MEMENTOGUI: LEARNING AGENTIC MULTIMODAL MEMORY CONTROL FOR LONG-HORIZON GUI AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent GUI agents have made substantial progress in visual grounding and action prediction, yet they remain brittle in long-horizon tasks that require maintaining task state across many interface transitions. Existing agents typically rely on raw history replay or text-only memory, which either overwhelms the model with redundant screenshots or discards localized visual evidence needed for future decisions. To address these limitations, we introduce **MEMENTOGUI**, a plug-in agentic memory framework that equips MLLM-based GUI agents with **MEMENTOCORE**, a learned controller for online memory selection, compression, and retrieval. Rather than treating interaction history as a fixed context, MEMENTOGUI formulates long-horizon GUI control as an online memory-control problem: working memory selectively preserves task-relevant interface events with textual summaries and ROI-level visual evidence, while episodic memory retrieves reusable past trajectories through learned relevance selection. MEMENTOCORE modularizes memory control into specialized operators for step processing, memory compression, episodic writing, and episodic selection, enabling plug-in memory augmentation without finetuning the GUI agent backbone. We further develop a scalable data curation pipeline that converts computer-use trajectories into memory-controller training data, introduce **MEMENTOGUI-BENCH** for evaluating long-horizon decision-making in GUI agents, and design MLLM-based metrics for semantic action matching, task progress, and memory consistency. Experiments on GUI-Odyssey, MM-Mind2Web, and MEMENTOGUI-BENCH show that MEMENTOGUI consistently improves GUI agents over no-history, history-replay, and text-only memory baselines, with larger MEMENTOCORE backbones further strengthening memory-augmented GUI control.

1 INTRODUCTION

Recent advances in multimodal large language models (MLLMs) (Bai et al., 2025; Hua et al., 2025b; Singh et al., 2026; Wang et al., 2025; Liu et al., 2023; Sun et al., 2025b) have enabled agentic systems that perceive, reason, and act in complex visual environments (Avogaro et al., 2026; Hua et al., 2024b; Hu et al., 2023; Thrush et al., 2022; Hua et al., 2025c;a). Graphical user interface (GUI) control is a representative setting for such agents, requiring visually grounded actions over dynamic software interfaces. While recent GUI agents have improved single-step grounding and action prediction (Deng et al., 2023; Gou et al., 2024; Lei et al., 2025; Zheng et al., 2024; Hua et al., 2024a; Zeng et al., 2025), long-horizon GUI control remains brittle (Koh et al., 2024; Lu et al., 2025; Rawles et al., 2023; Xie et al., 2024; Zhou et al., 2023). Agents have to preserve task state across many interface transitions, where crucial evidence can be local, transient, or unavailable in later screenshots, such as a selected widget state, a temporary menu option, or an earlier instruction needed for a later decision. As trajectories grow longer, these missed cues accumulate, causing agents to forget constraints, lose track of progress, or repeat ineffective actions. This failure mode appears in both cross-app mobile environments (Lu et al., 2025; Rawles et al., 2023) and multimodal web settings (Deng et al., 2023), suggesting a fundamental paradigm shift in GUI agent design: the primary bottleneck is no longer single-step visual understanding, but rather the active management of long-term multimodal state.

Existing GUI agents often address long-horizon interaction through passive history conditioning (Gao et al., 2025; Wang et al., 2024a; Xu et al., 2026; 2025a). However, longer histories or text-only memory representations do not necessarily provide decision-useful context, and may introduce

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.

redundant or distracting information. In long GUI trajectories, useful evidence is sparse and unevenly distributed: some past steps only reflect routine transitions, while others encode task constraints, completed subgoals, or localized visual cues that may no longer be visible in the current screenshot. This suggests that long-horizon GUI control is better viewed as a multimodal memory-control problem rather than a pure context-length problem. Effective agents should decide when to update memory, what to preserve, how to compress interaction history, and when to retrieve past evidence for future decisions.

To address this challenge, we introduce **MEMENTOGUI**, a plug-in agentic multimodal memory-control framework for long-horizon GUI agents. MEMENTOGUI augments a frozen GUI backbone with a learned memory controller rather than finetuning the action policy itself. The controller maintains memory at two complementary timescales: working memory for evolving in-task state and episodic memory for reusable experience from prior interactions. At each step, the controller transforms relevant interaction history into structured multimodal context, including concise event summaries and localized visual references. The frozen GUI backbone then predicts actions from the current screenshot with the memory context, turning interaction history from passive context replay into a decision-oriented control layer.

Trained with large-scale supervision automatically curated from computer-use trajectories, MEMENTOGUI consistently improves frozen GUI backbones across GUI-Odyssey (Lu et al., 2025), Multimodal-Mind2Web (Deng et al., 2023), and our MEMENTOGUI-BENCH. Beyond standard GUI metrics, we further evaluate long-horizon behavior with memory-aware metrics that measure semantic action matching, task progress, and memory consistency. For example, on GUI-Odyssey with UI-Venus-1.5-8B, MEMENTOGUI improves action matching from 54.58 to 68.32 and trajectory success from 1.29 to 3.57, outperforming no-history, history-replay, and text-only memory baselines. These results support our central hypothesis that learning to control multimodal memory is more effective than relying on longer raw interaction histories or text-only memory representations for long-horizon GUI agents. Our contributions are summarized as follows:

- We propose MEMENTOGUI, a plug-in online multimodal agent memory framework that reframes long-horizon GUI control from raw history conditioning to active memory management. MEMENTOGUI augments frozen GUI backbones with a learned controller that actively manages working and episodic memory, enabling agents to preserve and retrieve decision-relevant multimodal state without finetuning the underlying GUI action model.
- We develop an automatic data curation pipeline from PSAI computer-use trajectories to provide scalable supervision for memory control. The pipeline converts raw interactions into training signals for step processing, working-memory compression, episodic memory writing, and episodic memory selection, enabling MEMENTOGUI to learn memory operations with minimal trajectory-level annotation.
- We introduce MEMENTOGUI-BENCH, a benchmark for memory-dependent long-horizon GUI decision making, together with memory-aware metrics for semantic action matching, task progress, and memory consistency. Experiments across mobile and web environments show that MEMENTOGUI consistently improves frozen GUI backbones over strong no-history, raw-history, and text-only memory baselines.

2 RELATED WORK

Memory Systems for Autonomous Agents. Recent GUI-agent research has explored memory mechanisms beyond raw interaction history. MGA (Cheng et al., 2025) and adaptive history modeling (Wu et al., 2025) improve within-task state tracking by managing long GUI trajectories more compactly. For cross-task reuse, Chain-of-Memory (Gao et al., 2025), EchoTrail (Li et al., 2025), and HybridAgent (Zhu et al., 2026) store past trajectories as reasoning chains, retrievable traces, or structured knowledge. Other computer-use agents accumulate reusable knowledge through online interaction, demonstrations, or self-improvement, including AppAgentX (Jiang et al., 2025), MobileGPT (Lee et al., 2024), ScaleCUA (Liu et al., 2025b), UI-Mem (Xiao et al., 2026), EvoCUA (Xue et al., 2026), and AppAgent (Zhang et al., 2025). More broadly, autonomous-agent memory has developed around memory streams (Park et al., 2023), verbal replay (Shinn et al., 2023), skill libraries (Wang et al., 2023), and procedural memory (Fang et al., 2025; Wang et al., 2024c). Recent systems further study learned memory control (Hu et al., 2025; Yu et al., 2026), trainable memory

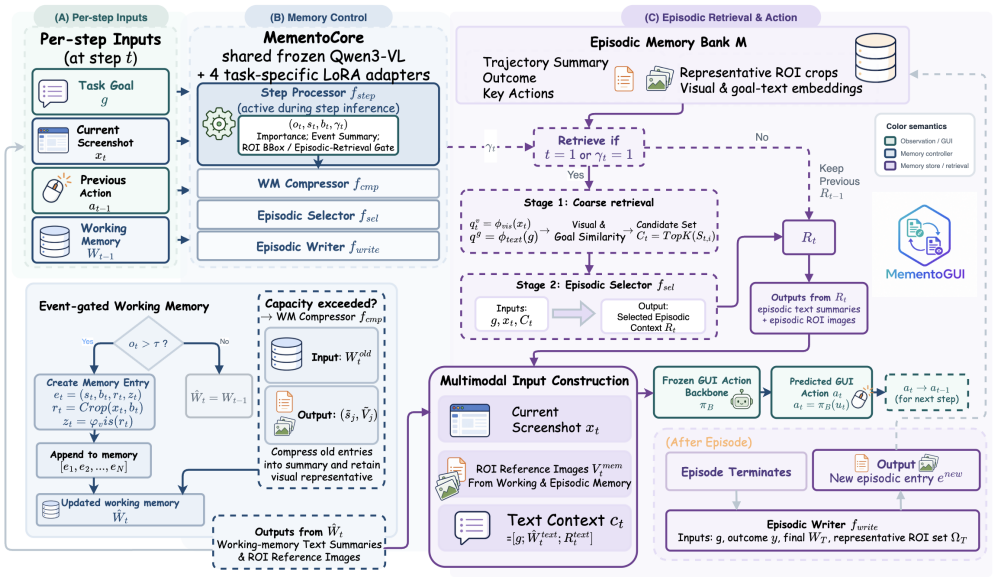


Figure 1: MEMENTOGUI augments a frozen GUI action backbone with multimodal working and episodic memory. It updates, retrieves, and writes memory, then serializes textual summaries and ROI references as multimodal context for GUI action prediction.

operations (Wang et al., 2026a; Zhang et al., 2026), self-organizing memory frameworks (Guo et al., 2026; Xu et al., 2025b), decision-theoretic memory management (Sun et al., 2025a), and efficient compressed or parametric memory representations (Borro et al., 2026; Liu et al., 2026a; Lu et al., 2026). Multimodal memory systems have also begun to store visual trajectories for open-world planning (Li et al., 2024; Wang et al., 2024b), unify visual and episodic memory for video reasoning (Yeo et al., 2025), or distill multimodal experience into reusable programs and lifelong memory (Chen et al., 2025a; Liu et al., 2025a; Sarch et al., 2024). However, they do not fully address long-horizon GUI control, where dense screenshot streams must be selectively compressed, localized visual state changes must be preserved, and memory retrieval must directly support action prediction.

