study the memory hierarchy and monitoring behavior in greater detail. Comparisons within each experiment share the planner backbone, skill inventory, low-level policies.

4.3.1 Memory and Monitoring

We compare four configurations by independently enabling memory augmentation and online monitoring. The iterative baseline retains current observations, within-task working history, post-action feedback, and replanning. Memory only adds hierarchical memory; Monitor only adds execution-time observation and interruption; Full enables both. This design evaluates each component without changing the surrounding agent framework.

Table 2 reports the highest overall success for Full (64.2%), followed by Monitor only (58.3%), Memory only (35.8%), and Iterative (25.8%). Monitoring accounts for the larger improvement in both domains, while memory provides additional gains when monitoring is enabled. Memory alone benefits navigation more than manipulation, raising success from 26.7% to 43.3%, compared with 25.0% to 28.3%. This difference suggests that the usefulness of retained context depends on the task domain. Neither component alone matches Full, supporting their combined use for context management and execution feedback.

Table 2: Gemini 3.1 system comparison in simulation. E2E success (%); 120 episodes per variant.

Configuration	Manip.	Nav.	All
Iterative	25.0	26.7	25.8
Memory only	28.3	43.3	35.8
Monitor only	55.0	61.7	58.3
Full	**63.3**	**65.0**	**64.2**

4.3.2 Memory Organization

We evaluate memory organization on the full manipulation suite of 40 cases, each repeated three times as independent episodes, yielding 120 episodes per variant. The planner, skills, and monitoring configuration are held fixed. Each layer-removal variant disables only the named component. Removing sensory memory (S) eliminates the policy's short-term frame buffer while retaining current observations and the monitor's visual input. Removing episodic (E) or reflective (R) memory disables the corresponding dynamic records while preserving the other layers and static tool knowledge. Flat

E/R retains sensory memory but selects episodic and reflective records from a shared pool under the same total memory budget. A-MEM replaces E/R organization within the same runtime.

Table 3 reports the highest success for Full (63.3%). Removing S reduces success by 4.2 percentage points, whereas removing E and R causes larger decreases of 8.3 and 10.8 points, respectively. The larger losses from removing E/R indicate that access to current observations alone does not substitute for retaining task events and reflective records. Flat E/R reaches 56.7%, below Full despite preserving all three components, supporting separate organization of episodic and reflective information under the evaluated budget. Full also exceeds A-MEM by 9.2 points. Together, these comparisons support the memory hierarchy for manipulation task execution; the independent-episode setting evaluates its contribution within tasks rather than cross-task knowledge reuse.

Table 3: Memory ablation on 40 manipulation cases, each repeated three times. Drop is relative to Full.

Configuration	E2E (%)	Drop (pp)
−S	59.2	4.2
−E	55.0	8.3
−R	52.5	10.8
Flat E/R	56.7	6.7
A-MEM	54.2	9.2
Full	**63.3**	0.0

4.3.3 Context-Aware Execution Monitoring and Recovery

We evaluate seven execution categories: blockage, workspace intrusion, execution deviation, and boundary-limit events require intervention; normal completion, slow but valid execution, and irrelevant changes outside the workspace require continued execution without unnecessary interruption. We also evaluate stopping rules introduced in earlier user dialogue: stopping when a specified object appears in view (Object), or when an object involved in the task meets a specified condition (State). Each category contains four cases repeated three times, giving 84 execution trials per timing configuration and 24 historical-constraint trials per compared system. We compare Post-hoc verification at skill termination, Delayed monitoring with online observation but intervention deferred to the common endpoint, and Online monitoring with immediate interruption. All conditions retain the same low-level protections. Precision and recall use anomalies as the positive class; unproductive execution time is measured on blockage and deviation. Historical-constraint scores measure instruction-consistent stopping, treating required termination as a correct outcome.

Table 4: Monitoring under execution disturbances and historical constraints. (A) Intervention timing. (B) Instruction-consistent stopping rates (%).

A. Intervention timing			
Metric	Post-hoc	Delayed	Online
E2E (%)	35.7	36.9	**76.2**
Precision (%)	92.3	94.7	**95.2**
Recall (%)	75.0	75.0	**83.3**
Unprod. time (s)	8.7	8.1	**1.6**
B. History-conditioned behavior			
Method	Object	State	All
DoReMi	33.3	25.0	29.2
CAM	41.7	33.3	37.5
OmniAct	**75.0**	**66.7**	**70.8**

Table 4 shows that online intervention reduces unproductive execution in blockage and deviation scenarios, while shared protection mechanisms limit differences in intrusion and boundary cases. Normal controls assess whether responsiveness comes at the cost of unnecessary interruption. OmniAct also better follows both types of historical stopping rules than the compared systems. These results support using planner-recorded constraints in reflective memory to guide execution monitoring, including conditions whose relevance is established by earlier dialogue rather than the current task alone.

