Hybrid Self-evolving Structured Memory for GUI Agents

Anonymous ACL submission

Abstract

The remarkable progress of vision–language models (VLMs) has enabled GUI agents to interact with computers in a human-like manner. Yet real-world computer-use tasks remain difficult due to long-horizon workflows, diverse interfaces, and frequent intermediate errors. Prior work equips agents with external memory built from large collections of trajectories, but relies on flat retrieval over discrete summaries or continuous embeddings, falling short of the structured organization and self-evolving characteristics of human memory. Inspired by the brain, we propose **Hybrid Self-evolving Structured Memory (HyMEM)**, a graph-based memory that couples discrete high-level symbolic nodes with continuous trajectory embeddings. HyMEM maintains a graph structure to support multi-hop retrieval, self-evolution via node update operations, and on-the-fly working-memory refreshing during inference. Extensive experiments show that HyMEM consistently improves open-source GUI agents, enabling 7B/8B backbones to match or surpass strong closed-source models; notably, it boosts Qwen2.5-VL-7B by +22.5% and outperforms Gemini2.5-Pro-Vision and GPT-4o. Our code and data will be publicly released.

1 Introduction

Benefiting from the strong perception and planning capabilities of large vision–language models (VLMs), VLM-based graphical user interface (GUI) agents have emerged as a promising direction for automating interactions on desktop and mobile platforms. To follow user instructions, these agents should interpret visual and linguistic context, reason and plan over dynamic layouts, predict and execute precise actions (*e.g.,* clicking and typing) (Qin et al., 2025; Hong et al., 2023). However, the long-horizon and highly diverse nature of real-world computer-use tasks still makes human-level performance challenging: current GUI agents often fail due to non-trivial intermediate errors or overlooking critical conditions (Yang et al., 2025).

To mitigate these limitations, recent work draws inspiration from the human brain's memory mechanisms and equips GUI agents with external memory modules (Wu et al., 2025b; Zhang et al., 2023). Humans can automatically store, organize, and update experiences—whether observed from others or acquired firsthand—and retrieve relevant memories as references when needed (Williams et al., 2022). Similarly, existing approaches collect an agent's past trajectories into a database (*i.e.,* an experience memory), where each item's multimodal content is either distilled into discrete tokens (*e.g.,* sentences or key phrases) (Ouyang et al., 2025) or compressed into continuous embeddings (Wu et al., 2025a). At inference time, the agent retrieves relevant items via similarity matching and uses them to construct complementary context as a working memory for the current instruction.

Despite these advances, current agent memory designs remain far weaker than human memory, particularly in how they organize knowledge and evolve it over time. The brain can preserve fine-grained evidence for reconstructing multimodal episodes, associate them with higher-level concepts or strategies for efficient search and reasoning, and continuously update as new experiences arrive. Neurobiological evidence suggests that human memory operates as a hybrid system: the hippocampus encodes rich multimodal experiences into continuous latent representations (Spens and Burgess, 2024a), while the neocortex extracts statistical regularities and forms higher-level discrete symbols (McClelland et al., 1995). New experiences are then linked to existing knowledge and only the novel information needs to be stored (Spens and Burgess, 2024b). This structured organization supports long-term retention of massive multimodal details while enabling efficient retrieval, reasoning, and continual evolution.

Method	Data Form	MM	Struct.	Global Up.	Local Up.
MemGPT	Discrete	✗	✗	✗	✓
ExpeL	Discrete	✗	✗	✓	✗
ReasoningBank	Discrete	✗	✗	✓	✗
AWM	Discrete	✗	✗	✓	✗
A-MEM	Discrete	✗	✓	✓	✗
G-Memory	Discrete	✗	✓	✓	✗
CoMEM	Continuous	✓	✗	✗	✗
HYMEM	**Hybrid**	✓	✓	✓	✓

Table 1: Comparison of Memory Paradigms. Abbreviations: **MM**: Multimodal support; **Struct.**: Structured organization; **Up.**: Update. Methods are compared by data form, MM, and structure, as well as Global (self-evolution) and Local (working memory refresh) updates.

Inspired by the human brain, we aim to build a hybrid structured external memory that organizes knowledge more effectively and supports flexible updates and self-evolution over time. Concretely, we first construct a hybrid graph-based memory from agent trajectories by merging trajectories into higher-level nodes with discrete symbolic semantics, while continuous embeddings of the trajectory are used to preserve fine-grained multimodal evidence. Building on this structure, we design a graph evolution and update mechanism that can add, update, and replace nodes as new trajectories arrive, enabling the memory to accumulate knowledge without uncontrolled growth. During inference, we further introduce an on-the-fly working-memory update that detects execution phase shifts and triggers re-retrieval and refreshing, allowing multi-turn interaction between the agent and the memory database throughout long-horizon tasks.

Based on these designs, we propose **Hybrid Self-evolving Structured Memory (HYMEM)**, a graph-based external memory that tightly couples symbolic guidance with multimodal experience to improve planning and grounding for GUI agents. Empirically, HYMEM consistently boosts open-source GUI agents across multiple benchmarks, and can enable lightweight 7B/8B-scale backbones to match or surpass strong closed-source models. For example, it improves Qwen2.5-VL 7B (Bai et al., 2025b) by +22.5% (12.5% to 35.0%), exceeding Gemini2.5-Pro-Vision (Google, 2023) by 5.4%, and GPT-4o (OpenAI, 2024) by 15.3%. Similar gains are observed on UI-TARS-1.5-7B (Qin et al., 2025) and Qwen3-VL-8B (Bai et al., 2025a), demonstrating HYMEM as a cost-effective path to state-of-the-art GUI agents.

2 Related Work

GUI Agents. Early GUI agents primarily relied on HTML or DOM parsing to ground actions (Deng et al., 2023; Zheng et al., 2024), but the field has rapidly shifted toward Vision-Language Models (VLMs) capable of perceiving screens visually (Hong et al., 2023; Qin et al., 2025). While autonomous frameworks like AppAgent (Zhang et al., 2023) integrate these vision experts into closed-loop systems, maintaining consistency across long-horizon, open-ended tasks remains a significant challenge. This bottleneck has driven recent research toward equipping agents with external memory systems (Hu et al., 2025b), allowing them to retain long-term context and generalize from historical interactions rather than relying solely on immediate perception (Hu et al., 2025a).

Language Model Memory. Current memory mechanisms generally fall into discrete, continuous, or graph-based paradigms. Discrete approaches log experiences as textual summaries (Packer et al., 2024) or reasoning traces (Ouyang et al., 2025); while interpretable, they often fail to capture the fine-grained visual nuances of GUI elements (Luo et al., 2025). Conversely, continuous memories compress multimodal inputs into dense embeddings (Wu et al., 2025a,b) or latent tokens (Zhang et al., 2025b), preserving sensory details but creating an information bottleneck that hinders explicit reasoning. To improve organization, graph-based systems (Xu et al., 2025; Zhang et al., 2025a) structure knowledge as connected nodes, enabling agents to perform multi-hop retrieval and reasoning beyond similarity matching (Shen et al., 2025).

Hybrid architectures attempt to merge these formats to balance abstraction with detail. However, existing hybrid methods like ExpeL (Zhao et al., 2024) often remain purely token-based, pairing trajectories with textual insights but potentially missing visual fidelity. In contrast, our **HYMEM** organizes discrete textual insights (capturing high-level strategies) and continuous embeddings (preserving fine-grained multimodal details) within a graph structure, ensuring the agent possesses both strategic abstraction and precise perceptual grounding.