Long-Horizon Challenges in GUI Agents. Recent vision-language models have substantially advanced GUI automation, from visual grounding (Cheng et al., 2024; Hong et al., 2024; Lin et al., 2025) to cross-platform foundation action models (Agashe et al., 2025; Qin et al., 2025; Wu et al., 2024). Recent technical reports and open-source systems, including MAI-UI (Zhou et al., 2025), GUI-Owl-1.5 (Xu et al., 2026), Step-GUI (Yan et al., 2025), and UI-Venus-1.5 (Gao et al., 2026), further improve GUI grounding and navigation across desktop, web, and mobile settings. Complementary efforts further improve efficiency through adaptive perception (Mehrotra et al., 2025), compositional planning (Agashe et al., 2025), and systematic skill acquisition via exploration (Liu et al., 2026b; Sun et al., 2025c). Yet long-horizon tasks remain a dominant failure mode: on benchmarks such as OSWorld (Xie et al., 2024) and WebArena (Zhou et al., 2023), success rates degrade sharply as task length grows, with agents forgetting prior observations, repeating actions, or losing track of sub-goals. The bottleneck has therefore shifted from perception to cross-step state management. To cope with growing context, prior work restructures history as structured prompts or program variables (Tian et al., 2025; Wang et al., 2026b), compresses trajectory tokens (Chen et al., 2025b), or maintains rule-based skill memory for computer control (Tan et al., 2024). These approaches improve how agents reason over history, but leave open what should be retained, when it should be compressed, and how experience should be reused over time.

3 DATA CURATION

To train the memory controller, we curate structured supervision from raw computer-use trajectories in PSAI (Howland et al., 2025). As illustrated in Figure 2, the pipeline first preprocesses the raw video and metadata into frame-level and subgoal-level annotations, then uses the annotations to construct the SFT training data for four memory-control operators. Finally, preference pairs for the

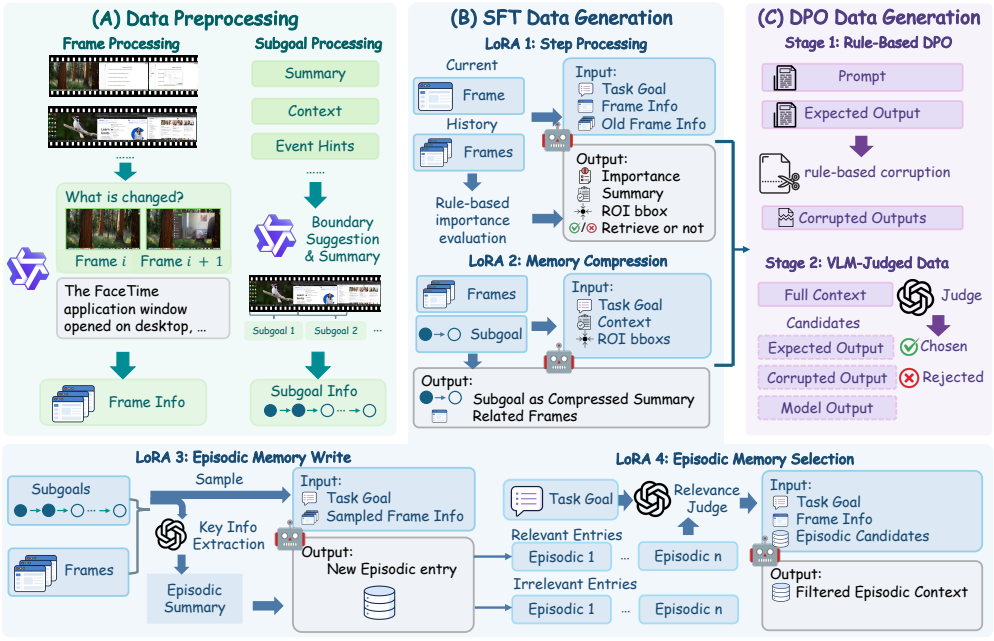


Figure 2: Overview of the MEMENTOGUI data curation pipeline. (A) Raw computer-use videos are parsed into hierarchical frame- and subgoal-level annotations. (B) These annotations are converted into SFT data for four MEMENTOCORE operators: step processing, memory compression, episodic memory writing, and episodic memory selection. (C) Step-processing and memory-compression samples are further corrupted and VLM-filtered to form DPO training pairs.

online memory operators are constructed through rule-based corruption and VLM-judged filtering. We assess annotation quality through human validation on 200 randomly sampled trajectories, of which 197 are judged fully correct; details and additional field-level validation of writing and retrieval decisions are provided in Appendix A.3.

3.1 DATA PREPROCESSING

Each trajectory in the raw computer-use dataset is converted into two annotation streams. Frame-level annotations capture fine-grained interface transitions by comparing adjacent video frames, including action occurrence, event description, input type, key sequence when applicable, and an ROI box for the changed interface region. Subgoal-level annotations capture coarse task progress by segmenting metadata events and interaction logs into chronological semantic units. Additional preprocessing details, including sampling rate, boundary heuristics, and validation rules, are provided in Appendix A.

3.2 MEMORY SUPERVISION CONSTRUCTION

We convert the preprocessed frame and subgoal annotations into operator-specific supervision for MEMENTOCORE. Specifically, we construct four supervised datasets, $\mathcal{D}_{\text{step}}$, $\mathcal{D}_{\text{cmp}}$, $\mathcal{D}_{\text{write}}$, and $\mathcal{D}_{\text{sel}}$, corresponding to the Step Processor, WM Compressor, Episodic Writer, and Episodic Selector. Each example pairs the task goal and relevant multimodal context with a structured target following the schema of the corresponding memory operation. Details of the input formats, output schemas, and validation rules are provided in Appendix B.

For SFT, step-processing examples are constructed from adjacent-frame annotations and subgoal context, with targets including importance scores, event summaries, ROI bounding boxes, and episodic-retrieval activation tags. Compression examples are built by simulating working-memory buffers and asking the model to summarize older entries while preserving representative visual identifiers. Episodic-writing examples convert completed trajectories into compact reusable memories, and episodic-selection examples train the model to filter retrieved candidates by relevance to the current task state. We further construct DPO preference data for the Step Processor and WM Compressor, the two operators most directly tied to online memory quality. Preference pairs are

obtained in two stages: rule-based corruptions create controlled negative outputs, and VLM-judged filtering selects outputs that better preserve task-relevant state, maintain visual grounding, and provide useful downstream context. The resulting preference sets are used for DPO training of the Step Processor and WM Compressor.

4 METHODOLOGY

4.1 THE MEMENTOGUI FRAMEWORK

Given a task goal g and a long-horizon GUI episode $\mathcal{E} = \{x_t\}_{t=1}^{T}$, where x_t denotes the screenshot at step t, the agent predicts actions $\{a_t\}_{t=1}^{T}$ to complete the task. We study a plug-in setting where the GUI action model is a frozen backbone π_B, and MEMENTOGUI augments it with an external multimodal memory controller, MEMENTOCORE. MEMENTOCORE implements a deterministic input-construction step and four learned operators: step processing (including memory writing and retrieval activation), working-memory compression, episodic writing, and episodic selection.

MEMENTOGUI contains an in-episode working memory W_t, a cross-episode episodic memory bank $\mathcal{M}$, and MEMENTOCORE. Working memory tracks transient task state, while episodic memory stores reusable experience from completed episodes. MEMENTOCORE is built by attaching four task-specific LoRA adapters to a shared frozen Qwen3-VL backbone, corresponding to step processing, working-memory compression, episodic writing, and episodic selection. Memory exposure is performed by the input constructor, which serializes textual summaries and ROI references into the native multimodal interface of the GUI backbone. Thus, MEMENTOGUI requires no memory-specific tokens, projection layers, architecture changes, or action-backbone finetuning.

At step t, the Step Processor outputs

$$(o_t, s_t, b_t, \gamma_t) = f_{\text{step}}(g, x_t, a_{t-1}, W_{t-1}), \tag{1}$$

where $o_t \in [0, 1]$ is a write-salience score, s_t is an event summary, b_t is a task-relevant ROI box, and γ_t indicates whether episodic retrieval is needed. This yields a pre-action working memory $\hat{W}_t$. Episodic retrieval is invoked at $t = 1$ and, afterward, only when $\gamma_t = 1$.

The frozen GUI backbone receives

$$\mathbf{u}_t = (x_t, \mathcal{V}_t^{\text{mem}}, c_t), \qquad c_t = [g; \hat{W}_t^{\text{text}}; R_t^{\text{text}}], \tag{2}$$

where $\mathcal{V}_t^{\text{mem}}$ contains selected ROI images from working and episodic memory, and c_t contains the task goal and textual memory summaries. The next action is predicted as

$$a_t = \pi_B(\mathbf{u}_t), \qquad W_t \leftarrow \hat{W}_t. \tag{3}$$

The input $\mathbf{u}_t$ is serialized using the standard multimodal chat template of the backbone, so all memory is consumed as ordinary text and images.