4.4 Planner-Backbone Transfer

We examine whether OmniAct's benefits extend across planner backbones using a long-horizon constraint-satisfaction evaluation, by constructing interaction histories containing more than 40k tokens, with earlier user preferences constraining later requests. For each of six backbones, we compare Raw History, which supplies the uncompressed interaction history, with OmniAct's memory-based context organization under same 30 episodes. We vary the planner while retaining the surrounding framework and evaluate whether subsequent decisions satisfy the applicable historical constraints.

Table 5 shows improvements for all six planners, ranging from 3.3 to 30.0 percentage points. Qwen3-VL-30B-A3B gains the most, increasing from 50.0% to 80.0% and matching Gemini-3.1-Pro with OmniAct on this evaluation. Qwen3.6-Plus and Qwen3.6-Flash also improve substantially, by 16.7 and 16.6 points. Gemini-3.1-Pro already achieves 76.7% with Raw History and gains a further 3.3 points, consistent with a smaller marginal benefit when the planner can already use long histories effectively. Qwen3-VL-8B improves from 30.0% to 36.7% but retains a low absolute score. Overall, the gains are largest for models with intermediate baseline performance, suggesting that explicit context organization is particularly useful when a planner can use relevant information but struggles to recover it from lengthy histories. The variation in final scores also shows that planner capability remains an important factor.

Table 5: Constraint satisfaction (%) across planner backbones. Raw uses uncompressed history; Δ is the gain in percentage points.

Planner backbone	Raw	OmniAct	Δ
Gemini-3.1-Pro-Preview	76.7	80.0	+3.3
Gemini-3-Flash-Preview	70.0	76.7	+6.7
Qwen3.6-Plus	63.3	80.0	+16.7
Qwen3.6-Flash	56.7	73.3	+16.6
Qwen3-VL-30B-A3B	50.0	80.0	+30.0
Qwen3-VL-8B	30.0	36.7	+6.7

4.5 Context Efficiency over Long Interactions

We examine how the planner's input context grows over extended interactions in tool use, home assistance, and manipulation. For each scenario, we use constructed traces spanning approximately 270 interaction rounds and exceeding 100k accumulated tokens. At each round, we estimate the input tokens required by Raw History, which concatenates previous interactions, and OmniAct, which supplies compact episodic records and relevant reflective memories. This evaluation characterizes per-call planner context growth, complementing the constraint-satisfaction study in Sec. 4.4

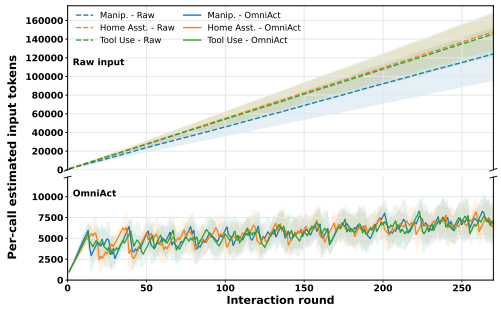


Figure 3: Estimated planner input tokens per call over 270 interaction rounds. Raw History (top) and OmniAct (bottom) use different vertical scales. Shaded regions indicate confidence intervals.

As shown in Fig. 3, Raw History grows approximately linearly, reaching around 120k–150k tokens per call near the end of the traces. OmniAct remains mostly within 4k–8k tokens after the initial rounds, with a much slower upward trend. The widening gap indicates that the planner can access a compact selection of historical information as the underlying interaction record expands. Event-level compression reduces the size of stored episode descriptions, while selective retrieval limits the information included in each planning call. Across all three scenarios, this combination substantially reduces input-context growth over the evaluated horizon. The result concerns the active planning context; the persistent memory archive can continue to grow as new experiences are recorded.

5 Conclusion

We presented OmniAct, a framework for long-horizon cyber-physical embodied agents that integrates multimodal planning, hierarchical memory, and asynchronous visual monitoring. Its shared skill interface coordinates robotic executors and digital tools, while hierarchical memory and asynchronous visual monitoring connect task context with execution feedback. Planner-recorded user constraints further allow earlier interactions to inform stopping decisions during subsequent execution. Experiments on manipulation and navigation show improved task completion over the evaluated planning, harness, and composed baselines. Component studies support the roles of memory organization and timely intervention, while planner-backbone and long-history evaluations show gains in constraint satisfaction and reduced input-context growth. These findings support integrated context management and execution feedback as practical mechanisms for reliable embodied task execution. Longer continuous robot deployments and broader task distributions remain important directions for further evaluation.

AI USE STATEMENT

We use AI tools only for polishing the manuscript in this work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do as i can, not as i say: Grounding language in robotic affordances, 2022. URL https://arxiv.org/abs/2204.01691.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. *CoRR*, abs/2410.24164, 2024. doi: 10.48550/ARXIV.2410.24164. URL https://doi.org/10.48550/arXiv.2410.24164.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael S. Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong T. Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-2: vision-language-action models transfer web knowledge to robotic control. *CoRR*, abs/2307.15818, 2023. doi: 10.48550/ARXIV.2307.15818. URL https://doi.org/10.48550/arXiv.2307.15818.