3 Methodology

We propose **Hybrid Self-evolving Structured Memory (HYMEM)**, a graph-based external memory for guiding GUI agents. It is a brain-inspired

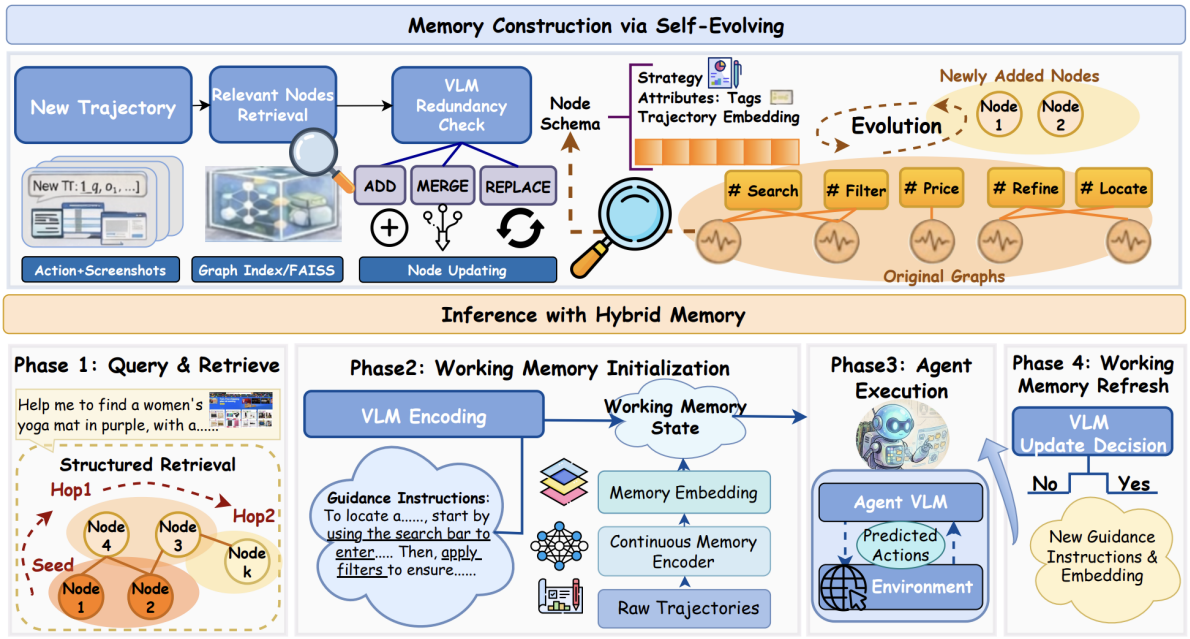

Figure 1: Overview of the **Hybrid Self-Evolving Memory (HYMEM)** system. Top: memory construction via graph evolution, where new trajectories are integrated using retrieval and VLM-based redundancy checks. Bottom: four inference phases—(1) *Query & Retrieve* via structured graph expansion, (2) *Working Memory Initialization* using hybrid encoding, (3) *Agent Execution*, and (4) *On-the-fly Working Memory Refresh* based on new observations.

hybrid memory system with a hippocampal-like Continuous Pathway that encodes trajectories as latent embeddings and a neocortical-like Discrete Pathway that extracts symbolic schemas for high-level concepts. Based on that, HYMEM organizes the memory into an evolving graph, enabling structured search to gather diverse related experiences, and continuously self-evolution for global updates and on-the-fly local refresh during execution.

3.1 Hybrid Structured Memory

Graph Schema. We maintain the memory as an evolving graph $G = (V, E)$, where V and E denote the set of nodes and edges.

The node set V consists of Trajectory Nodes, where each node $v_i \in V$ represents a specific, successful interaction sequence. To support both high-level planning and low-level execution, we define each node as a tuple $v_i = (c_i, A_i, m_i)$:

- High-level Strategy (c_i): A heuristic summarizing the core strategy (*e.g.,* price filter low-to-high).
- Middle-level Attributes (A_i): A semantic tag (*e.g.,* `#search`, `#filter`, `$price`) providing cues about actions, UI elements, and domain concepts.
- Low-level Trajectory Embeddings (m_i): The continuous representations of the trajectory, to preserve fine-grained multimodal details.

We establish undirected edges between trajectory nodes that share identical Middle-level Attributes. This design yields an associative topology that supports multi-hop structured search.

Node Encoding. To guarantee the efficiency and low cost, we leverage lightweight VLMs (*i.e.,* Qwen2.5-VL-7B), for encoding the three types of nodes. For trajectory embedding nodes, we use CoMEM (Wu et al., 2025a,c) to condense the whole trajectory into 8 embeddings that can be directly concatenated into GUI agent embedding layers for reading. For strategy nodes, we prompt a VLM to summarize a concise heuristic describing the high-level strategy behind the trajectory, based on its task description and action sequence. For attribute nodes, we then extract word- and phrase-level semantic tags from the generated strategy using the same VLM. In this way, the strategy and attribute node representations are discrete word tokens, and the trajectory nodes are continuous embeddings.

3.2 Memory Construction via Self-Evolving

We construct the memory in a self-evolving manner: as new trajectories arrive, the memory graph is incrementally refined to add new nodes and update existing structures. To control memory quality, we adopt a three-stage pipeline for adding each new trajectory: retrieving relevant nodes, check-

ing redundancy via information gain, and applying structured update.

Relevant Nodes Retrieval. Given a new trajectory $\tau = [q, o_1, a_1, \cdots]$, we retrieve a small set of related nodes to serve as the comparison context for subsequent updates. Concretely, we construct the semantic vector for each trajectory by concatenating the normalized CLIP representations of the text query q and the first observation o_1:

$$\mathbf{v} = [\text{CLIP}_{txt}(q); \text{CLIP}_{img}(o_1)] \quad (1)$$

where CLIP_{txt} and CLIP_{img} denote the text and image encoders of CLIP (Radford et al., 2021). Then, we use FAISS (Johnson et al., 2017) to perform nearest-neighbor search based on cosine similarity to obtain the top-K most similar trajectory nodes (and their linked attribute/strategy nodes).

Redundancy Checking. Similarity cannot distinguish whether a new trajectory provides sufficient information gain in current memory. We therefore introduce a redundancy check using a VLM judge.

Conditioned on the new trajectory and the retrieved neighbors (along with their first screenshots for visual context), the judge evaluates: (i) whether the new trajectory represents a genuinely different strategy not covered by existing nodes (ADD), (ii) whether it describes the same strategy as an existing node but can improve the existing takeaway by incorporating complementary insights (MERGE), or (iii) whether it covers the same strategy but is strictly superior in specificity and actionability, warranting replacement of the existing node (REPLACE). This decision explicitly measures the *marginal utility* of storing the trajectory and drives the subsequent structured update operation.

Memory Update. Based on the judge results, we update the graph by following operations:

- ADD: If the trajectory introduces new attributes or a previously unseen strategy, we create the corresponding strategy and/or attribute nodes (if missing), insert the new trajectory embedding node, and connect it to its attributes and strategy.
- MERGE: If the trajectory matches an existing strategy but provides complementary evidence (*e.g.,* a new UI variant or a different action path), we attach the new trajectory node to the matched attribute nodes and update the associated strategy node by consolidating newly observed attributes.
- REPLACE: If the trajectory is a strictly better instance of an existing strategy (*e.g.,* fewer steps, higher success, or fewer risky actions under the same conditions), we replace the outdated trajectory evidence and refresh the attribute connections to reflect the improved execution.

After node updates, we evolve the graph connectivity by instantiating or strengthening edges between strategy and attribute nodes according to newly observed co-occurrences. This continual refinement encourages stable regularities to emerge over time, making the memory increasingly coherent, less redundant, and easier to retrieve via associative multi-hop search.

3.3 Memory Utilization

Based on the constructed hybrid structured memory, we maintain an agent *working memory* as dynamic inference-time context that guides execution and can be refreshed as the GUI state evolves. Concretely, we first retrieve relevant experiences from the global memory to initialize the working memory, and then update it on-the-fly whenever the task transitions into a new interaction phase.