4.2 MEMORY SYSTEM DESIGN

4.2.1 EVENT-GATED WORKING MEMORY

Working memory preserves task-relevant state without replaying the full interaction history. Rather than logging every frame, MEMENTOGUI writes memory only when the current interface may affect future decisions. For a retained step, the memory item is

$$e_t = (s_t, b_t, r_t, z_t), \qquad r_t = \text{Crop}(x_t, b_t), \qquad z_t = \phi_{\text{vis}}(r_t), \tag{4}$$

where r_t is the ROI crop and z_t is used only for memory organization. The update rule is

$$\hat{W}_t = \begin{cases} \text{Append}(W_{t-1}, e_t), & \text{if } o_t > \tau, \\ W_{t-1}, & \text{otherwise}, \end{cases} \tag{5}$$

where the fixed threshold $\tau = 0.3$ converts the learned salience score into a deterministic write decision. The action backbone never receives z_t as a custom token; selected ROI crops are passed as ordinary images.

To control context growth, older uncompressed entries are consolidated when the recent-memory capacity is exceeded:

$$(\tilde{s}_j, \tilde{\mathcal{V}}_j) = f_{\text{cmp}}(g, W_t^{\text{old}}), \tag{6}$$

where $\tilde{s}_j$ is a compact summary and $\tilde{\mathcal{V}}_j$ contains retained visual identifiers resolved into ROI crops during input construction. We pass at most K_{roi} ROI references to the backbone from compressed blocks and recent entries.

4.2.2 On-Demand Episodic Memory

Episodic memory stores reusable experience across completed episodes. Each entry contains a trajectory summary, metadata such as outcome and key actions, representative ROI crops, and retrieval embeddings. Unlike static retrieval, MementoGUI initializes episodic context at the first step and refreshes it only when $\gamma_t = 1$.

When retrieval is invoked, MementoGUI first performs coarse retrieval using the current screenshot and task goal:

$$s_{t,i} = \lambda_v \cos(q_t^v, m_i^v) + \lambda_g \cos(q^g, m_i^g), \qquad \mathcal{C}_t = \text{TopK}_i(s_{t,i}), \tag{7}$$

where $q_t^v = \phi_{\text{vis}}(x_t)$, $q^g = \phi_{\text{text}}(g)$, and (m_i^v, m_i^g) are visual and goal-text embeddings of episodic entry m_i. The Episodic Selector then filters the coarse candidates:

$$\tilde{R}_t = f_{\text{sel}}(g, x_t, \{m_i\}_{i \in \mathcal{C}_t}), \tag{8}$$

where each m_i includes its summary, metadata, and ROI crops. The episodic context is updated by

$$R_t = \begin{cases} \tilde{R}_t, & \text{if } t = 1 \text{ or } \gamma_t = 1, \\ R_{t-1}, & \text{otherwise.} \end{cases} \tag{9}$$

This two-stage design combines efficient vector retrieval with multimodal relevance filtering, while allowing the accumulated working memory to gate when retrieval is invoked.

After an episode ends, the Episodic Writer converts the trajectory into a compact memory:

$$e^{\text{new}} = f_{\text{write}}(g, y, W_T, \Omega_T), \tag{10}$$

where y is the outcome and Ω_T is the representative ROI set from the final working memory. The new entry is stored in $\mathcal{M}$ with its metadata, embeddings, and ROI crops.

4.3 Training MementoCore

We train the four LoRA adapters of MementoCore as structured memory-control tasks on top of a shared frozen Qwen3-VL (Bai et al., 2025) backbone. The supervised datasets $\mathcal{D}_{\text{step}}$, $\mathcal{D}_{\text{cmp}}$, $\mathcal{D}_{\text{write}}$, and $\mathcal{D}_{\text{sel}}$ are produced by the data curation pipeline in Section 3.2. For each operator $k \in \{\text{step}, \text{cmp}, \text{write}, \text{sel}\}$ with LoRA parameters α_k and frozen backbone parameters θ, we minimize

$$\mathcal{L}_{\text{SFT}}^{(k)} = -\mathbb{E}_{(x,y)\sim\mathcal{D}_k} \log p_{\theta,\alpha_k}(y \mid x). \tag{11}$$

We further apply DPO to the Step Processor and WM Compressor using preference sets $\mathcal{P}_{\text{step}}$ and $\mathcal{P}_{\text{cmp}}$, since these operators directly trade off informativeness against context budget. The Episodic Writer and Selector have direct supervised targets and are trained with SFT only. For $k \in \{\text{step}, \text{cmp}\}$, DPO is initialized from the SFT adapter α_k^{SFT}, with reference policy $p_{\text{ref},k} = p_{\theta,\alpha_k^{\text{SFT}}}$. Given $(x, y^+, y^-) \sim \mathcal{P}_k$, we optimize

$$\mathcal{L}_{\text{DPO}}^{(k)} = -\mathbb{E} \log \sigma \left[\beta \log \frac{p_{\theta,\alpha_k}(y^+ \mid x)\, p_{\text{ref},k}(y^- \mid x)}{p_{\text{ref},k}(y^+ \mid x)\, p_{\theta,\alpha_k}(y^- \mid x)} \right]. \tag{12}$$

4.4 Benchmarking Long-Horizon GUI Agents

MementoGUI-Bench. We construct MementoGUI-Bench, an offline benchmark derived from PSAI computer-use videos (Howland et al., 2025) for memory-dependent GUI decision making. It contains 200 trajectories with 6,953 steps, averaging 34.8 steps per trajectory: 80 are used for evaluation and 120 only to construct the episodic-memory bank. Before sample construction, we split all PSAI data by original trajectory ID, keeping controller training, evaluation, and bank trajectories disjoint. Task, app, UI, and workflow overlap is allowed: this is an in-domain episodic-reuse setting, not an out-of-distribution benchmark.

Table 1: Quantitative results on three GUI benchmarks. We evaluate history and memory augmentation strategies on four frozen open-source GUI backbones, with closed-source generalist MLLMs included as direct-prompting baselines. Metrics include AMS/Traj. SR for GUI-Odyssey, Step SR for MM-Mind2Web, and VAM/TPS/MCS for MementoGUI-Bench.

Backbone	Method	GUI-Odyssey		MM-Mind2Web	MementoGUI-Bench		
		AMS ↑	Traj. SR ↑	Step SR ↑	VAM ↑	TPS ↑	MCS ↑
Proprietary generalist MLLM agents							
GPT-5.5 (Singh et al., 2026)	Direct Prompting	54.46	2.02	18.81	1.95	2.00	2.86
Gemini-3.1-Pro (Google DeepMind, 2026)	Direct Prompting	60.62	1.81	22.97	2.18	2.67	2.75
Open-source frozen GUI backbones							
UI-Venus-1.5-8B (Gao et al., 2026)	No History	54.58	1.29	5.29	**1.80**	2.00	0.00
	Pred. Hist. All	66.31	2.33	7.57	1.58	2.29	5.36
	Text Summary Memory	62.18	2.12	11.66	1.58	3.38	3.08
	Working Memory	67.69	2.69	11.80	1.41	4.63	7.00
	Working + Episodic Memory	**68.32**	**3.57**	**12.60**	1.67	**5.16**	**7.14**
MAI-UI-8B (Zhou et al., 2025)	No History	35.70	0.36	12.53	1.81	2.00	0.00
	Pred. Hist. All	44.79	1.35	13.44	1.54	2.11	7.55
	Text Summary Memory	38.33	0.62	13.28	1.63	3.00	3.23
	Working Memory	49.08	1.97	14.61	1.76	5.13	**8.18**
	Working + Episodic Memory	**49.31**	**2.12**	**14.67**	**1.88**	**5.36**	8.13
GUI-Owl-1.5-8B (Xu et al., 2026)	No History	40.15	0.16	**17.78**	1.60	2.00	0.00
	Pred. Hist. All	38.88	0.47	13.56	1.04	2.00	5.25
	Text Summary Memory	45.45	0.62	14.14	1.51	3.05	3.10
	Working Memory	48.25	1.40	15.53	1.79	3.38	7.39
	Working + Episodic Memory	**49.45**	**1.71**	16.42	**1.82**	**4.18**	**7.73**
GUI-Owl-1.5-32B (Xu et al., 2026)	No History	45.73	0.57	18.98	2.10	2.00	0.00
	Pred. Hist. All	49.02	1.55	16.48	1.83	2.11	4.73
	Text Summary Memory	47.21	0.72	17.00	2.37	3.40	3.40
	Working Memory	51.36	2.33	17.71	2.50	4.24	7.85
	Working + Episodic Memory	**55.17**	**2.59**	**19.12**	**2.89**	**4.31**	**8.30**

Semantic and Memory-Aware Evaluation Framework. Reference-based GUI evaluation is standardized but incomplete for long-horizon tasks, where multiple action paths may be valid and decision quality depends on accumulated state. We therefore report VLM-based metrics alongside standard reference-based scores. **VLM-based Action Match (VAM)** measures whether a predicted action realizes the immediate intent inferred by the judge from the current screenshot and reference ROI/action, conditioned on pre-action memory; the subgoal is not a ground-truth annotation. **Task Progress Score (TPS)** evaluates whether the predicted sequence moves the task forward without loops, regressions, or stalling. **Memory Consistency Score (MCS)** assesses whether the memory state evolves consistently with task progress, including prior selections, completed subgoals, user constraints, and retrieved episodic experience. These offline metrics cannot verify task completion or alternative subgoal branches (Appendix E).

5 Experiments

5.1 Experimental Setup

Implementation Details. All GUI backbones are frozen during evaluation (runtime settings: Appendix C.3). We evaluate four open-source GUI models: UI-Venus-1.5-8B (Gao et al., 2026), MAI-UI-8B (Zhou et al., 2025), GUI-Owl-1.5-8B, and GUI-Owl-1.5-32B (Xu et al., 2026). MementoGUI is used as a plug-in memory controller that injects working- and episodic-memory context into the backbone prompt without backbone finetuning, using controllers trained as described in Sec. 4.3. We also evaluate GPT-5.5 (Singh et al., 2026) and Gemini-3.1-Pro (Google DeepMind, 2026) as API-based MLLM agents under the same observation, instruction, and action format. Latency is measured on NVIDIA H100 GPUs under the same inference setup and includes memory-controller inference, retrieval, prompt construction, and GUI-backbone inference.