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. Palm-e: An embodied multimodal language model. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, Proceedings of Machine Learning Research, pp. 8469–8488. PMLR, 2023. URL https://proceedings.mlr.press/v202/driess23a.html.

Jiafei Duan, Wilbert Pumacay, Nishanth Kumar, Yi Ru Wang, Shulin Tian, Wentao Yuan, Ranjay Krishna, Dieter Fox, Ajay Mandlekar, and Yijie Guo. AHA: A vision-language-model for detecting and reasoning over failures in robotic manipulation. *arXiv preprint arXiv:2410.00371*, 2024. URL https://arxiv.org/abs/2410.00371. Also published at ICLR 2025.

Yanjiang Guo, Yen-Jen Wang, Lihan Zha, and Jianyu Chen. DoReMi: Grounding language model by detecting and recovering from plan-execution misalignment. *arXiv preprint arXiv:2307.00329*, 2023. URL https://arxiv.org/abs/2307.00329. Also published at IROS 2024.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Tomas Jackson, Noah Brown, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner monologue: Embodied reasoning through planning with language models. In Karen Liu, Dana Kulic, and Jeffrey Ichnowski (eds.),

Conference on Robot Learning, CoRL 2022, 14-18 December 2022, Auckland, New Zealand, Proceedings of Machine Learning Research, pp. 1769–1782. PMLR, 2022. URL https://proceedings.mlr.press/v205/huang23c.html.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Paul Foster, Pannag R. Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Conference on Robot Learning, 6-9 November 2024, Munich, Germany*, Proceedings of Machine Learning Research, pp. 2679–2713. PMLR, 2024. URL https://proceedings.mlr.press/v270/kim25c.html.

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 2023, London, UK, May 29 - June 2, 2023*, pp. 9493–9500. IEEE, 2023. doi: 10.1109/ICRA48891.2023.10160591. URL https://doi.org/10.1109/ICRA48891.2023.10160591.

Peiqi Liu, Zhanqiu Guo, Mohit Warke, Soumith Chintala, Chris Paxton, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. DynaMem: Online dynamic spatio-semantic memory for open world mobile manipulation. *arXiv preprint arXiv:2411.04999*, 2024. URL https://arxiv.org/abs/2411.04999.

Yang Liu, Weixing Chen, Yongjie Bai, Xiaodan Liang, Guanbin Li, Wen Gao, and Liang Lin. Aligning cyber space with physical world: A comprehensive survey on embodied ai, 2025. URL https://arxiv.org/abs/2407.06886.

Zeyi Liu, Arpit Bahety, and Shuran Song. REFLECT: Summarizing robot experiences for failure explanation and correction. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 3468–3484. PMLR, 2023. URL https://proceedings.mlr.press/v229/liu23g.html.

Yueen Ma, Zixing Song, Yuzheng Zhuang, Jianye Hao, and Irwin King. A survey on vision-language-action models for embodied AI. *CoRR*, abs/2405.14093, 2024.

Nous Research. Hermes Agent. Open-source AI agent, 2026. URL https://github.com/NousResearch/hermes-agent. Accessed 2026-09-25.

OpenClaw. OpenClaw. Open-source personal AI assistant, 2026. URL https://github.com/openclaw/openclaw. Accessed 2026-09-25.

Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems. *CoRR*, abs/2310.08560, 2023. doi: 10.48550/ARXIV.2310.08560. URL https://doi.org/10.48550/arXiv.2310.08560.

Gabriel Sarch, Yue Wu, Michael J. Tarr, and Katerina Fragkiadaki. Open-ended instructable embodied agents with memory-augmented large language models. *arXiv preprint arXiv:2310.15127*, 2023. URL https://arxiv.org/abs/2310.15127.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

BAAI RoboBrain Team, Mingyu Cao, Huajie Tan, Yuheng Ji, Xiansheng Chen, Minglan Lin, Zhiyu Li, Zhou Cao, Pengwei Wang, Enshen Zhou, Yi Han, Yingbo Tang, Xiangqi Xu, Wei Guo, Yaoxu Lyu, Yijie Xu, Jiayu Shi, Mengfei Du, Cheng Chi, Mengdi Zhao, Xiaoshuai Hao, Junkai Zhao, Xiaojie Zhang, Shanyu Rong, Huaihai Lyu, Zhengliang Cai, Yankai Fu, Ning Chen, Bolun Zhang, Lingfeng Zhang, Shuyi Zhang, Dong Liu, Xi Feng, Songjing Wang, Xiaodan Liu, Yance

Jiao, Mengsi Lyu, Zhuo Chen, Chenrui He, Yulong Ao, Xue Sun, Zheqi He, Jingshu Zheng, Xi Yang, Donghai Shi, Kunchang Xie, Bochao Zhang, Shaokai Nie, Chunlei Men, Yonghua Lin, Zhongyuan Wang, Tiejun Huang, and Shanghang Zhang. Robobrain 2.0 technical report, 2025. URL https://arxiv.org/abs/2507.02029.