Structured Retrieval. We start by identifying a set of high-relevance *seed* nodes via semantic matching (Eq. 1). Specifically, we perform k-nearest neighbor search with the multimodal query embedding $\mathbf{v}$ against the stored node embeddings to obtain an initial candidate set $\tilde{N}$. To improve coverage beyond surface-level similarity, we expand retrieval through the graph: for each seed node $n \in \tilde{N}$, we collect its 1-hop neighbors, rerank them by cosine similarity to $\mathbf{v}$, and add the top-k neighbors into $\tilde{N}$. Since these neighbors often capture conceptually necessary knowledge that may not be visually similar to the current screen, we repeat this expand-and-rerank procedure for t iterations, resulting in a diverse yet relevant set of nodes for constructing the working memory.

Working Memory Initialization. Given the retrieved node set $\tilde{N}$, we form the working memory in two complementary views. First, we digest the associated strategy and attribute nodes into concise *Guidance Instructions* using a VLM, which filters irrelevant details and synthesizes actionable, strategy-level advice; these instructions are injected into the system prompt to steer high-level planning. Second, following CoMEM (Wu et al., 2025c,a), we concatenate the corresponding continuous trajectory embeddings with the VLM input to provide implicit access to fine-grained visual/action

evidence. Together, the discrete Guidance Instructions act as a semantic anchor that primes attention, while the continuous embeddings preserve details that are difficult to express textually, enabling more grounded and reliable action prediction.

On-the-fly Working Memory Refresh. Because GUI tasks can change substantially after certain actions, a fixed retrieved context can quickly become stale. We therefore enable on-the-fly local evolution during agent inference. After each action, a VLM compares the transition $(o_t \rightarrow o_{t+1})$ with the current Guidance Instructions to detect phase shifts (*e.g.,* from "search" to "checkout"). When a shift is detected, the VLM further determines which parts of the current working memory should be preserved (*e.g.,* long-term goals and constraints) and which should be discarded. We then rerun retrieval conditioned on the updated state, merge newly retrieved guidance with the preserved takeaways, and refresh both the Guidance Instructions and the continuous embeddings. This cyclic update keeps the context compact and relevant, ensuring that the agent remains synchronized with the evolving GUI environment throughout long-horizon execution.

4 Experiments

4.1 Experiment Settings

This section presents our experimental setup, including the agent's action interface, memory construction and retrieval settings, evaluation benchmarks, and baseline comparisons.

Action Interface. Our agent follows a ReAct-style paradigm, combining structured reasoning with tool-based execution. The model accepts screenshot multimodal inputs and interacts with a predefined toolset comprising GUI tools (*e.g.,* `CLICK`, `TYPE`, `SCROLL PAGE`, `WAIT`, `GO BACK`, `STOP`) and semantic tools (*e.g.,* `Page Content Analyzer`, `Change Page`). To support accurate grounding (*e.g.,* clicking or typing on specific UI elements), we adopt a two-stage approach:

- *Stage 1:* Screenshots are augmented using a SOM-based method, which assigns unique labels to UI elements. The agent generates references based on these labels.
- *Stage 2:* If the agent fails to produce a valid label, we fall back to UI-INS-7B (Chen et al., 2025), a robust vision-language grounding model, to resolve the location.

This hybrid mechanism ensures robust and reliable UI interaction, especially in visually complex or dynamic web environments.

Memory Settings. We collect a total of 2,883 successful trajectories from the following sources: GUIAct (Chen et al., 2024): A curated set of successful agent behaviors; Mind2Web Training Set (Deng et al., 2023; Zheng et al., 2024): Human-annotated demonstrations; Agent Rollouts: High-confidence trajectories generated by our own agent during training. In the impact of memory size ablation study 4.3.2, we also include trajectories from the automated data flywheel(Wu et al., 2025c) to expand the memory graph. To support structured and scalable retrieval, we organize the trajectories into a graph: Each node represents one or more similar trajectories, annotated with metadata such as *task descriptions*, *strategy*, and *attributes*. Edges are created between nodes that share overlapping tags. The final graph we used in the main table 2 contains 1,858 nodes and over 1 million edges. Some trajectories are merged into the same node for redundancy reduction. At inference time, the agent retrieves 10 relevant trajectories with the structured retrieval method introduced in 3.3: Top-5 trajectories are selected as seed nodes based on multimodal (visual + text) similarity to the current input. The remaining 5 are selected via graph expansion by taking 1-hop neighbors of the seed nodes.

Training Recipes. we perform parameter-efficient fine-tuning on the Q-Former and LoRA layers within the VLM memory encoder, and only 1.2% of the total model parameters are updated during training. Empirically, we observe that training converges rapidly, and a single epoch is sufficient to achieve strong performance. More details about training settings can be found in A.

Evaluation Benchmarks. We evaluate our approach on three challenging multimodal web-agent benchmarks: WebVoyager (He et al., 2024), Multimodal-Mind2Web (Deng et al., 2023; Zheng et al., 2024), and MMInA (Tian et al., 2025).

For evaluation on WebVoyager and Multimodal-Mind2Web, we adopt the LLM-as-Judge protocol (He et al., 2024), where we feed the task description and screenshot trajectory to a VLM to determine if the task was successfully completed. For MMInA, we compare the model's final answer with the ground truth using a language model. We report Task Accuracy as the primary metric.

To assess the effectiveness of our proposed memory-augmented framework, we compare the performance against the following baselines:

Closed-Source Base Models: GPT-4o (OpenAI, 2024), Gemini2.5-Pro-Vision (Google, 2023), and Claude-4 (Anthropic, 2025), to provide a comparison with strong proprietary base model capabilities.

Open-Source Base Models: We select a diverse set of open-source models to evaluate performance across different scales and adaptations. This includes large-scale models like Qwen2.5-VL-32B (Bai et al., 2025b), and task-specific models such as CogAgent (Hong et al., 2023) and WebSight (Bhathal and Gupta, 2025). Additionally, we evaluate on smaller LLMs including Qwen2.5-VL-7B (Bai et al., 2025b), Qwen3-VL-8B (Bai et al., 2025a), and UI-TARS-1.5-7B (Qin et al., 2025), which serve as the foundation for our experiments.

Open-Source + Memory-Augmented Models:

- *Textual Memory:* ReasoningBank (Ouyang et al., 2025) and Agent Workflow Memory (AWM) (Wang et al., 2024) are advanced text-based memory methods. They distill experience from past trajectories in the form of tokenized textual prompts, serving as a baseline to assess the impact of text-based memory.
- *Discrete Memory:* Our graph-based retrieval system retrieves relevant trajectories and digests these experiences using a VLM to provide contextual guidance for the current state.
- *Continuous Memory:* Following (Wu et al., 2025c), we encode multimodal trajectory data into continuous embeddings, evaluating the benefits of rich, multimodal memory representations for complex agent tasks.
- *Hybrid Memory:* A fusion of discrete and continuous memory that allows for dynamic memory updating during task execution.

All results in Table 2 are based on our own implementation. We maintain consistent environment setup, task sampling, and evaluation criteria across all models to ensure fair comparisons.

4.2 Result Analysis

Table 2 compares task success rates across three benchmarks. The results highlight the impact of different memory strategies across various base models. Notably, HyMEM consistently improves performance and even outperforms strong closed-source models like GPT-4o and Gemini2.5-Pro-Vision. Without memory, open-source baselines such as Qwen2.5-VL-7B and Qwen3-VL-8B perform not well, achieving 12.5% and 21.9% respectively. Larger or specialized models like Qwen2.5-VL-32B, CogAgent, and WebSight also struggle, indicating that model scale or task-specific fine-tuning alone is insufficient for long-horizon web tasks. Adding advanced textual memory summaries (*e.g.,* Reasoning Bank, AWM) yields moderate gains (*e.g.,* +5.0%, +0.6% on Qwen2.5-VL-7B), but these methods are limited by their unimodal nature and lack of memory management mechanisms like updating or pruning. Consequently, they often fail to capture the rich, evolving context required for robust reasoning in complex environments. In contrast, both Discrete and Continuous memory individually offer consistent gains. For instance, Qwen2.5-VL-7B improves from 12.5% to 17.0% (Discrete) and 21.7% (Continuous); Qwen3-VL-8B shows similar trends. However, neither method alone is universally effective. For example, UI-TARS-1.5-7B performs better with continuous memory than with discrete memory, highlighting the complementary strengths of them.