Evaluation Metrics. We report standard offline GUI metrics following each benchmark protocol, including action-matching score (AMS), step success rate, and trajectory-level success rate. For finer-grained analysis, we additionally use VAM, TPS, and MCS to measure semantic action matching, task-progress plausibility, and consistency with accumulated task state/memory context, respectively, with Gemini-3.1-Pro as the VLM judge. For API-based and scaling experiments, we further report trajectory-level inference time and token usage to quantify compute and context cost. For action-only results, see Appendix E.3.

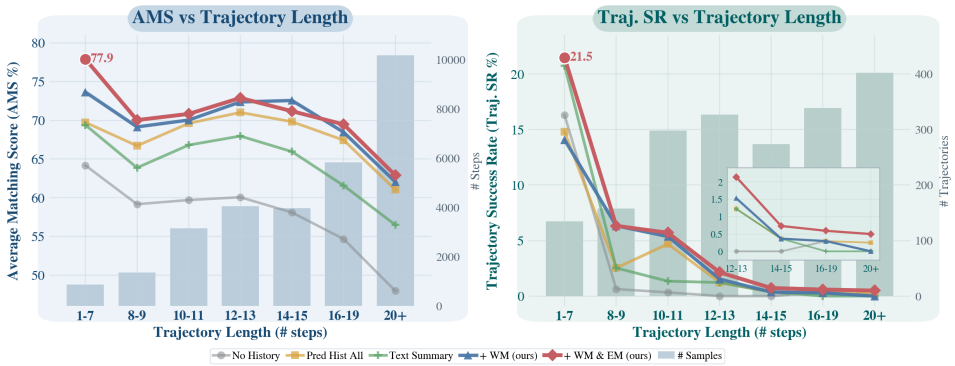


Figure 3: GUI-Odyssey performance by trajectory length on UI-Venus-1.5-8B.

5.2 Quantitative Results

Table 1 reports results on GUI-Odyssey (Lu et al., 2025), MM-Mind2Web (Deng et al., 2023), and MementoGUI-Bench. MementoGUI consistently improves frozen open-source GUI backbones over no-history, predicted-history, and text-summary baselines. For example, on GUI-Odyssey with UI-Venus-1.5-8B, working memory raises AMS from 54.58 to 67.69 and trajectory success from 1.29 to 2.69; adding episodic memory further improves them to 68.32 and 3.57. Similar gains across MAI-UI-8B and GUI-Owl variants confirm its effectiveness as plug-in memory augmentation. Figure 3 shows stronger AMS across trajectory-length bins and higher trajectory success than history-based and text-only memory baselines, especially when working and episodic memory are combined. Figure 4 shows that larger episodic memory banks generally improve trajectory success, suggesting that reusable experience mainly benefits long-horizon completion. Working memory drives the gains; episodic benefits depend on relevant experience and vary across metrics. Table 2 further shows that the same working-memory context can augment proprietary MLLM agents in a stateless single-step setting.

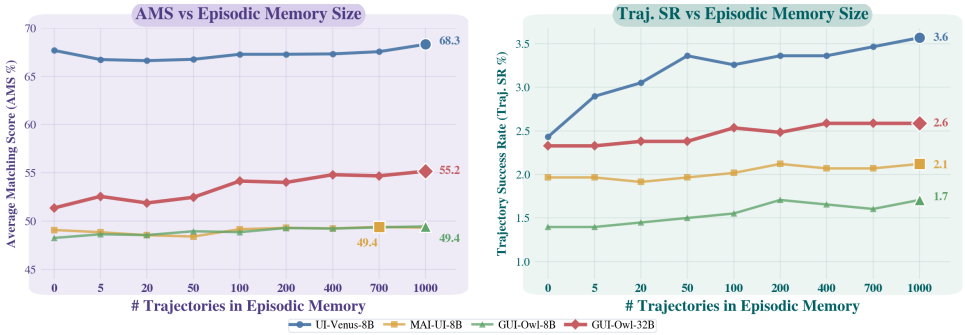


Figure 4: Effect of episodic memory bank size on GUI-Odyssey across frozen GUI backbones.

Table 2: Plug-in working-memory augmentation for API-based MLLM agents. Working Memory replaces native conversation history with stateless single-step inference conditioned on the current screenshot, task instruction, and MementoGUI memory context.

Backbone	Method	GUI-Odyssey				MementoGUI-Bench				
		AMS ↑	Traj. SR ↑	Token	Time	VAM ↑	TPS ↑	MCS ↑	Token	Time
GPT-5.5	Direct Prompting	54.46	2.02	29,974	66.28 s	1.95	2.00	2.86	80,066	133.15 s
	Working Memory	55.73$_{+2.33\%}$	2.07$_{+2.48\%}$	37,798	80.73 s	1.63$_{-16.41\%}$	2.57$_{+28.50\%}$	6.57$_{+129.72\%}$	66,718	174.69 s
Gemini-3.1-Pro	Direct Prompting	60.62	1.81	26,100	58.44 s	2.18	2.67	2.75	75,769	125.37 s
	Working Memory	61.94$_{+2.18\%}$	1.97$_{+8.84\%}$	43,918	78.91 s	2.59$_{+18.81\%}$	3.71$_{+38.95\%}$	7.22$_{+162.55\%}$	106,237	160.82 s

5.3 Ablation Study

Table 3 studies whether working-memory gains come only from learned text summarization or also require visual grounding. *WM w/o Visual Memory* uses the same learned memory-writing and compression controller as full working memory but removes ROI reference images from the memory context, isolating the contribution of localized GUI visual evidence. Table 4 studies how episodic memories should be selected before injection. *Random Episodic Context* controls for adding unrelated past experience, *Single-stage Retrieval* directly uses the top embedding-retrieved

episode, and *Two-stage Retrieval* corresponds to our full WM+EM setting with learned relevance selection. Across both backbones, removing visual memory or learned episodic selection consistently degrades performance, confirming that MEMENTOGUI benefits from both ROI-level grounding and filtered episodic experience rather than merely adding more context. Prompt-only and shared-LoRA comparisons are in Appendix G.

Table 3: Ablation of visual grounding in working memory. *WM w/o Visual Memory* refers to using the learned memory control but removing ROI reference images from the memory context, while full *Working Memory* uses both textual summaries and selected ROI images.

Method	GUI-Odyssey		MementoGUI-Bench		
	AMS ↑	Traj. SR ↑	VAM ↑	TPS ↑	MCS ↑
Backbone: UI-Venus-1.5-8B (Gao et al., 2026)					
Text Summary Memory	62.18	2.12	**1.58**	3.38	3.08
WM w/o Visual Memory	64.22	2.48	1.53	4.50	6.68
Working Memory	**67.69**	**2.69**	1.41	**4.63**	**7.00**
Backbone: GUI-Owl-1.5-8B (Xu et al., 2026)					
Text Summary Memory	45.45	0.62	1.51	3.05	3.10
WM w/o Visual Memory	47.58	1.19	1.76	3.33	5.78
Working Memory	**48.25**	**1.40**	**1.79**	**3.38**	**7.39**

Table 4: Ablation of episodic retrieval. *Single-stage Retrieval* injects the top embedding-retrieved episode, while *Two-stage Retrieval* applies learned relevance selection before memory injection.

Method	GUI-Odyssey		MementoGUI-Bench		
	AMS ↑	Traj. SR ↑	VAM ↑	TPS ↑	MCS ↑
Backbone: UI-Venus-1.5-8B (Gao et al., 2026)					
WM Only	67.69	2.69	1.41	4.63	7.00
Random Episodic Context	64.46	2.22	1.31	4.48	6.35
Single-stage Retrieval	64.40	2.33	1.62	4.79	6.65
Two-stage Retrieval	**68.32**	**3.57**	**1.67**	**5.16**	**7.14**
Backbone: GUI-Owl-1.5-8B (Xu et al., 2026)					
WM Only	48.25	1.40	1.79	3.38	7.39
Random Episodic Context	45.66	1.29	1.72	3.48	6.83
Single-stage Retrieval	48.08	1.19	1.69	3.89	7.51
Two-stage Retrieval	**49.45**	**1.71**	**1.82**	**4.18**	**7.73**

5.4 SCALING MEMENTOCORE

Table 5 studies the effect of memory-controller scale while keeping the memory architecture and frozen GUI action backbone fixed. We compare 2B, 4B, and 8B MEMENTOCORE variants under working-memory-only and working-plus-episodic-memory settings and report both GUI performance and end-to-end trajectory latency. Increasing controller capacity generally improves long-horizon decision support, especially in the working-plus-episodic-memory setting. The 8B controller achieves the strongest results on several key metrics, including action matching for both UI-Venus-1.5-8B and GUI-Owl-1.5-8B in GUI-Odyssey, as well as VAM for GUI-Owl-1.5-8B in MEMENTOGUI-BENCH. Episodic memory further provides complementary gains over working memory alone in most settings, suggesting that retrieved past evidence improves decisions beyond compressed in-task state. These improvements come with additional latency in some cases but require no finetuning of the underlying action model. Overall, the results indicate that MEMENTOGUI can scale as a plug-in memory layer by replacing the controller with stronger variants.

Table 5: We vary the memory-controller size with fixed memory architecture and frozen GUI backbones, comparing working memory with working-plus-episodic memory. Results report GUI performance and end-to-end trajectory latency on GUI-Odyssey and MEMENTOGUI-BENCH.