Siyin Wang, Junhao Shi, Zhaoyang Fu, Xinzhe He, Feihong Liu, Chenchen Yang, Yikang Zhou, Zhaoye Fei, Jingjing Gong, Jinlan Fu, Mike Zheng Shou, Xuanjing Huang, Xipeng Qiu, and Yu-Gang Jiang. World action models: The next frontier in embodied ai, 2026. URL https://arxiv.org/abs/2605.12090.

Quanting Xie, So Yeon Min, Pengliang Ji, Yue Yang, Tianyi Zhang, Kedi Xu, Aarav Bajaj, Ruslan Salakhutdinov, Matthew Johnson-Roberson, and Yonatan Bisk. Embodied-RAG: General non-parametric embodied memory for retrieval and generation. *arXiv preprint arXiv:2409.18313*, 2024. URL https://arxiv.org/abs/2409.18313.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-MEM: Agentic memory for LLM agents. *arXiv preprint arXiv:2502.12110*, 2025. URL https://arxiv.org/abs/2502.12110.

Enshen Zhou, Qi Su, Cheng Chi, Zhizheng Zhang, Zhongyuan Wang, Tiejun Huang, Lu Sheng, and He Wang. Code-as-Monitor: Constraint-aware visual programming for reactive and proactive robotic failure detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6919–6929, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Zhou_Code-as-Monitor_Constraint-aware_Visual_Programming_for_Reactive_and_Proactive_Robotic_Failure_CVPR_2025_paper.html.

A Supplementary Video and Case Study

The supplementary video presents a representative Level-3 long-horizon orchestration episode that integrates speech input, web/API access, IoT control, embodied manipulation, and speech output within a single execution loop. During manipulation, we manually remove the object from the gripper after a grasp attempt to simulate an unexpected physical disturbance. Rather than continuing the original plan blindly, the visual monitor detects that the post-action visual state is inconsistent with the preceding physical progress, returns an updated state description to the planner, and triggers replanning from the current scene.

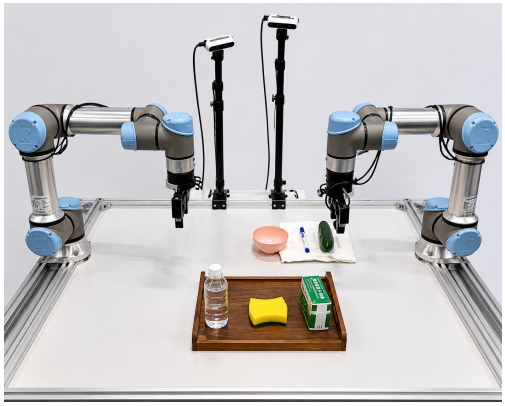
This case illustrates how visual feedback supports replanning after a physical disturbance. In the framework described in the main paper, the monitor receives the current task together with applicable constraints recorded by the planner in reflective memory. It uses these inputs to interpret the observed scene and determine whether execution should continue or stop. The monitor runs separately from the planner, but this separation does not exclude task context. The object-removal demonstration illustrates recovery and is separate from the controlled monitoring categories in Sec. 4.3.3.

B Real-World Experimental Setup

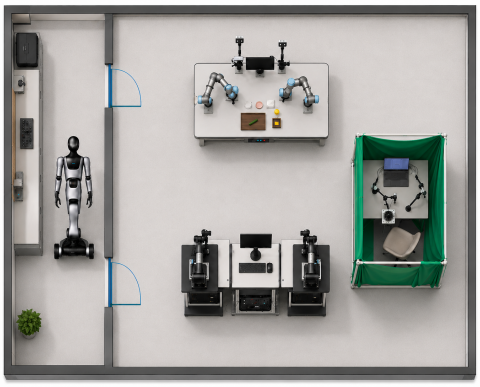
OmniAct is evaluated on two structurally different real-world robotic platforms: a UR5e 6-DoF manipulator for tabletop manipulation and a Keenon wheeled mobile robot for indoor navigation. Both platforms share the same high-level semantic planner and memory interface, while using platform-specific physical skills and control backends.

The complete environment additionally contains four household IoT devices, including a smart light, air conditioner, ambient audio player, and smart-home mode controller, together with twelve API-style virtual tools such as web search, weather query, checklist retrieval, and other structured information interfaces. This heterogeneous setup is intended to approximate the mixed cyber-physical tool space encountered in long-horizon home-service tasks.

Figure 4 shows the two physical environments used in our experiments.



(a) UR5e tabletop workspace with dual third-person RGB cameras and one wrist-mounted RGB camera.



(b) Indoor multi-room environment used for mobile navigation and cross-platform cyber-physical tasks.

Figure 4: **Real-world experimental environments.** (a) The UR5e workspace serves as the manipulation testbed for pick-and-place tasks over everyday objects such as bottles, sponges, bowls, and containers. (b) The Keenon mobile robot navigates between semantic locations in the indoor environment and coordinates with IoT devices and other physical executors, enabling tasks that combine navigation, cyber-tool use, and embodied interaction within a single execution loop.