The HyMEM approach, which combines discrete high level concepts with continuous multimodal representations, achieves the best results across all open-source models. Qwen2.5-VL-7B + Hybrid reaches 35.0% (+22.5%), outperforming Claude-4 in domains like Travel and Wikipedia, and surpasses GPT-4o and Gemini2.5-Pro-Vision in average. Qwen3-VL-8B and UI-TARS-1.5-7B also achieve their highest scores with HyMEM. These findings demonstrate that Hybrid Memory enables agents to reason wisely in the detailed multimodal context, which is a key capability for solving complex and multi-step web tasks.

4.3 Further Analysis

4.3.1 Memory Self-evolving Analysis

We evaluate the self-evolving mechanisms of HyMEM by comparing a static-memory baseline with the self-evolving variant on Amazon and Google Maps domains. As shown in Figure 2, we isolate the contributions of *global evolution*, the long-term consolidation, and *local evolution*, the working memory adaptation.

Global evolution drives continuous learning by integrating successful experiences into reusable

Backbone	Model/Method	WebVoyager				Mind2Web				MMInA	Overall
		Amz	Cour	Recp	Map	Info	Svc	Ent	Trav	Wiki	
Closed-Source	GPT-4o	24.4%	7.1%	13.3%	36.6%	7.8%	14.1%	3.4%	19.4%	51.0%	19.7%
	Gemini-Pro-Vision	41.7%	42.9%	20.0%	53.7%	16.7%	22.4%	0.0%	19.4%	50.0%	29.6%
	Claude-4	63.4%	28.6%	33.3%	70.0%	25.6%	40.0%	6.9%	9.7%	52.0%	36.6%
Open-Source	Qwen2.5-VL-32B	46.3%	26.2%	6.7%	29.3%	14.1%	20.0%	6.9%	9.7%	43.0%	22.5%
	CogAgent	12.2%	9.5%	26.7%	9.8%	24.4%	8.2%	13.8%	16.1%	21.0%	15.7%
	Websight	24.4%	4.8%	13.3%	29.3%	10.3%	3.5%	3.4%	0.0%	12.0%	11.2%
UI-TARS-1.5-7B	Baseline	31.7%	16.7%	20.0%	31.7%	6.4%	4.7%	6.9%	0.0%	36.0%	17.1%
	+ Reasoning Bank	0.0%	11.9%	6.7%	12.2%	7.7%	8.2%	3.7%	**6.7%**	44.0%	11.2%
	+ AWM	4.9%	7.1%	4.4%	14.6%	10.3%	3.5%	3.7%	3.3%	26.0%	8.6%
	+ Discrete	19.5%	16.7%	8.9%	17.1%	11.5%	8.2%	7.4%	3.4%	46.0%	15.4%
	+ Continuous	43.9%	**33.3%**	24.4%	43.9%	16.7%	**28.2%**	10.3%	3.2%	54.0%	28.7%
	+ **HyMEM**	**58.5%**	28.6%	**26.7%**	**53.7%**	**16.7%**	25.9%	**10.3%**	6.5%	**54.0%**	**31.2%**
Qwen2.5-VL-7B	Baseline	14.6%	2.4%	15.9%	16.7%	9.0%	11.8%	0.0%	4.4%	38.0%	12.5%
	+ Reasoning Bank	29.3%	9.5%	6.7%	29.3%	9.0%	20.0%	3.4%	6.5%	44.0%	17.5%
	+ AWM	17.1%	4.8%	11.1%	29.3%	7.7%	10.6%	0.0%	6.5%	31.0%	13.1%
	+ Discrete	22.0%	21.4%	8.9%	31.7%	9.0%	12.9%	10.3%	3.2%	34.0%	17.0%
	+ Continuous	24.4%	17.1%	8.9%	34.1%	16.7%	23.5%	**10.3%**	12.9%	47.0%	21.7%
	+ **HyMEM**	**63.4%**	**54.8%**	**20.0%**	**53.7%**	**17.9%**	**23.5%**	3.4%	**22.6%**	**56.0%**	**35.0%**
Qwen3-VL-8B	Baseline	36.6%	16.7%	13.3%	**43.9%**	9.0%	20.0%	10.3%	9.7%	38.0%	21.9%
	+ Reasoning Bank	36.6%	11.9%	17.8%	31.7%	10.3%	**24.7%**	**17.2%**	6.5%	**48.0%**	22.7%
	+ AWM	39.0%	19.0%	8.9%	31.7%	12.5%	16.5%	6.9%	16.1%	48.0%	22.1%
	+ Discrete	31.7%	16.7%	15.6%	31.7%	11.5%	16.5%	13.8%	16.1%	38.0%	21.3%
	+ Continuous	43.9%	19.0%	17.8%	34.1%	12.5%	17.9%	3.4%	19.4%	42.0%	23.3%
	+ **HyMEM**	**46.3%**	**19.0%**	**26.7%**	39.0%	**17.9%**	20.0%	6.9%	**25.8%**	47.0%	**27.6%**

Table 2: Task success rates (%) across WebVoyager, Mind2Web, and MMInA. Bolded entries indicate best-performing open-source models in each domain. Overall is averaged across available domains.

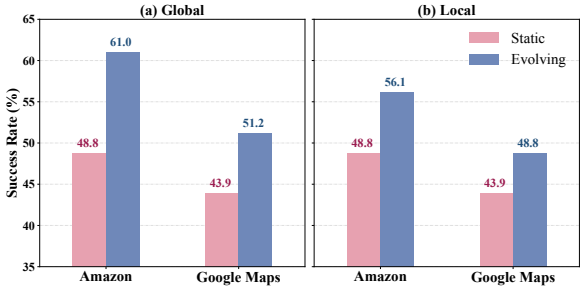

Figure 2: Success rate comparison between static memory and self-evolving HyMEM. (a): Global evolution. (b): Local evolution via working memory refresh.

strategies. As shown in Figure 2 (a), this yields substantial relative gains: ∼25% on Amazon (48.8% → 61.0%) and ∼16.6% on Google Maps (43.9% → 51.2%). Our VLM-based deduplication balances growth with compression, where half of successful trajectories (*e.g.,* 52% on Google Maps) trigger UPDATE operations to merge existing strategies rather than creating redundant nodes. This loop refines the memory without uncontrolled expansion, enabling continual learning in application.

Local evolution adapts the agent's working context to shifting UI states. Figure 2 (b) highlights consistent performance improvements, such as a ∼15% gain on Amazon. Empirically, the agent triggered Dynamic Relevance Assessment frequently (*e.g.,* 39 times on Google Maps) to detect workflow "phase shifts." For instance, in a task finding EV charging near museums, HyMEM successfully identified the transition from generic navigation to specific filtering, explicitly reasoning that the task phase had changed. This active context management maintains high relevance throughout long-horizon tasks, whereas the static baseline fails as context drifts from the initial retrieval.

4.3.2 Impact of Memory Size

To investigate the scaling behavior of our hybrid graph memory system, we analyze three aspects: (i) how performance improves with larger memory graphs; (ii) how adding more memory items influences success; and (iii) how efficiently our graph representation compresses redundant trajectories.

Size of Memory Graph As shown in Table 3, increasing the size of the memory graph leads to consistent gains in task success across domains. For instance, in the Amazon domain, scaling the memory from 500 to 1,000 trajectories boosts suc-

cess from 19.5% to 31.7%, while using the largest graph (8,000 examples) achieves 63.4%. This trend demonstrates that larger memory graphs enable more effective retrieval of relevant trajectories, enhancing the agent's performance.