Backbone	#Params	Method	GUI-Odyssey			MementoGUI-Bench			
			AMS ↑	Traj. SR ↑	Time	VAM ↑	TPS ↑	MCS ↑	Time
UI-Venus-1.5-8B (Gao et al., 2026)	2B	Working Memory	67.69	2.69	41.98 s	1.41	4.63	7.00	86.91 s
		Working + Episodic Memory	68.32$_{+0.93\%}$	3.57$_{+32.71\%}$	51.68 s	1.67$_{+18.44\%}$	5.16$_{+11.45\%}$	7.14$_{+2.00\%}$	100.84 s
	4B	Working Memory	67.82	2.74	61.95 s	1.53	4.84	7.36	124.60 s
		Working + Episodic Memory	68.84$_{+1.50\%}$	3.20$_{+16.79\%}$	60.76 s	1.44$_{-5.88\%}$	4.90$_{+1.24\%}$	7.71$_{+4.76\%}$	140.63 s
	8B	Working Memory	67.38	2.85	72.47 s	1.60	4.83	7.39	105.52 s
		Working + Episodic Memory	69.83$_{+3.64\%}$	3.36$_{+17.89\%}$	56.20 s	1.63$_{+1.87\%}$	4.96$_{+2.69\%}$	7.96$_{+7.71\%}$	124.78 s
GUI-Owl-1.5-8B (Xu et al., 2026)	2B	Working Memory	48.25	1.40	49.02 s	1.79	3.38	7.39	83.28 s
		Working + Episodic Memory	49.45$_{+2.49\%}$	1.71$_{+22.14\%}$	54.57 s	1.82$_{+1.68\%}$	4.18$_{+23.67\%}$	7.73$_{+4.60\%}$	94.89 s
	4B	Working Memory	48.49	1.60	60.81 s	1.64	3.90	7.58	114.37 s
		Working + Episodic Memory	49.05$_{+1.15\%}$	1.71$_{+6.88\%}$	61.21 s	1.87$_{+14.02\%}$	3.68$_{-5.64\%}$	7.53$_{-0.66\%}$	114.96 s
	8B	Working Memory	49.47	1.81	64.45 s	1.65	4.14	7.50	94.49 s
		Working + Episodic Memory	50.70$_{+2.49\%}$	1.81$_{+0.00\%}$	55.24 s	2.18$_{+32.12\%}$	4.58$_{+10.63\%}$	7.75$_{+3.33\%}$	107.15 s

6 CONCLUSION

We introduced MEMENTOGUI, a plug-in online multimodal memory-control framework for long-horizon GUI agents. It enables frozen GUI backbones to update, preserve, compress, and retrieve decision-relevant multimodal state. We further developed an automatic data curation pipeline from PSAI computer-use trajectories and introduced MEMENTOGUI-BENCH, a benchmark for memory-dependent long-horizon GUI decision making. Across mobile and web environments, MEMENTOGUI consistently improves frozen GUI backbones over strong history- and memory-based baselines. Ablations show that localized visual evidence and learned episodic selection provide complementary gains, and scaling experiments suggest that stronger memory controllers can further improve long-horizon decision support.

REPRODUCIBILITY STATEMENT

We will release all source code and model checkpoints necessary to reproduce the results reported in this paper. We provide configuration files in Appendix.

AI USE STATEMENT

Generative Al was used to assist in outlining and language refinement for this work. All final arguments, analysis, factual verification, and editorial decisions remain solely the responsibility of the author.

REFERENCES

Saaket Agashe, Kyle Wong, Vincent Tu, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent s2: A compositional generalist-specialist framework for computer use agents. *arXiv preprint arXiv:2504.00906*, 2025.

Niccolo Avogaro, Nayanika Debnath, Li Mi, Thomas Frick, Junling Wang, Zexue He, Hang Hua, Konrad Schindler, and Mattia Rigotti. Sparc: Separating perception and reasoning circuits for test-time scaling of vlms. *arXiv preprint arXiv:2602.06566*, 2026.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Luiz C Borro, Luiz AB Macarini, Gordon Tindall, Michael Montero, and Adam B Struck. Memori: A persistent memory layer for efficient, context-aware llm agents. *arXiv preprint arXiv:2603.19935*, 2026.

Chunliang Chen, Ming Guan, Xiao Lin, Jiaxu Li, Luxi Lin, Qiyi Wang, Xiangyu Chen, Jixiang Luo, Changzhi Sun, Dell Zhang, et al. Telemem: Building long-term and multimodal memory for agentic ai. *arXiv preprint arXiv:2601.06037*, 2025a.

Gongwei Chen, Xurui Zhou, Rui Shao, Yibo Lyu, Kaiwen Zhou, Shuai Wang, Wentao Li, Yinchuan Li, Zhongang Qi, and Liqiang Nie. Less is more: Empowering gui agent with context-aware simplification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5901–5911, 2025b.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Li YanTao, Jianbing Zhang, and Zhiyong Wu. Seeclick: Harnessing gui grounding for advanced visual gui agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9313–9332, 2024.

Weihua Cheng, Ersheng Ni, Wenlong Wang, Yifei Sun, Junming Liu, Wangyu Shen, Yirong Chen, Botian Shi, and Ding Wang. Mga: Memory-driven gui agent for observation-centric interaction. *arXiv preprint arXiv:2510.24168*, 2025.

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

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Memp: Exploring agent procedural memory. *arXiv preprint arXiv:2508.06433*, 2025.

Changlong Gao, Zhangxuan Gu, Yulin Liu, Xinyu Qiu, Shuheng Shen, Yue Wen, Tianyu Xia, Zhenyu Xu, Zhengwen Zeng, Beitong Zhou, et al. Ui-venus-1.5 technical report. *arXiv preprint arXiv:2602.09082*, 2026.

Xinzge Gao, Chuanrui Hu, Bin Chen, and Teng Li. Chain-of-memory: Enhancing gui agents for cross-application navigation. *arXiv preprint arXiv:2506.18158*, 2025.

Google DeepMind. Gemini 3.1 Pro Model Card. https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-1-Pro-Model-Card.pdf, February 2026. Published February 2026; updated 19 February 2026.

Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. Navigating the digital world as humans do: Universal visual grounding for gui agents. *arXiv preprint arXiv:2410.05243*, 2024.

Ziliang Guo, Ziheng Li, and Zhiyu Li. Memfactory: Unified inference & training framework for agent memory. *arXiv preprint arXiv:2603.29493*, 2026.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, et al. Cogagent: A visual language model for gui agents. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14281–14290, 2024.

Anais Howland, Ashwin Thinnappan, and Jameel Shahid Mohammed. Computer use data - paradigm shift ai, 2025. URL https://huggingface.co/datasets/anaisleila/computer-use-data-psai.

Mengkang Hu, Tianxing Chen, Qiguang Chen, Yao Mu, Wenqi Shao, and Ping Luo. 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)*, pp. 32779–32798, 2025.

Yushi Hu, Hang Hua, Zhengyuan Yang, Weijia Shi, Noah A. Smith, and Jiebo Luo. Promptcap: Prompt-guided image captioning for vqa with gpt-3. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2951–2963, 2023. URL https://api.semanticscholar.org/CorpusID:257637217.

Hang Hua, Jing Shi, Kushal Kafle, Simon Jenni, Daoan Zhang, John P. Collomosse, Scott Cohen, and Jiebo Luo. Finematch: Aspect-based fine-grained image and text mismatch detection and correction. In *European Conference on Computer Vision*, 2024a. URL https://api.semanticscholar.org/CorpusID:269303150.

Hang Hua, Yunlong Tang, Ziyun Zeng, Liangliang Cao, Zhengyuan Yang, Hangfeng He, Chenliang Xu, and Jiebo Luo. Mmcomposition: Revisiting the compositionality of pre-trained vision-language models. *arXiv preprint arXiv:2410.09733*, 2024b.

Hang Hua, Qing Liu, Lingzhi Zhang, Jing Shi, Soo Ye Kim, Zhifei Zhang, Yilin Wang, Jianming Zhang, Zhe Lin, and Jiebo Luo. Finecaption: Compositional image captioning focusing on wherever you want at any granularity. In *Proceedings of the computer vision and pattern recognition conference*, pp. 24763–24773, 2025a.

Hang Hua, Yunlong Tang, Chenliang Xu, and Jiebo Luo. V2xum-llm: Cross-modal video summarization with temporal prompt instruction tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 3599–3607, 2025b.

Hang Hua, Ziyun Zeng, Yizhi Song, Yunlong Tang, Liu He, Daniel G. Aliaga, Wei Xiong, and Jiebo Luo. Mmig-bench: Towards comprehensive and explainable evaluation of multi-modal image generation models. *ArXiv*, abs/2505.19415, 2025c. URL https://api.semanticscholar.org/CorpusID:278905580.

Wenjia Jiang, Yangyang Zhuang, Chenxi Song, Xu Yang, Joey Tianyi Zhou, and Chi Zhang. Appagentx: Evolving gui agents as proficient smartphone users. *arXiv preprint arXiv:2503.02268*, 2025.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 881–905, 2024.

Sunjae Lee, Junyoung Choi, Jungjae Lee, Munim Hasan Wasi, Hojun Choi, Steve Ko, Sangeun Oh, and Insik Shin. Mobilegpt: Augmenting llm with human-like app memory for mobile task automation. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, pp. 1119–1133, 2024.

Weixian Lei, Difei Gao, and Mike Zheng Shou. Grounding multimodal large language model in gui world. In *The Thirteenth International Conference on Learning Representations*, 2025.

Runze Li, Yuwen Zhai, Bo Xu, LiWu Xu, Nian Shi, Wei Zhang, Ran Lin, and Liang Wang. Echotrail-gui: Building actionable memory for gui agents via critic-guided self-exploration. *arXiv preprint arXiv:2512.19396*, 2025.

Zaijing Li, Yuquan Xie, Rui Shao, Gongwei Chen, Dongmei Jiang, and Liqiang Nie. Optimus-1: Hybrid multimodal memory empowered agents excel in long-horizon tasks. *Advances in neural information processing systems*, 37:49881–49913, 2024.

Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Stan Weixian Lei, Lijuan Wang, and Mike Zheng Shou. Showui: One vision-language-action model for gui visual agent. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 19498–19508, 2025.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

Jiaqi Liu, Yaofeng Su, Peng Xia, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. Simplemem: Efficient lifelong memory for llm agents. *arXiv preprint arXiv:2601.02553*, 2026a.

Jiateng Liu, Zhenhailong Wang, Rushi Wang, Bingxuan Li, Jeonghwan Kim, Aditi Tiwari, Pengfei Yu, Denghui Zhang, and Heng Ji. Osexpert: Computer-use agents learning professional skills via exploration. *arXiv preprint arXiv:2603.07978*, 2026b.

Junming Liu, Yifei Sun, Weihua Cheng, Haodong Lei, Yirong Chen, Licheng Wen, Xuemeng Yang, Daocheng Fu, Pinlong Cai, Nianchen Deng, et al. Memverse: Multimodal memory for lifelong learning agents. *arXiv preprint arXiv:2512.03627*, 2025a.

Zhaoyang Liu, JingJing Xie, Zichen Ding, Zehao Li, Bowen Yang, Zhenyu Wu, Xuehui Wang, Qiushi Sun, Shi Liu, Weiyun Wang, et al. Scalecua: Scaling open-source computer use agents with cross-platform data. *arXiv preprint arXiv:2509.15221*, 2025b.

Quanfeng Lu, Wenqi Shao, Zitao Liu, Lingxiao Du, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang, Kaipeng Zhang, and Ping Luo. Guiodyssey: A comprehensive dataset for cross-app gui navigation on mobile devices. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 22404–22414, 2025.

Sidi Lu, Zhenwen Liang, Dongyang Ma, Yan Wang, Haitao Mi, and Dong Yu. Locas: Your models are principled initializers of locally-supported parametric memories. *arXiv preprint arXiv:2602.05085*, 2026.

Sarthak Mehrotra, Sairam VC Rebbapragada, Mani Hemanth Reddy Bonthu, and Vineeth N Balasubramanian. ishift: Lightweight slow-fast gui agent with adaptive perception. *arXiv preprint arXiv:2512.22009*, 2025.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pp. 1–22, 2023.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, et al. Ui-tars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*, 2025.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL `https://qwen.ai/blog?id=qwen3.5`.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. Androidinthewild: A large-scale dataset for android device control. *Advances in Neural Information Processing Systems*, 36:59708–59728, 2023.

Gabriel Sarch, Lawrence Jang, Michael J Tarr, William W Cohen, Kenneth Marino, and Katerina Fragkiadaki. Vlm agents generate their own memories: Distilling experience into embodied programs of thought. *Advances in Neural Information Processing Systems*, 37:75942–75985, 2024.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2026.

Changzhi Sun, Xiangyu Chen, Jixiang Luo, Dell Zhang, and Xuelong Li. Beyond heuristics: A decision-theoretic framework for agent memory management. *arXiv preprint arXiv:2512.21567*, 2025a.

Guohao Sun, Hang Hua, Jian Wang, Jiebo Luo, Sohail A. Dianat, Majid Rabbani, Raghuveer Rao, and Zhiqiang Tao. Latent chain-of-thought for visual reasoning. *ArXiv*, abs/2510.23925, 2025b. URL https://api.semanticscholar.org/CorpusID:282400907.

Zeyi Sun, Ziyu Liu, Yuhang Zang, Yuhang Cao, Xiaoyi Dong, Tong Wu, Dahua Lin, and Jiaqi Wang. Seagent: Self-evolving computer use agent with autonomous learning from experience. *arXiv preprint arXiv:2508.04700*, 2025c.

Weihao Tan, Wentao Zhang, Xinrun Xu, Haochong Xia, Ziluo Ding, Boyu Li, Bohan Zhou, Junpeng Yue, Jiechuan Jiang, Yewen Li, et al. Cradle: Empowering foundation agents towards general computer control. *arXiv preprint arXiv:2403.03186*, 2024.

Tristan Thrush, Ryan Jiang, Max Bartolo, Amanpreet Singh, Adina Williams, Douwe Kiela, and Candace Ross. Winoground: Probing vision and language models for visio-linguistic compositionality. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5238–5248, 2022.

Shizuo Tian, Hao Wen, Yuxuan Chen, Jiacheng Liu, Shanhui Zhao, Guohong Liu, Ju Ren, Yunxin Liu, and Yuanchun Li. Agentprog: Empowering long-horizon gui agents with program-guided context management. *arXiv preprint arXiv:2512.10371*, 2025.

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

Junyang Wang, Haiyang Xu, Haitao Jia, Xi Zhang, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. Mobile-agent-v2: Mobile device operation assistant with effective navigation via multi-agent collaboration. *Advances in Neural Information Processing Systems*, 37:2686–2710, 2024a.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, et al. Internvl3. 5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025.

Xinyu Wang, Mingze Li, Peng Lu, Xiao-Wen Chang, Lifeng Shang, Jinping Li, Fei Mi, Prasanna Parthasarathi, and Yufei Cui. Infmem: Learning system-2 memory control for long-context agent. *arXiv preprint arXiv:2602.02704*, 2026a.

Zihao Wang, Shaofei Cai, Anji Liu, Yonggang Jin, Jinbing Hou, Bowei Zhang, Haowei Lin, Zhaofeng He, Zilong Zheng, Yaodong Yang, et al. Jarvis-1: Open-world multi-task agents with memory-augmented multimodal language models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(3):1894–1907, 2024b.

Ziwei Wang, Leyang Yang, Xiaoxuan Tang, Sheng Zhou, Dajun Chen, Wei Jiang, and Yong Li. History-aware reasoning for gui agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 36448–36456, 2026b.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. *arXiv preprint arXiv:2409.07429*, 2024c.

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

Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, et al. Os-atlas: A foundation action model for generalist gui agents. *arXiv preprint arXiv:2410.23218*, 2024.

Han Xiao, Guozhi Wang, Hao Wang, Shilong Liu, Yuxiang Chai, Yue Pan, Yufeng Zhou, Xiaoxin Chen, Yafei Wen, and Hongsheng Li. Ui-mem: Self-evolving experience memory for online reinforcement learning in mobile gui agents. *arXiv preprint arXiv:2602.05832*, 2026.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024.

Haiyang Xu, Xi Zhang, Haowei Liu, Junyang Wang, Zhaozai Zhu, Shengjie Zhou, Xuhao Hu, Feiyu Gao, Junjie Cao, Zihua Wang, et al. Mobile-agent-v3. 5: Multi-platform fundamental gui agents. *arXiv preprint arXiv:2602.16855*, 2026.

Ran Xu, Kaixin Ma, Wenhao Yu, Hongming Zhang, Joyce C Ho, Carl Yang, and Dong Yu. Retrieval-augmented gui agents with generative guidelines. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 17877–17886, 2025a.

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*, 2025b.

Taofeng Xue, Chong Peng, Mianqiu Huang, Linsen Guo, Tiancheng Han, Haozhe Wang, Jianing Wang, Xiaocheng Zhang, Xin Yang, Dengchang Zhao, et al. Evocua: Evolving computer use agents via learning from scalable synthetic experience. *arXiv preprint arXiv:2601.15876*, 2026.

Haolong Yan, Jia Wang, Xin Huang, Yeqing Shen, Ziyang Meng, Zhimin Fan, Kaijun Tan, Jin Gao, Lieyu Shi, Mi Yang, et al. Step-gui technical report. *arXiv preprint arXiv:2512.15431*, 2025.

Woongyeong Yeo, Kangsan Kim, Jaehong Yoon, and Sung Ju Hwang. Worldmm: Dynamic multimodal memory agent for long video reasoning. *arXiv preprint arXiv:2512.02425*, 2025.

Yi Yu, Liuyi Yao, Yuexiang Xie, Qingquan Tan, Jiaqi Feng, Yaliang Li, and Libing Wu. Agentic memory: Learning unified long-term and short-term memory management for large language model agents. *arXiv preprint arXiv:2601.01885*, 2026.

Ziyun Zeng, Hang Hua, and Jiebo Luo. Mira: Multimodal iterative reasoning agent for image editing. *arXiv preprint arXiv:2511.21087*, 2025.

Chi Zhang, Zhao Yang, Jiaxuan Liu, Yanda Li, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. Appagent: Multimodal agents as smartphone users. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pp. 1–20, 2025.

Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. Memskill: Learning and evolving memory skills for self-evolving agents. *arXiv preprint arXiv:2602.02474*, 2026.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v (ision) is a generalist web agent, if grounded. *arXiv preprint arXiv:2401.01614*, 2024.

Hanzhang Zhou, Xu Zhang, Panrong Tong, Jianan Zhang, Liangyu Chen, Quyu Kong, Chenglin Cai, Chen Liu, Yue Wang, Jingren Zhou, et al. Mai-ui technical report: Real-world centric foundation gui agents. *arXiv preprint arXiv:2512.22047*, 2025.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.

Sibo Zhu, Wenyi Wu, Kun Zhou, Stephen Wang, and Biwei Huang. Hybrid self-evolving structured memory for gui agents. *arXiv preprint arXiv:2603.10291*, 2026.

A Data Preprocessing Details

We used `Qwen3.5-397B-A17B-FP8` (Qwen Team, 2026) to preprocess our dataset.

A.1 Frame Processing

Frame processing extracts fine-grained action annotations from raw computer-use videos. We first sample frames from each video using `ffmpeg` at 2.0 FPS, then pair adjacent frames, i.e., frame i and frame $i+1$, for transition analysis. Each frame pair is submitted to a VLM, which determines whether the interface changed in a meaningful way and infers the likely user interaction that caused the transition.

For each detected action, the VLM outputs a `has_action` flag, a concise one-sentence transition description, and action-type indicators for mouse and keyboard input. When keyboard interaction is present, the model extracts the corresponding key sequence, such as Ctrl+L, type 'foo', Enter. For localized visual changes, the model also predicts an ROI bounding box over the active UI region in the second frame. The box is first produced in normalized $[0, 1]$ coordinates and then mapped back to absolute pixel coordinates.