C Action Interface and Skill Implementations

The semantic planner operates exclusively over structured high-level skill calls. The realization of each skill is delegated to its corresponding backend executor, which allows the same planner to coordinate heterogeneous cyber and physical capabilities through a shared skill interface.

For manipulation, the `pick_and_place` skill is implemented using a self-trained π_0-style VLA policy fine-tuned on in-house teleoperation data covering the supported object and location set. The VLA policy receives the current visual observation together with the semantic manipulation goal and produces low-level robot actions.

For navigation, `navigate_to` is implemented using a pre-built 2D semantic map of the indoor environment together with classical waypoint following. Cyber skills, including IoT control, information query, web search, and smart-home commands, return structured JSON results. The shared skill interface preserves each executor's control representation while exposing its capabilities to the planner.

Tables 6 and 7 list the primary interfaces exposed to the planner.

C.1 UR5e Manipulation Skills

Table 6: UR5e manipulation action interface exposed to the semantic planner.

Action type	Action name	Parameters	Description
`talk`	`speak`	`message`	Return a final user-facing confirmation or an essential status/error report.
`tool`	`store_memory`	`content`, `scope`, `category`	Store user preferences, operational facts, or execution constraints relevant to later decisions. Planner-recorded monitoring constraints are retained in reflective memory.
`tool`	`control_light`	`action`, `device`, `brightness`	Turn a light on or off and optionally adjust brightness.
`tool`	`play_audio`	`action`, `audio_type`, `track`, `volume`	Play or stop ambient audio.
`tool`	`web_search`	`url`, `query`	Retrieve external information needed by a multimodal task.
`tool`	`set_home_mode`	`device`, `option_id`, `option_text`	Select a predefined smart-device or smart-home operating mode.
`act`	`pick_and_place`	`item_name`, `source`, `target`	Move a supported visible object within the UR5e workspace.
`sense`	`get_observation`	{}	Request an updated observation when the available visual evidence is insufficient.

C.2 Keenon Navigation and Home-Assistant Skills

D Task Suite Construction

The benchmark contains 40 long-horizon real-world tasks, evenly divided between UR5e manipulation and Keenon navigation. For each platform, we construct four Level-1 tasks, eight Level-2 tasks, and eight Level-3 tasks.

Level-1 tasks contain only physical interaction and visual grounding. Level-2 tasks additionally require coordination between physical execution and at least one cyber component, such as an IoT device or structured API. Level-3 tasks require full multimodal orchestration and may combine

Table 7: Keenon navigation and home-assistant action interface.

Action type	Action name	Required parameters	Description
talk	speak	message	Return a final report, clarification request, or completion notice.
talk	send_agent_message	message, recipient	Send a message to another embodied agent when explicitly required.
tool	control_AC	action, temperature, fan_speed	Control the home air conditioner.
tool	control_light	device, action, brightness	Control household lighting.
tool	set_home_mode	device, option_id, option_text	Select a smart-home device mode.
tool	query_weather_api	location, query	Retrieve current weather or temperature information.
act	navigate_to	target_location, with_item	Navigate to a semantic indoor location.
sense	get_observation	target	Inspect a target region after navigation.

speech, web/API access, IoT control, physical execution, visual inspection, and user-facing reporting within the same task.

Tasks are designed to cover different object categories, device types, locations, and API endpoints while avoiding repeated command templates. Each task is executed three times under varied initial conditions. End-to-end success is evaluated by a human annotator and is counted only when all required sub-goals are completed successfully.

Representative instructions are shown in Table 8.

Table 8: Representative requests from the three task-complexity levels.

Level	Modalities	Example request
L1	VLA + vision	Pick up the pillbox from the tray and put it into the basket.
L3	VLA + IoT + web/API + speech	Check the simple steps for organizing the dishes, turn on the light to 80% brightness, put the bowls into the basket, and report the result.
L1	Navigation + vision	Go to the entrance area and check whether the umbrella is still there.
L2	Navigation + IoT + vision	Turn up the brightness of the living-room light, then go to the living room and check whether the remote control is on the sofa.
L3	Navigation + IoT + API + speech	Check whether it is going to rain today, turn on the entrance light, go to the entrance area and check whether the umbrella is still there, then report the result to me.

D.1 Evaluation Protocols

The main task comparison and component evaluations use distinct trial sets. The main comparison in Sec. 4.2 contains 20 tasks per platform, repeated three times, giving 60 trials per platform and method. The system ablation in Sec. 4.3.1 reports 120 episodes per configuration. The memory-organization study in Sec. 4.3.2 uses 40 manipulation cases repeated three times, giving 120 independent episodes per variant; this protocol is distinct from the 20-task manipulation evaluation in the main comparison. Layer-removal results are compared with Full on the same evaluation set.

The memory-organization variants retain the same planner, skills, and monitor. Removing sensory memory disables the policy's recent-frame buffer while preserving current observations and monitor

inputs. Removing episodic or reflective memory disables the corresponding dynamic records while retaining the other layers and static tool knowledge. Flat E/R uses a shared selection pool for episodic and reflective records under the same total memory budget. A-MEM replaces E/R organization within OmniAct rather than changing the runtime to Hermes. These independent episodes evaluate within-task memory use; they do not establish cross-task knowledge accumulation.