Number of Retrieved Memory Items Figure 3(a) further illustrates this scaling trend by showing how success rate improves as we increase the number of retrieved memory items. We observe steady returns, suggesting that richer memory inputs contribute to better performance.

Graph Compression Finally, as shown in Figure 3(b), our graph memory exhibits a desirable compression property by merging semantically similar trajectories into unified nodes. As the number of raw trajectories increases, the number of graph nodes grows sublinearly, enabling efficient scaling without linear memory overhead.

Memory Size	500	1000	2000	5000	8000
Amazon	19.5%	31.7%	63.4%	65.9%	63.4%
Google Map	31.7%	46.3%	51.2%	51.2%	56.1%
AllRecipes	11.1%	13.3%	13.3%	15.6%	20.0%

Table 3: Impact of Memory Bank Size on Task Success across different domains. Increasing the number of stored trajectories significantly improves the retrieval quality and overall agent success rate.

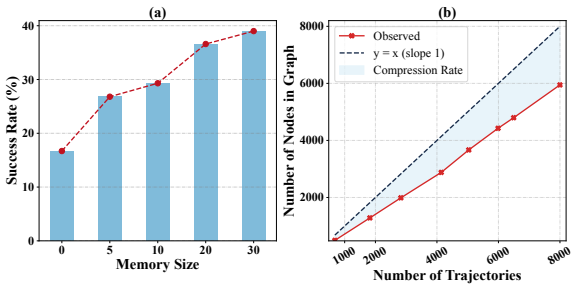

Figure 3: (a) Scaling Behavior of Memory Size in Coursera domain and (b) Graph Compression Analysis.

4.3.3 Similarity and Diversity Trade-off

As stated in section 3.3, Graph based retrieval and expansion can increase the diversity of retrieved trajectories. To investigate the trade-off between similarity and diversity in our graph-based memory retrieval, we compare three retrieval strategies: (i) $|\widetilde{\mathcal{N}}_{\text{initial}}| = 10$ where all 10 trajectories are retrieved solely using FAISS based on multimodal similarity to the current task, prioritizing similarity over diversity; (ii) $|\widetilde{\mathcal{N}}_{\text{initial}}| = 5$, where 5 seed trajectories are retrieved via FAISS and the remaining 5 are selected from their 1-hop graph neighbors, keeping the balance between similarity and diversity; and (iii) $|\widetilde{\mathcal{N}}_{\text{initial}}| = 1$, where only 1 similar trajectory is used as the seed and the other 9 are drawn from the 1-hop neighborhood, prioritizing the diversity.

As shown in Table 4, the strategy with $|\widetilde{\mathcal{N}}_{\text{initial}}| = 5$ achieves the highest overall task success rate across all three domains (Amazon: 63.4%, Coursera: 54.8%, GMap: 53.7%), outperforming the other two settings. This suggests that combining relevant yet diverse trajectories provides the most effective contextual grounding for the agent. It also highlights the importance of balancing semantic similarity and diversity in memory retrieval to support generalizable and effective reasoning.

Strategy	Amazon	Coursera	GMap
$\|\widetilde{\mathcal{N}}_{\text{initial}}\| = 5$	**56.1%**	**38.1%**	**53.7%**
$\|\widetilde{\mathcal{N}}_{\text{initial}}\| = 10$	53.7%	31.0%	46.3%
$\|\widetilde{\mathcal{N}}_{\text{initial}}\| = 1$	46.3%	31.0%	48.8%

Table 4: Task success rate under different retrieval strategies, showing the similarity and diversity trade-off.

5 Conclusion

Existing GUI agent memory approaches remain limited by flat storage and retrieval that do not support structured organization or continual evolution. To solve it, inspired by human memory, we proposed Hybrid Self-evolving Structured Memory (HYMEM), which unifies discrete high-level symbolic strategy nodes with continuous trajectory embeddings in a graph-based memory. This structure enables (i) multi-hop, seed-and-expand retrieval for more informative and diverse context, (ii) self-evolution through node add/merge/replace updates as new trajectories arrive, and (iii) on-the-fly working-memory refreshing during inference to better support multi-turn execution and phase changes. Extensive experiments across multiple benchmarks demonstrate that HYMEM consistently strengthens open-source GUI agents and substantially narrows—and in some settings surpasses—the gap to strong closed-source systems.

Looking ahead, the hybrid, self-evolving memory paradigm offers a promising foundation for scalable agent improvement: extending the graph to richer strategy abstractions, improving update policies under distribution shift, and leveraging memory evolution for continual learning.

6 Limitations

While our proposed method demonstrates strong performance gains across multiple benchmarks, there are several limitations for further exploration. First, although the memory update mechanism leverages the base VLM to decide whether an update is necessary, this approach still relies on heuristics rather than being fully optimized. A more principled solution—such as training the model with reinforcement learning could enable the agent to learn when and how to use or update memory more effectively, potentially leading to further improvements. Second, due to limitations in computational resources, we were unable to evaluate our method on larger-scale models (*e.g.,* 32B or 70B). Investigating the scalability of our approach with more powerful backbones remains an important direction for future work.

7 Ethical considerations

This work focuses on enhancing the memory and reasoning capabilities of open-source GUI agents. All experiments are conducted using publicly accessible websites and publicly available datasets that are commonly used in prior research. Our data collection and evaluation procedures strictly adhere to website usage policies and do not involve any login-protected content, private user data, or actions that would violate terms of service. As such, our work does not present any privacy risks or ethical concerns in its current form.

However, in real-world applications, deploying autonomous agents capable of interacting with web interfaces at scale necessitates careful safeguards. Without proper constraints, such agents could unintentionally perform harmful or unauthorized actions. We admit it is crucial to incorporate robust safety mechanisms, access controls, and ethical oversight when applying this technology beyond controlled research settings.

References

Anthropic. 2025. Claude 4. https://www.anthropic.com/index/claude-4. Accessed: 2025-09-21.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, and 45 others. 2025a. Qwen3-vl technical report. *Preprint*, arXiv:2511.21631.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, and 8 others. 2025b. Qwen2.5-vl technical report. *Preprint*, arXiv:2502.13923.

Tanvir Bhathal and Asanshay Gupta. 2025. Websight: A vision-first architecture for robust web agents. *arXiv preprint arXiv:2508.16987*.

Liangyu Chen, Hanzhang Zhou, Chenglin Cai, Jianan Zhang, Panrong Tong, Quyu Kong, Xu Zhang, Chen Liu, Yuqi Liu, Wenxuan Wang, Yue Wang, Qin Jin, and Steven Hoi. 2025. Ui-ins: Enhancing gui grounding with multi-perspective instruction-as-reasoning. *Preprint*, arXiv:2510.20286.

Wentong Chen, Junbo Cui, Jinyi Hu, Yujia Qin, Junjie Fang, Yue Zhao, Chongyi Wang, Jun Liu, Guirong Chen, Yupeng Huo, Yuan Yao, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2024. Guicourse: From general vision language models to versatile gui agents.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems*, volume 36, pages 28091–28114. Curran Associates, Inc.

Gemini Team Google. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. 2024. Webvoyager: Building an end-to-end web agent with large multimodal models. *arXiv preprint arXiv:2401.13919*.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, and Jie Tang. 2023. Cogagent: A visual language model for gui agents. *Preprint*, arXiv:2312.08914.

Mengkang Hu, Tianxing Chen, Qiguang Chen, Yao Mu, Wenqi Shao, and Ping Luo. 2025a. HiAgent: Hierarchical working memory management for solving long-horizon agent tasks with large language model. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32779–32798, Vienna, Austria. Association for Computational Linguistics.

Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, Boyang Liu, Fangyi Zhu, Jiahang Lin, Honglin Guo, Shihan Dou, Zhiheng Xi, Senjie Jin, Jiejun Tan, Yanbin Yin, Jiongnan Liu, Zeyu Zhang, Zhongxiang Sun, Yutao Zhu, Hao Sun, Boci Peng, and 28 others. 2025b. Memory in the age of ai agents. *Preprint*, arXiv:2512.13564.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with gpus. *Preprint*, arXiv:1702.08734.