Finally, the extracted annotations, including timestamps, action descriptions, action flags, key sequences, and ROI coordinates, are validated for consistency and written to a per-video JSON file. For example, descriptions must be empty when no action is detected, and bounding boxes must be valid before being stored.

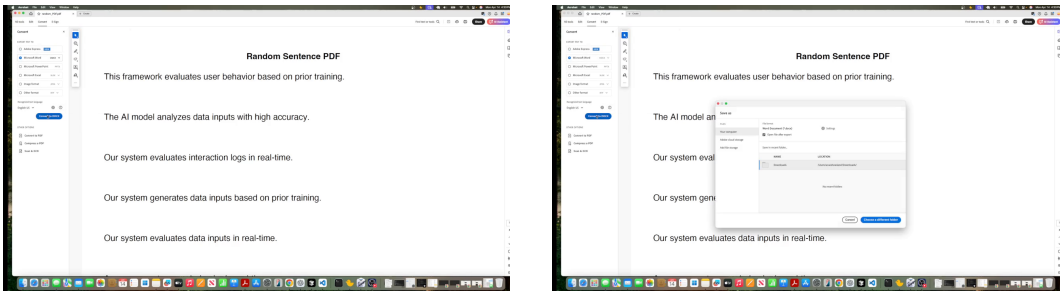


Figure 5: Example adjacent-frame transition used for frame-level annotation. The left frame shows the PDF viewer before the interaction, while the right frame shows the interface after a *Save As* dialog appears. This transition is detected as a meaningful GUI event and converted into a structured low-level action annotation.

Figure 5 and Listing 1 show an example frame-level annotation. The sampled frame pair captures a transition in which a PDF viewer changes from a static document page to a newly opened *Save As* dialog. The VLM identifies this as a meaningful mouse-driven action, summarizes the transition, and localizes the newly appeared dialog as the task-relevant region. The resulting annotation records both normalized and absolute pixel coordinates for this ROI, allowing the same transition to be used as supervision for memory writing and visual grounding.

Listing 1: Example of frame-level step annotation.

```
{
    "frame_i": 44,
    "frame_i1": 45,
    "start_s": 21.5,
    "end_s": 22.0,
```

```
    "description": "A 'Save As' dialog window appeared in the center of
        the screen, indicating the user initiated a save/export action.",
    "has_action": true,
    "has_mouse_action": true,
    "has_keyboard_action": false,
    "bbox_x0_norm": 0.32,
    "bbox_y0_norm": 0.35,
    "bbox_x1_norm": 0.68,
    "bbox_y1_norm": 0.65,
    "bbox_x0_px": 409,
    "bbox_y0_px": 252,
    "bbox_x1_px": 870,
    "bbox_y1_px": 467,
    "key_sequence": ""
}
```

The bounding box in Listing 1 is visualized in Figure 6. The highlighted region covers the active dialog window, which contains the interface state most relevant to the transition. This example illustrates how frame processing converts dense video observations into compact, structured supervision consisting of an event summary, action metadata, and ROI-level visual evidence.

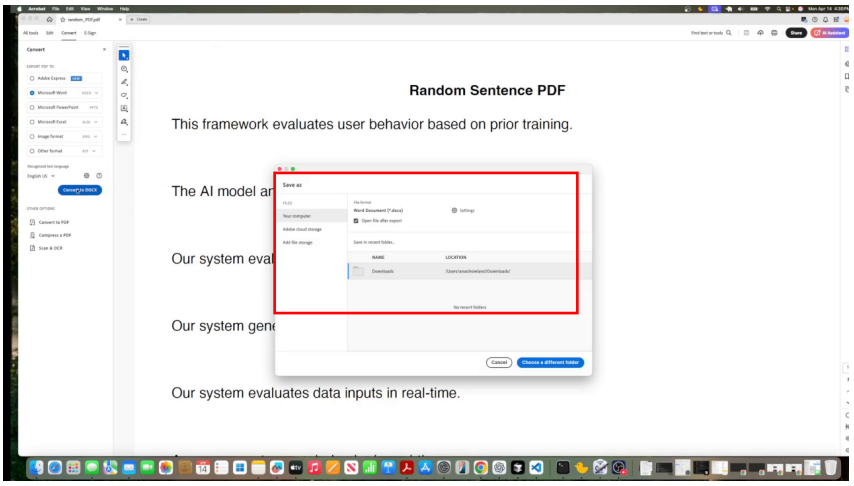


Figure 6: Visualization of the ROI bounding box predicted for the frame-level annotation in Listing 1. The red box localizes the newly appeared *Save As* dialog, which is the task-relevant visual evidence retained for memory supervision.

A.2 Subgoal Processing

Subgoal processing extracts coarse semantic progress from trajectory metadata and user interaction logs. We first parse raw interaction events and normalize their timestamps so that the initial event corresponds to the beginning of the video. The events are then grouped into fixed-size temporal buckets, typically 1-second intervals, where each bucket records an activity signal based on whitelisted interaction types such as clicks, scrolls, DOM snapshots, window focus changes, and cursor movements.

Candidate subgoal boundaries are identified using both event activity and heuristic cues. Strong boundaries are placed at timestamps associated with major UI shifts, such as DOM snapshots and window focus events. Additional boundaries are inserted near the midpoint of long idle gaps, e.g., gaps longer than 3.0 seconds. Near-duplicate boundaries are pruned, and overly short segments are merged to produce a clean set of candidate segments.

An LLM is then prompted with the task metadata, summarized event buckets, and candidate segments to generate a chronological list of major subgoals. Each subgoal contains a start time, an end time,

and a concise 5-to-15-word description of the user's progress. The generated steps are post-processed to enforce monotonicity, non-overlap, and video coverage.

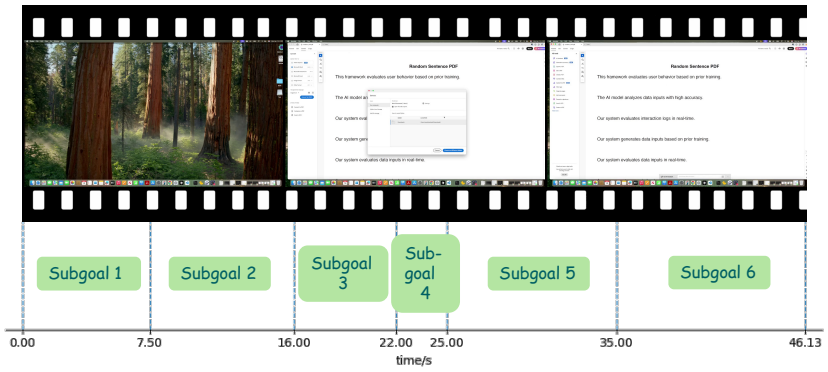


Figure 7: Example subgoal segmentation for a long-horizon GUI trajectory. The top row shows representative video frames, while the bottom timeline aligns six inferred subgoals with their temporal spans. The trajectory progresses from opening Adobe Acrobat and loading the target PDF to exporting it as a Word document, configuring the Save As dialog, waiting for conversion, and viewing the converted file. This example demonstrates how low-level interaction logs are converted into temporally ordered, semantically meaningful subgoal annotations for memory-supervision construction.

Figure 7 illustrates an example subgoal decomposition for a PDF export trajectory. The video is segmented into six consecutive stages, covering opening Adobe Acrobat, loading the target PDF, selecting the export-to-Word tool, configuring the Save As dialog, waiting for the export to complete, and viewing the converted Word document. Listing 2 shows the corresponding structured subgoal annotation.

Listing 2: Example of subgoal annotation.

```
    {
        "start": 0.0,
        "end": 7.5,
        "description": "Open Adobe Acrobat and navigate to the PDF file."
    },
    {
        "start": 7.5,
        "end": 16.0,
        "description": "Open the PDF document titled Random Sentence PDF.
            "
    },
    {
        "start": 16.0,
        "end": 22.0,
        "description": "Select Export PDF tool and choose Word document
            format."
    },
    {
        "start": 22.0,
        "end": 25.0,
        "description": "Configure save options in the Save As dialog
            window."
    },
    {
        "start": 25.0,
        "end": 35.0,
        "description": "Save the file and wait for export process to
            complete."
    },
    {
```

```
        "start": 35.0,
        "end": 46.133,
        "description": "View the converted Word document and close
            overlay window."
        }
```

A.3 Human Validation

To improve the quality of the automatically curated data, we conducted human validation on 200 sampled trajectories with 10 annotators. The sampled trajectories covered frame-level transition annotations, subgoal segmentation, ROI grounding, and memory-control targets used for SFT and preference construction. Annotators were asked to judge whether each trajectory was fully correct in terms of visual faithfulness, temporal consistency, and memory-target validity. In total, 197 out of 200 trajectories were judged as completely correct, indicating that the automatic curation pipeline produces high-quality supervision with only minimal residual noise.

Field-level validation. We additionally validate importance-based writing and retrieval activation. For each field, we sample 100 cases balanced by the model prediction (50 positive and 50 negative). Importance predictions are positive when $o_t > 0.3$ and negative otherwise. Five annotators independently judge whether each step should be written into working memory and whether episodic retrieval should be activated, yielding 500 human judgments per field as reference labels. Table 6 reports the results. This is a prediction-balanced validation sample, not an estimate of the natural prevalence of positive decisions.

Table 6: Field-level human validation of the two Step Processor decisions. Each field has 100 prediction-balanced cases and five independent human judgments per case.