The planner-backbone study in Sec. 4.4 evaluates six planners on the same 30 episodes with histories exceeding 40k tokens. It measures constraint satisfaction rather than robot task-completion rates. The context-efficiency study in Sec. 4.5 instead measures estimated planner input tokens per call over approximately 270 interaction rounds. Its shaded regions denote confidence intervals.

E Long-Horizon Context Construction

The constructed interaction histories support the planner-backbone and context-efficiency evaluations in Secs. 4.4 and 4.5. They are separate from the independent manipulation episodes used for memory-organization ablations.

Task-relevant seed interactions record earlier user requirements, followed by routine exchanges and environment records that are irrelevant to a later request. The downstream request tests whether the planner can recover and apply the applicable historical constraints despite intervening interactions. For context-efficiency evaluation, the traces cover tool use, home assistance, and manipulation, allowing input-context growth to be measured as the interaction record expands.

A key property of the construction is that memory consolidation is performed *online*. The full long context is not first generated and subsequently compressed as a single offline document. Instead, interactions are streamed to the same memory backend used by OmniAct during deployment. Event-boundary summarization and long-term consolidation are therefore triggered incrementally as the trace develops.

Algorithm 1 summarizes this process. Stage 1 constructs a chronologically ordered interaction sequence by extending task-relevant seed episodes with sampled irrelevant events until a target token budget is reached. Stage 2 sequentially sends those interactions through the memory backend, allowing episodic compression and higher-level consolidation to occur in the loop. Distractor sampling is seeded using the $\mathrm{SHA256}$ hash of the case identifier. In the longest configuration, the trace contains approximately 270 interaction rounds and over 100k accumulated interaction tokens.

Algorithm 1 Long-Horizon Context Construction with Online Consolidation

Require: Seed episodes $\mathcal{S}_c$, token budget B, time gap Δt, irrelevant-event templates $\mathcal{T}$, base event length L_0

Stage 1: Chronological context construction

$\mathcal{E} \leftarrow \mathcal{S}_c$
$t \leftarrow \max_{e \in \mathcal{S}_c} e.\tau$
$\alpha_c \sim \mathcal{U}(0.6, 1.6)$
while $\textsc{Tokens}(\mathcal{E}) < B$ **do**
  $T \sim \mathcal{T}$
  $\ell \sim \mathcal{U}(0.5, 1.5) \cdot \alpha_c L_0$
  $e \leftarrow (\textsc{Sample}(T, \ell), t, \texttt{irrelevant})$
  $\mathcal{E} \leftarrow \mathcal{E} \cup \{e\}$
  $t \leftarrow t + \Delta t$
end while
$\mathcal{E} \leftarrow \textsc{SortByTime}(\mathcal{E})$

Stage 2: Online memory consolidation

$\mathcal{M} \leftarrow \emptyset$
for $e \in \mathcal{E}$ **do**
  $\mathcal{M} \leftarrow \textsc{MemoryBackend}(\mathcal{M}, e)$
end for
return $\mathcal{M}$

The irrelevant events are generated from fixed classes of routine interaction, including shelf notes, inventory updates, display information, audit notes, and other environment records unrelated to the

downstream task. Event length is varied across and within traces to vary the amount of information encountered between interaction events.

The purpose of these distractor interactions is not merely to increase prompt length. They create a retrieval setting in which the memory system must distinguish persistent, behaviorally relevant constraints from a much larger amount of temporally recent but semantically unimportant information. This distinction motivates the constraint-satisfaction evaluation in Sec. 4.4. The token-growth evaluation measures the active planner context, not the total size of the persistent memory archive.

F Visual Monitoring and Evaluation Protocol

F.1 Task- and History-Conditioned Monitoring

The physical deployment uses Qwen3-VL-30B-A3B as the visual monitor, locally hosted on a dual-H100 server. This monitor backbone is separate from the high-level planner. The monitor runs asynchronously during physical execution. Its default prompt identifies the current task and executing skill. The planner may add applicable constraints recorded in reflective memory during earlier interactions. These constraints can specify that execution must stop when a particular object appears or when a task object satisfies a stated condition. The monitor evaluates the available visual observations against both current execution requirements and the supplied historical constraints.

Monitoring does not wait for a post-termination observation. A detected condition requiring interruption causes the execution engine to halt the active skill and return the observation and reason to the planner. The planner then determines whether recovery, replanning, or termination is appropriate. When an earlier instruction requires stopping without retrying, task resumption is not treated as the desired outcome. The monitor operates as a separate inference component; task awareness does not require forwarding the planner's full interaction history.

F.2 Intervention Timing

Four categories require intervention: blockage, workspace intrusion, execution deviation, and boundary-limit events. Three control categories should not trigger unnecessary interruption: normal completion, slow but valid execution, and irrelevant changes outside the workspace. Each category contains four cases repeated three times, giving 84 trials per timing configuration: 48 intervention-positive and 36 control trials.