Tiange Luo, Lajanugen Logeswaran, Justin Johnson, and Honglak Lee. 2025. Visual test-time scaling for gui agent grounding. *Preprint*, arXiv:2505.00684.

James L. McClelland, Bruce L. McNaughton, and Randall C. O'Reilly. 1995. Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological Review*, 102(3):419–457.

OpenAI. 2024. Gpt-4o technical report. https://openai.com/index/hello-gpt-4o/. Accessed September 2025.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. 2025. Reasoningbank: Scaling agent self–evolving with reasoning memory. *Preprint*, arXiv:2509.25140.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. 2024. Memgpt: Towards llms as operating systems. *Preprint*, arXiv:2310.08560.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, Wanjun Zhong, Kuanye Li, Jiale Yang, Yu Miao, Woyu Lin, Longxiang Liu, Xu Jiang, Qianli Ma, Jingyu Li, and 16 others. 2025. Ui-tars: Pioneering automated gui interaction with native agents. *Preprint*, arXiv:2501.12326.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. *Preprint*, arXiv:2103.00020.

Zhili Shen, Chenxin Diao, Pavlos Vougiouklis, Pascual Merita, Shriram Piramanayagam, Enting Chen, Damien Graux, Andre Melo, Ruofei Lai, Zeren Jiang, Zhongyang Li, Ye Qi, Yang Ren, Dandan Tu, and Jeff Z. Pan. 2025. GeAR: Graph-enhanced agent for retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 12049–12072, Vienna, Austria. Association for Computational Linguistics.

Eleanor Spens and Neil Burgess. 2024a. Consolidation of sequential experience into a deep generative network explains human memory, prediction and planning. *bioRxiv*. Preprint.

Eleanor Spens and Neil Burgess. 2024b. A generative model of memory construction and consolidation. *Nature Human Behaviour*, 8(3):526–543.

Shulin Tian, Ziniu Zhang, Liangyu Chen, and Ziwei Liu. 2025. Mmina: Benchmarking multihop multimodal internet agents. *Preprint*, arXiv:2404.09992.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. 2024. Agent workflow memory: Continually inducing and applying workflows for web agents. *Preprint*, arXiv:2409.07429.

Samantha E. Williams, Jaclyn H. Ford, and Elizabeth A. Kensinger. 2022. The power of negative and positive episodic memories. *Cognitive, Affective, & Behavioral Neuroscience*, 22(5):869–903.

Wenyi Wu, Zixuan Song, Kun Zhou, Yifei Shao, Zhiting Hu, and Biwei Huang. 2025a. Towards general continuous memory for vision-language models. *Preprint*, arXiv:2505.17670.

Wenyi Wu, Kun Zhou, Ruoxin Yuan, Vivian Yu, Stephen Wang, Zhiting Hu, and Biwei Huang. 2025b. Auto-scaling continuous memory for GUI agent. *Preprint*, arXiv:2510.09038.

Wenyi Wu, Kun Zhou, Ruoxin Yuan, Vivian Yu, Stephen Wang, Zhiting Hu, and Biwei Huang. 2025c. Auto-scaling continuous memory for gui agent. *Preprint*, arXiv:2510.09038. ArXiv preprint arXiv:2510.09038.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2025. A-mem: Agentic memory for LLM agents. *Preprint*, arXiv:2502.12110.

Leyang Yang, Ziwei Wang, Xiaoxuan Tang, Sheng Zhou, Dajun Chen, Wei Jiang, and Yong Li. 2025. Probench: Benchmarking gui agents with accurate process information. *Preprint*, arXiv:2511.09157.

Chi Zhang, Zhao Yang, Jiaxuan Liu, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. 2023. Appagent: Multimodal agents as smartphone users. *Preprint*, arXiv:2312.13771.

Guibin Zhang, Muxin Fu, Guancheng Wan, Miao Yu, Kun Wang, and Shuicheng Yan. 2025a. G-memory: Tracing hierarchical memory for multi-agent systems. *Preprint*, arXiv:2506.07398.

Guibin Zhang, Muxin Fu, and Shuicheng Yan. 2025b. Memgen: Weaving generative latent memory for self-evolving agents. *Preprint*, arXiv:2509.24704.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024. Expel: Llm agents are experiential learners. *Preprint*, arXiv:2308.10144.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. 2024. Gpt-4v(ision) is a generalist web agent, if grounded.

A Implementation Details

Evaluation Benchmarks We evaluate our method on three public benchmarks. We introduce our detailed settings here:

- **WebVoyager** (He et al., 2024): A dataset focusing on multimodal reasoning in dynamic web environments. We show results on four representative websites in the main table 2: Amazon (Amz), Coursera (Cour), AllRecipes (Recp), and Google Maps (Map).

- **Multimodal-Mind2Web** (Deng et al., 2023; Zheng et al., 2024): A large-scale benchmark with 2,000+ open-ended tasks, which include domains and websites not seen during training, thereby evaluating the out-of-distribution (OOD) generalizability of the agent. We skip tasks where the associated website is no longer accessible at evaluation time.

- **MMInA** (Tian et al., 2025): A real-world multistep benchmark requiring multimodal reasoning and planning across domains such as shopping and travel. We evaluate on the Wikipedia domain using the first 100 tasks.

Continuous Encoder Training For training data, we use 1,009 trajectories from the Mind2Web training set (Deng et al., 2023), along with 1,000 successful trajectories collected through rollouts by our agent. These tasks are generated from publicly accessible websites such as Amazon, Coursera, Google Maps, and others.

Using this data, we perform parameter-efficient fine-tuning on the Q-Former and LoRA layers within the VLM encoder. To improve efficiency, we apply LoRA with a rank of 16 and share parameters across all layers of the Q-Former. As a result, only 1.2% of the total model parameters are updated during training. These parameter- and data-efficient design choices allow the entire training process to be completed within 20 hours on a single NVIDIA H100 GPU.

Empirically, we observe that training converges rapidly, and a single epoch is sufficient to achieve strong performance. We apply the same training strategy across multiple vision-language models, including Qwen2.5-VL-7B (Bai et al., 2025b), Qwen3-VL-8B (Bai et al., 2025a), and UI-TARS-1.5-7B (Qin et al., 2025).

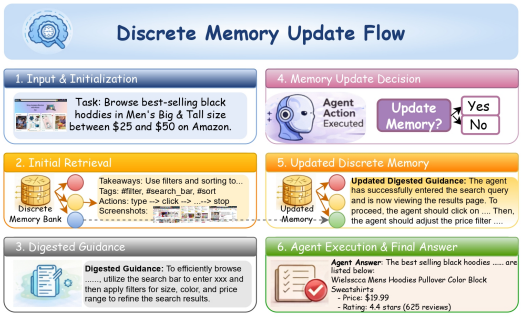

Figure 4: Visualization of Discrete Memory Update Flow.

B Additional Qualitative Results

Visualization of Hybrid Memory Figure 4 illustrates the complete workflow of discrete memory within our hybrid memory system, highlighting how it is generated, updated, and utilized during agent execution. The process begins with user task input and environment initialization (Step 1), followed by the retrieval of relevant past memory from the discrete memory bank (Step 2). This retrieval includes structured takeaways, action traces, and visual references, which are then distilled into actionable guidance (Step 3).

As the agent executes actions in the environment, a memory update decision is triggered (Step 4). If new knowledge or behaviors are observed, the memory is updated accordingly (Step 5), incorporating the latest digested guidance. Finally, the agent uses this refined memory to guide subsequent actions and generate the final answer (Step 6). This workflow enables the agent to continually improve its decision-making by leveraging structured, interpretable memory updates throughout the task.