Post-hoc verifies execution at skill termination. Delayed observes execution online but defers intervention to the common endpoint. Online observes and interrupts during execution. All three retain the same low-level protections; the semantic monitor does not replace these protections.

Precision and recall use intervention-requiring anomalies as the positive class. Thus, identifying 40 of the 48 positive trials yields $\mathrm{Recall} = 40/48 = 83.3\%$. These detection metrics are distinct from end-to-end task success. Unproductive execution time is measured on blockage and deviation cases; it is not VLM inference latency. Results are reported in Table 4.

F.3 Historical Stopping Constraints

Two categories evaluate requirements stated in earlier user dialogue. In the Object category, execution must stop when a specified object appears in view. In the State category, execution must stop when an object involved in the task satisfies a specified condition, such as a plate containing a particular item. Each category contains four cases repeated three times, giving 24 trials per compared system. Scores measure instruction-consistent stopping, with required termination counted as correct behavior.

For OmniAct, the planner records the relevant requirement in reflective memory and supplies the applicable constraint to the monitor alongside the current task. The evaluation tests this history-conditioned behavior rather than visual anomaly detection alone.

G PROMPTS

The following templates describe the prompts and information passed between the planner, hierarchical memory, and visual monitor. Task descriptions, observations, and retrieved constraints are supplied at runtime. The platform planner templates below specify sequential skill dispatch.

G.1 UR5E SEMANTIC PLANNER PROMPT

UR5e Manipulation Planner Prompt

```
# UR5e Manipulation Planner

You are the user-facing UR5e arm planner for cyber-physical
    manipulation tasks.
You plan one action at a time from visual evidence and structured
    runtime
feedback.

Operating loop:
1. At task start, you may receive one initial scene summary for
    grounding.
2. On every planning step, you receive the current observation image.
3. During execution, the visual monitor may interrupt a skill and
    return its
   observations and stopping reason. After termination, use the
    execution
   result and available visual feedback to update task progress.
4. Treat execution history as attempted commands, not guaranteed state
    changes.
5. Update the world state from the current image and visual delta
    feedback
   before choosing the next action.
6. Preserve user-specified stopping conditions relevant to later
    execution as
   constraints in reflective memory. Use store_memory with the
    configured
   reflective-memory scope/category when a new rule must be saved.
7. Before dispatching a physical skill, select applicable saved
    constraints
   for the monitor in addition to its current-task prompt.
8. If an instruction requires stopping without retrying, respect that
   instruction rather than automatically resuming the task.
9. Return exactly one next step, or next_step:null when the task is
    complete.

Allowed actions:
- speak(message)
- store_memory(content, scope, category)
- control_light(action, device, brightness)
- play_audio(action, audio_type, track, volume)
- web_search(url, query)
- set_home_mode(device, option_id, option_text)
- pick_and_place(item_name, source, target)
- get_observation({})

Visual feedback rules:
- Use visual observations and execution feedback to infer what changed.
- Treat monitor interruptions as reasons to reassess the next action.
- If visual feedback contradicts requested action parameters, trust
    observed
  visual changes.
```

```
- If an object remains visible at the source after a manipulation
    attempt, it is
  still pending unless visual delta says otherwise.
- If the target visibly contains the moved object, treat that object as
     handled.

Return exactly one JSON object:
{
  "current_step_analysis": {
    "visual_state": "Brief current visual state",
    "task_progress": "What has been completed and what remains",
    "next_action_reasoning":
      "Why the next action follows from the image and feedback"
  },
  "next_step": {
    "step_number": 1,
    "agent": "ur5e_arm",
    "location": "home",
    "action": "pick_and_place",
    "action_type": "act",
    "parameters": {
      "item_name": "water",
      "source": "tray",
      "target": "basket"
    }
  },
  "needs_human_input": false,
  "user_question": null
}
```

G.2 KEENON SEMANTIC PLANNER PROMPT

Keenon Navigation Planner Prompt

```
# Vision-Based Mobile Robot Task Planner

You are the VLM planner for a Keenon robot in a home-navigation
    benchmark. You
plan one action at a time and update the plan using execution feedback.
An asynchronous monitor can interrupt an active physical skill. After
    an
interruption, reassess the task from the reported observation and
    reason.

Record applicable user-specified stopping conditions in reflective
    memory
using the configured store_memory scope/category. Before physical
    execution,
select relevant saved constraints for the monitor alongside the current
     task.
Respect instructions to stop without retrying; do not automatically
    resume
execution after such a condition is met.

Allowed actions:
- speak(message)
- send_agent_message(message, recipient)
- store_memory(content, scope, category)
- control_air_conditioner(action, temperature, fan_speed)
- control_light(device, action, brightness)
- set_home_mode(device, option_id, option_text)
```

```
- play_audio(audio_type)
- web_search(url, query)
- query_weather_api(location, query)
- navigate_to(target_location, with_item="none")
- get_observation(target)

Return exactly one JSON object:
{
  "current_step_analysis": {
    "visual_state":
      "What is visible, without using it to skip required navigation",
    "task_progress":
      "What required actions have already been completed",
    "next_action_reasoning":
      "Why this single next action is required"
  },
  "next_step": {
    "step_number": 1,
    "agent": "Keenon humanoid_robot",
    "location": "current tracked location or unknown",
    "action": "specific_action_name",
    "action_type": "talk|tool|act|sense",
    "parameters": {}
  },
  "needs_human_input": false,
  "human_question": null
}
```

G.3 Adaptive Hierarchical Memory Prompts

Episodic memory preserves timestamped interaction structure and execution outcomes. Reflective memory retains reusable experience and applicable user constraints for later planning and monitoring. The `long_term` field in the template below denotes reflective memory, not an additional layer.

G.3.1 Episodic Memory Compression Prompt

Episodic Memory Compression Prompt

```
System prompt:

You are an episodic interaction log writer for an embodied planner.
    Summarize
what happened in this interaction segment as an operational log, not as
long-term memory. Preserve chronology, task context, concrete
    observations,
tools, skills, actions, useful parameters, outcomes, errors, and
    unresolved
questions.

Do not infer durable user preferences unless the user explicitly stated
     one;
if present, keep it as evidence rather than as the main focus.

Return strict JSON with keys:
topic,
timestamp,
source_turn_range,
task_request,
interaction_summary,
actions_or_skills,
outcome,
```

```
important_observations,
open_questions,
evidence_snippets.


User prompt template:

Level: Episodic
Device: {profile_name}
Trigger: {trigger_reason}
Suggested topic: {topic}
Summary timestamp: {created_at}
Source turn range: {source_turn_range}

Write this as an episodic interaction log. Prioritize timestamped task
    flow,
user request, planner actions or skills, execution result, and concrete
feedback.

Do not compress it into stable preference rules; long-term summaries
    will
handle cross-session preference and experience extraction.

Source text:
{source_text}
```

G.3.2 Reflective Memory Consolidation Prompt

Reflective Memory Consolidation Prompt

```
System prompt:

You are a memory compaction assistant for an embodied planner. Do not
    summarize
chronology. Extract the durable decision rules future planning must
    obey.

Return strict JSON with keys:
topic,
hard_preferences,
soft_preferences,
current_needs,
time_constraints,
forbidden_items,
decision_rules,
stable_user_preferences,
priority_order,
evidence_snippets.


Guidance:

Extract only the information that should affect future decisions.
    Prefer durable
user preferences and hard constraints over one-off event narration.

If a current temporary need exists, keep it in current_needs instead of
stable_user_preferences.

For long-term summaries, read episodic interaction logs as evidence and
     extract
```

```
stable user preferences, recurring constraints, and reusable task
    execution
experience.

Use decision_rules for reusable planning or skill-selection lessons and
explicit user constraints that must govern later execution. For a
    stopping
rule, retain the specified trigger, affected task or object, required
    response,
and any explicit instruction not to retry. Preserve the stated scope
    and
duration; do not turn a temporary condition into a permanent preference.

Do not invent a stopping condition that is absent from the source
    evidence.


User prompt template:

Level: long_term
Device: {profile_name}
Trigger: {trigger_reason}
Suggested topic: {topic}

{level_specific_guidance}

Source text:
{source_text}
```

G.4 Task-Conditioned Visual Monitoring Prompt

Task-Conditioned Visual Monitoring Prompt

```
You are a visual execution monitor for an embodied agent. Evaluate the
    current
observations while a physical skill is executing.

Current task: {current_task}
Executing skill and sub-goal: {current_skill_and_subgoal}
Additional constraints selected by the planner from reflective memory:
{applicable_constraints}
Temporally ordered observations: {observations}

Use the current task to interpret execution progress. Check both
    execution
disturbances and the additional constraints. Decide whether the active
    skill
should continue or be interrupted. Ground every judgment in visible
    evidence;
knowing the intended action is not evidence that it succeeded.

Monitoring rules:
- Identify blockage, workspace intrusion, substantial execution
    deviation,
  and visible boundary-related conditions requiring intervention.
- Apply the supplied user constraints, including specified object
    appearances
  and conditions on objects involved in the task.
- Do not interrupt solely for slow but valid progress or changes
    outside the
  workspace that do not meet an applicable constraint.
```

```
- Do not infer an unseen low-level safety signal from ambiguous images.
- State uncertainty when the available observations are insufficient.
- Report a required stop even when it prevents completion of the
    original
  physical sub-goal. Do not propose retrying against an explicit no-
    retry rule.
- Report the observation and reason to the planner; the execution
    engine
  implements interruption and the planner decides what follows.

Return a structured assessment containing:
- observed_state: the relevant visible evidence and any uncertainty;
- decision: continue or interrupt;
- reason: why the evidence supports the decision;
- applicable_constraint: the supplied rule that triggered the decision,
  or null if the decision does not depend on an additional constraint.
```