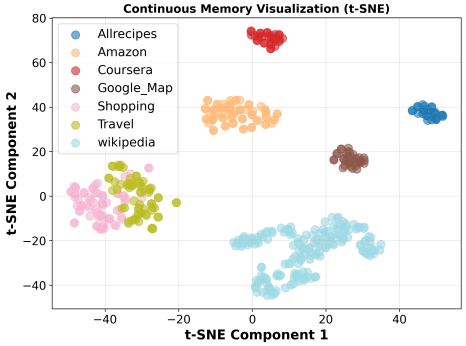

Figure 5: t-SNE visualization of continuous memory embeddings across domains. Each point represents a stored trajectory embedding, color-coded by domain.

To better understand how hybrid memory or-

ganizes knowledge across domains, we visualize the continuous memory embeddings using t-SNE, as shown in Figure 5. Each point represents a trajectory stored in memory, and colors indicate different domains. The clear clustering structure demonstrates that the memory embeddings capture meaningful semantic distinctions between domains. For instance, trajectories from Amazon, Coursera, and Google Map form well-separated clusters, while related domains such as Travel and Shopping show some proximity, reflecting shared behavioral patterns. This interpretable structure supports effective cross-domain retrieval and highlights the advantage of learning unified memory representations.

Case Study In this example 6, the agent is tasked with finding a USB-C hub on Amazon that meets some specific criteria. The agent powered by Qwen2.5-VL-7B + Hybrid Memory successfully completes the task by leveraging both current context and retrieved memory.

The agent first interprets the goal accurately, identifying key constraints such as compatibility, port requirements, and price. During execution, it recognizes the need to update memory and retrieves relevant trajectories involving similar filtering and product selection behavior. These include both high-level concepts (*e.g.,* filtering by features) and low-level UI actions (*e.g.,* selecting "Best Sellers"). Guided by this contextualized memory, the agent efficiently navigates the interface, applies the correct filters, and selects a product that meets all constraints. This case illustrates how hybrid memory enhances reasoning by grounding decisions in both semantic and procedural knowledge.

In contrast, the same task is attempted using the Qwen2.5-VL-7B base model without access to hybrid memory 7. Initially, the agent is able to locate the search bar and input the query correctly. It also selects the sorting option, but fails to apply the necessary filters: Best Sellers. As the task progresses, the model repeatedly opens similar product pages without making meaningful progress toward the goal. Eventually, it hits the maximum step limit without identifying a suitable product. Without access to relevant examples or structured memory, the model struggles to decompose the high-level instruction into effective UI actions and loses context part way through the task.

C Prompts Used in HyMEM

Guidance Digestion Prompt This prompt is used in HyMEM's inference-time digestion stage to convert retrieved experience summaries into a compact *Guidance Block*. As shown in Figure 8, given the current task intent, current screenshot, and retrieved summaries, it synthesizes task-specific navigation cues (*e.g.,* key actions/filters) while explicitly avoiding termination-related instructions.

Local Self-Evolution Prompt This prompt is used in HyMEM's inference-time *local evolution* stage to decide whether the current working memory should be refreshed after each environment transition. As shown in Figure 9, it conservatively distinguishes *operational errors* (no refresh) from genuine *phase shifts* (refresh), and identifies which existing takeaways should be preserved when re-retrieving for the new phase.

Global Evolution Prompt This prompt is used in HyMEM's *global evolution* stage to deduplicate newly observed successful trajectories against existing strategy nodes and decide whether to ADD, UPDATE, or REPLACE a node. As illustrated in Figure 10, the judge leverages both textual metadata (task/takeaway/tags/domain) and visual context (screenshots) and outputs a strict JSON decision for downstream graph updates.

D LLM Usage Statement

Large Language Models (LLMs) were utilized in this work solely for the purpose of grammatical refinement and language polishing to enhance readability. No substantive edits were made to the scientific content, conceptual development, or experimental results by these tools. The authors retain full responsibility for the final content and accuracy of the manuscript.

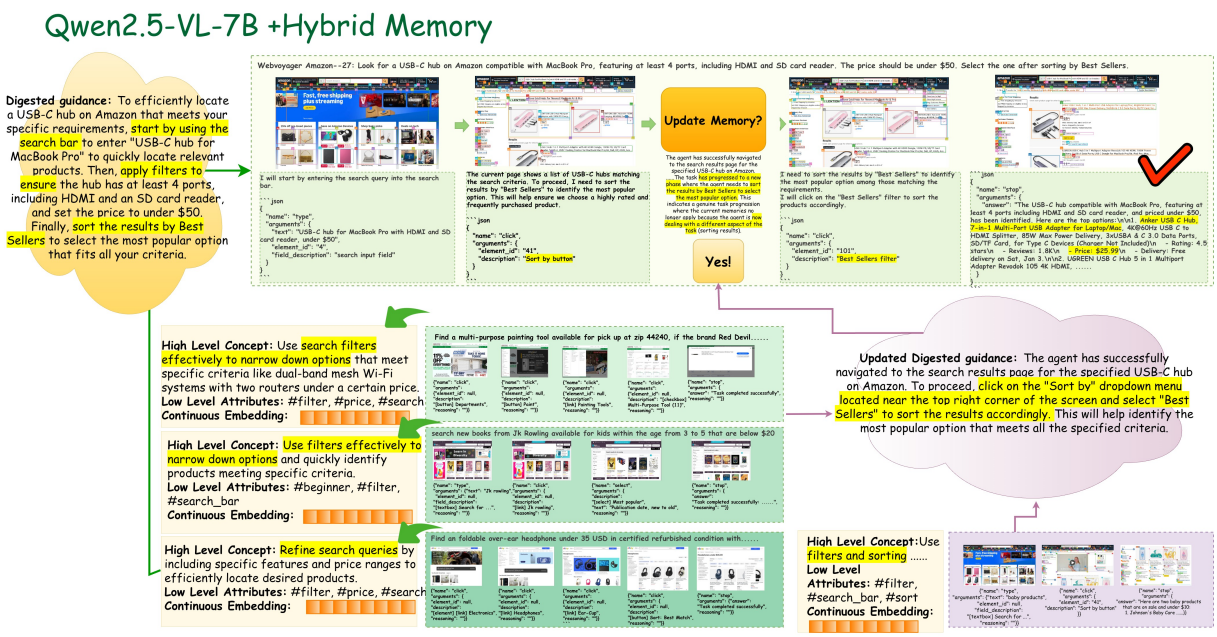

Figure 6: Successful case by Qwen2.5-VL-7B + Hybrid Memory in the Amazon task. The model accurately understands the goal, retrieves relevant memory, and selects the correct product by filtering.

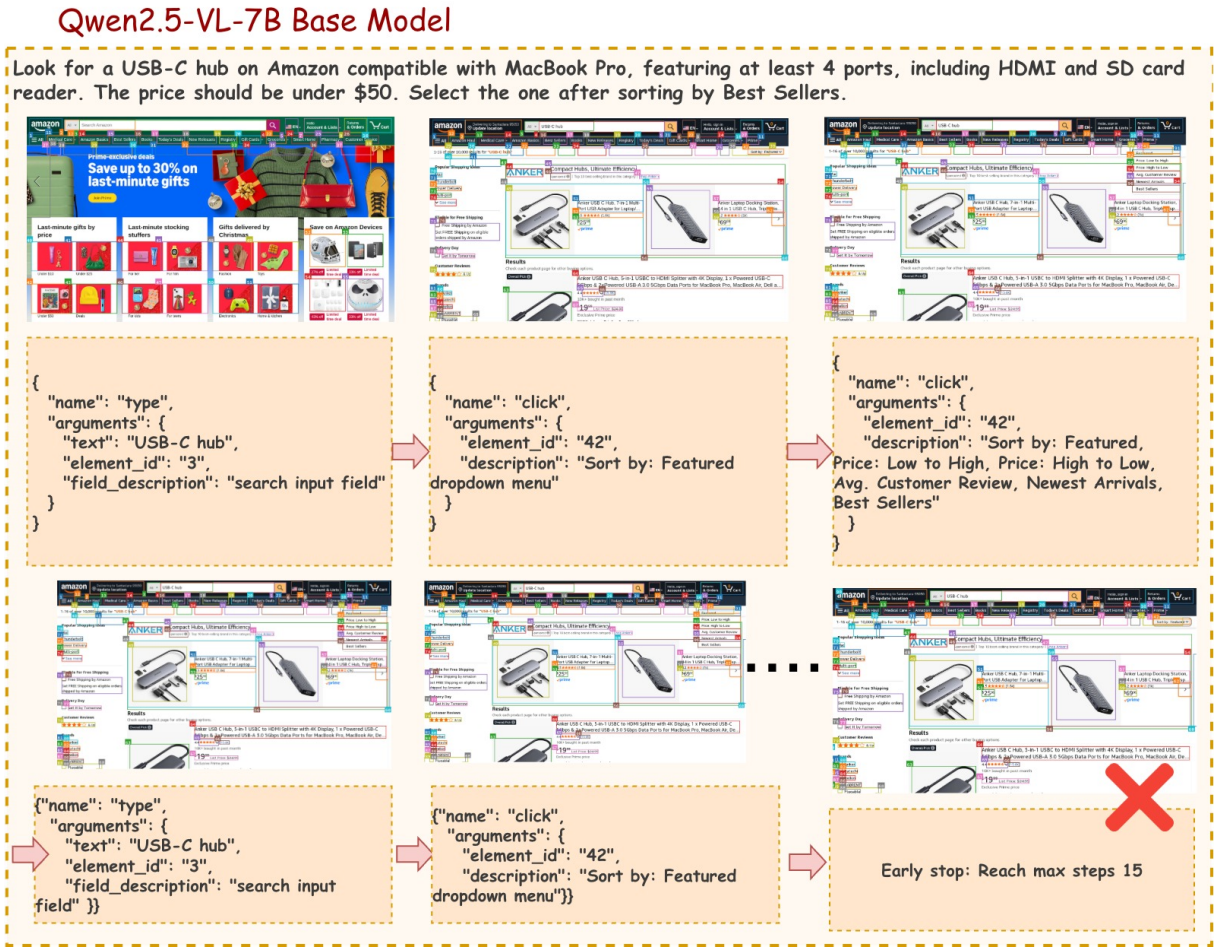

Figure 7: Failure case by Qwen2.5-VL-7B base model in the Amazon task. The model does not know how to filter products by the requirement, and loses track of the task state.

Guidance Digestion Prompt

System Instruction

```
You are an expert at analyzing past GUI agent experiences to help with a new task.
Given the current task, current screenshot, and retrieved experience summaries,
synthesize them into focused, actionable guidance.

Output format: ONE concise paragraph (2-3 sentences) that answers:
1. Which strategies from past experiences are MOST relevant to this specific task?
2. What key actions or filters should be prioritized?

IMPORTANT RULES:
- Focus ONLY on navigation/search strategies, NOT on when to stop.
- Do NOT mention stopping, completing, or finishing the task.
- Do NOT give instructions about providing answers or explanations.
- Be specific to the current task. Do NOT just repeat the summaries.
- Do NOT use bullet points. Write as a coherent paragraph.
```

Input Prompt

```
Current task:
{TASK_INTENT}

Current screenshot:
{SCREENSHOT}

Retrieved experience summaries:
{RETRIEVED_SUMMARIES}
```

Figure 8: **Guidance Digestion Prompt** used to synthesize retrieved experience summaries into a compact *Guidance Block* for inference-time navigation.

Local Self-Evolution Prompt

System Instruction

```
You are evaluating whether the retrieved experience memories are still relevant for the task.

KEY PRINCIPLES:
1. Memories provide NAVIGATION STRATEGIES (how to search, filter, click, scroll, etc.), NOT
    answers.
2. Be VERY CONSERVATIVE - default to keeping current memories.
3. CRITICAL: Distinguish between OPERATIONAL ERRORS vs MEMORY RELEVANCE:

   OPERATIONAL ERRORS (DO NOT update memory):
   - Agent landed on wrong page (login page, unrelated website, error page)
   - Agent is stuck or action failed
   - Agent made a navigation mistake
   -> These are execution errors, NOT memory problems. The current memories are still valid for
    when agent gets back on track.

   MEMORY RELEVANCE ISSUE (May update memory):
   - Task has genuinely progressed to a completely new phase
   - Agent successfully completed initial steps and now needs different strategies
   - Current screen shows the task is in a fundamentally different domain than memories cover
   -> Only update if memories are irrelevant to the ACTUAL TASK PROGRESS, not temporary
    navigation mistakes.

4. PRESERVE USEFUL MEMORIES: When updating, some existing takeaways may still be valuable:
   - General domain knowledge that applies across task phases
   - Strategies that might be needed again later (e.g., navigation, filtering)
   - Takeaways that provide complementary information to new phase requirements

Your job: Decide if memories should be updated, but ONLY for genuine relevance issues, NOT
    operational errors.
When updating, also identify which existing takeaways should be PRESERVED and combined with new
    retrieval.

OUTPUT FORMAT (strict):
Decision: KEEP or UPDATE
Preserve: [comma-separated IDs of takeaways to keep, or NONE]
Reason: one sentence (no more than 20 words)
```

Input Prompt

```
Current task:
{TASK_INTENT}

Previous screenshot:
{SCREENSHOT_t}

Current screenshot:
{SCREENSHOT_t+1}

Current guidance block (optional):
{GUIDANCE_BLOCK}

Current retrieved takeaways (with IDs):
{TAKEAWAYS_WITH_IDS}
```

Figure 9: **Local self-evolution prompt** for conservative working-memory refresh: it updates only on genuine phase shifts and preserves reusable takeaways.

Global Evolution Prompt

System Instruction

```
You deduplicate GUI agent strategy takeaways using both text and visual context.
Output valid JSON only, no markdown code blocks.
```

User Prompt

```
You are managing a knowledge base of GUI agent strategies for deduplication.

NEW TRAJECTORY (see Screenshot 1):
- task_description: {new_task}
- takeaway: {new_takeaway}
- tags: {new_tags}
- domain: {new_domain}

EXISTING SIMILAR TRAJECTORIES:
1. ID: {traj_id} (similarity: {sim_score:.3f}, see {screenshot_ref})
   task_description: {existing_task}
   takeaway: {existing_takeaway}
   tags: {existing_tags}
   domain: {existing_domain}
[... more similar trajectories ...]

DECIDE one of:
1. UPDATE: NEW describes the SAME or very similar strategy as an existing trajectory.
   -> Keep the existing node but IMPROVE its takeaway by incorporating insights from NEW.
   -> Provide the improved takeaway (must start with "takeaway:").
   -> Provide the complete updated tags list (can add, remove, or keep tags).

2. REPLACE: NEW describes the SAME strategy as an existing trajectory, but NEW is STRICTLY
    BETTER
   (more specific, more actionable, more complete trajectory).
   -> Remove the existing node, add NEW as replacement.

3. ADD: NEW describes a GENUINELY DIFFERENT strategy not covered by existing trajectories.
   -> Add NEW as a new node.

Use the screenshots to understand the visual context and UI state of each trajectory.

OUTPUT (JSON only, no markdown):
{
  "action": "update" | "replace" | "add",
  "reasoning": "one sentence explanation",
  "target_id": "ID to update (required for update)",
  "updated_takeaway": "takeaway: improved takeaway text (required for update)",
  "updated_tags": ["#tag1", "#tag2", ...],
  "old_id": "ID to replace (required for replace)"
}
```

Figure 10: **Global evolution prompt** for strategy deduplication and graph maintenance. The judge selects ADD/UPDATE/REPLACE using multimodal context and returns a strict JSON decision for deterministic memory updates.