Decision	Accuracy	Precision	Recall	F1
Importance-based writing	94.6%	98.3%	90.8%	94.4%
Retrieval activation	88.6%	84.1%	95.2%	89.3%

B Memory Supervision Construction Details

Adapter	#Train	#DPO pairs	Imgs/ex.
Step Processor	626,119	18,951	1
WM Compressor	195,471	4,431	1–8
Episodic Writer	166,739	—	2–5
Episodic Selector	26,326	—	2–10

Table 7: Training data for the four MementoGUI LoRA adapters. All adapters are trained with SFT. Step Processor and WM Compressor are additionally refined with DPO, while Episodic Writer and Episodic Selector use SFT only.

This appendix provides additional details on how we construct memory-control supervision from the preprocessed PSAI trajectories. The goal is to train **MementoCore** as four structured memory operators rather than a single monolithic memory model. Each operator has its own input context, output schema, and training objective.

B.1 SFT Dataset Construction

Step Processor. The Step Processor is trained from adjacent-frame annotations and subgoal context. Each input contains the task goal, current screenshot, previous action or transition description, and recent working-memory state. The target output is a structured JSON object containing an importance score, a concise event summary, an ROI bounding box, and an episodic-retrieval activation tag. The importance target is generated by a VLM comparing the current and previous screenshots for substantial task-relevant change. The retrieval target is derived deterministically from VLM-produced

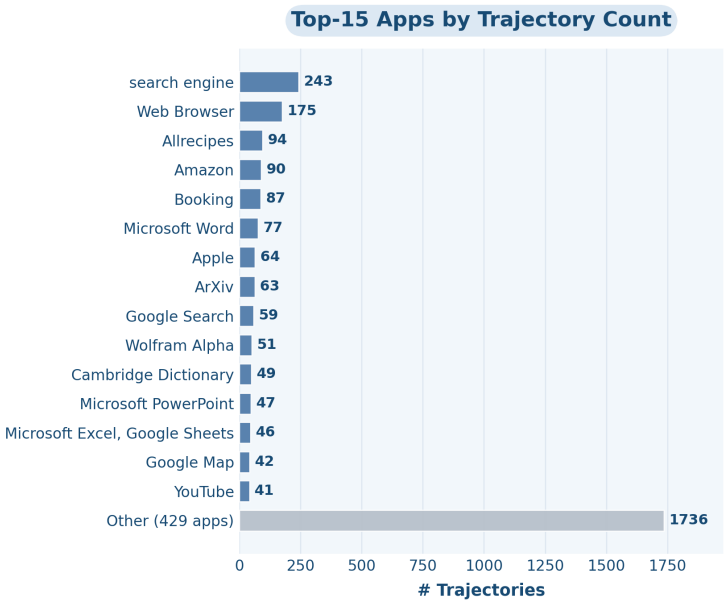


Figure 8: Top-15 applications in the MEMENTOGUI training set by trajectory count. The top-15 together cover only 1,228 trajectories (41%), while the remaining 1,736 trajectories (59%) are spread across 429 long-tail apps, yielding broad application diversity.



Figure 9: Word cloud of the 429 long-tail applications in the training set (1,736 trajectories). The heavy tail illustrates that MEMENTOGUI training data exercises memory dynamics across a wide range of real-world desktop software.

subgoal annotations based on trajectory-level video understanding and interaction metadata: $\gamma_t = 1$ when a completed subgoal transitions to a new, semantically unrelated subgoal. At inference, the Step Processor predicts both fields in its structured output. The ROI bounding box identifies localized visual evidence to retain when the step is written into working memory.

WM Compressor. To construct compression examples, we simulate working-memory buffers by accumulating retained step entries along each trajectory. Once the number of recent uncompressed entries exceeds the compression window, the oldest entries are grouped into a compression instance. The input consists of the task goal and a sequence of older working-memory entries, including their textual summaries and visual identifiers. The target output is a compact summary together with a selected list of visual identifiers that should remain available as ROI references after compression. This trains the compressor to preserve task-relevant state while reducing context length.

Episodic Writer. For each completed episode, we construct one trajectory-level writing example. The input contains the task goal, task outcome, subgoal summaries, final working-memory state, and representative ROI references sampled from the trajectory. The target output is a reusable episodic memory entry, including a trajectory-level summary, key actions or subgoals, outcome metadata, and visual identifiers for representative evidence. This converts a raw interaction trajectory into a compact multimodal memory item for future reuse.

Episodic Selector. For episodic selection, we build candidate sets by retrieving or sampling past episodic entries for a given task state. Each input contains the task goal, current screenshot, and a set of candidate episodic memories. The target output identifies the entries that are relevant to the current task and filters out irrelevant or weakly related candidates. This supervision trains the selector to act as a relevance filter after coarse embedding-based retrieval.

Validation. Across all four operators, outputs are validated before training. We check schema correctness, chronological consistency, valid ROI coordinates, resolvable visual identifiers, and consistency between textual summaries and available visual evidence. Invalid or inconsistent examples are discarded.

B.2 DPO Preference Dataset Construction

We construct DPO preference data only for the Step Processor and WM Compressor, since these two operators directly affect online memory updates and downstream context construction. For each operator, we create preference pairs (x, y^+, y^-), where y^+ denotes the preferred memory-control output and y^- denotes a lower-quality alternative.

Rule-based contrastive pairs. The first stage creates controlled negative outputs by corrupting high-quality SFT targets. For the Step Processor, negative outputs may drop important event information, select an incorrect or overly broad ROI, assign an inappropriate importance score, or disable episodic retrieval when prior experience is useful. For the WM Compressor, negative outputs may remove key subgoal information, retain irrelevant visual identifiers, over-compress important state, or hallucinate content unsupported by the trajectory. These controlled corruptions produce preference pairs that emphasize faithfulness, conciseness, and visual grounding.

VLM-judged preference filtering. The second stage uses a VLM judge to refine preference data. We sample candidate outputs from the SFT-initialized adapters and compare them with rule-based variants under the full input context. The judge selects the output that better preserves task-relevant state, maintains ROI-level visual grounding, and provides useful context for later GUI decisions. Preference pairs with invalid JSON, unresolved visual identifiers, inconsistent ROI references, or ambiguous judgments are discarded.

The resulting preference sets $\mathcal{P}_{\text{step}}$ and $\mathcal{P}_{\text{cmp}}$ are used for DPO training. The Episodic Writer and Episodic Selector are trained with SFT only.

C TRAINING DETAILS

C.1 TRAINING DATASET

This section summarizes the training data used to fit the four MEMENTOGUI memory-control adapters. Starting from the preprocessed PSAI computer-use trajectories, we convert each trajectory into operator-specific supervision for step processing, working-memory compression, episodic writing, and episodic selection. These four datasets correspond to different stages of the memory-control pipeline: the Step Processor learns when and what to write into working memory, the WM Compressor learns how to consolidate older memory entries under a context budget, the Episodic Writer learns how to summarize a completed trajectory into reusable cross-episode memory, and the Episodic Selector learns how to filter retrieved past episodes before they are injected into the action model.

The resulting training data is intentionally heterogeneous across adapters, since each memory operator receives different input context and produces a different structured output. Step-processing examples are the most numerous because they are generated at the frame-transition level and therefore provide dense supervision for online memory updates. WM-compression examples are generated from simulated working-memory buffers and contain longer textual context together with multiple visual identifiers. Episodic-writing examples are constructed at the trajectory level and require the model to summarize completed task executions into compact reusable memories. Episodic-selection examples are built from candidate episodic-memory sets and train the model to distinguish task-relevant past experience from unrelated or weakly useful trajectories.

Figures 8 and 9 summarize the application distribution of the training set. Figure 8 shows the top-15 applications by trajectory count, while Figure 9 visualizes the long-tail portion of the data. The distribution is deliberately broad rather than concentrated in a small number of applications: the top-15 applications cover only part of the training set, while the remaining trajectories are spread across hundreds of long-tail desktop and browser applications. This diversity is important for training a memory controller, because the controller should not overfit to a fixed set of UI layouts or application-specific action patterns. Instead, it must learn general memory operations, such as retaining selected values, remembering completed subgoals, preserving transient UI evidence, and retrieving relevant past experience across different software environments.

Table 7 reports the final number of SFT examples, DPO preference pairs, and images per example for each adapter. All four adapters are first trained with supervised fine-tuning. The Step Processor and WM Compressor are additionally refined with DPO because these two operators directly determine the quality of online working memory: overly aggressive writing can introduce redundant context, while overly aggressive compression can remove information needed later. In contrast, the Episodic Writer and Episodic Selector have more direct supervised targets and are trained with SFT only. Overall, the dataset provides dense step-level supervision, longer-range compression supervision, trajectory-level episodic-writing supervision, and retrieval-filtering supervision, enabling MEMENTOGUI to learn memory control as a set of specialized but coordinated operations.

C.2 TRAINING HYPERPARAMETERS

We report the training configurations for the 2B, 4B, and 8B MEMENTOCORE variants. All four adapters are trained with SFT, while the Step Processor and WM Compressor receive an additional DPO refinement stage. Across model scales, we keep the operator design, LoRA rank, cutoff lengths, learning rates, and global batch sizes fixed where possible, and adjust per-device batch size and gradient accumulation to fit the larger backbones. The training hyperparameters are shown in Table 10 ,Table 11 and Table 12

C.3 RUNTIME CONFIGURATION

Table 8 lists the single configuration shared across all GUI backbones, benchmarks, and 2B/4B/8B MEMENTOCORE variants. These values are not tuned on the evaluation sets. Compression is triggered by whichever condition occurs first. Between retrieval activations, the previously selected episodic context is retained.

Table 8: Runtime configuration shared across backbones, benchmarks, and controller sizes.

Runtime parameter	Value
Write threshold	$\tau = 0.3$; write only if $o_t > \tau$
WM compression trigger	12 uncompressed entries or serialized WM context $>$ 2,048 tokens, whichever occurs first
Maximum memory ROI images	$K_{\text{roi}} = 6$
Coarse episodic candidates	Top 4 before learned selection
Episodic retrieval schedule	$t = 1$ or predicted $\gamma_t = 1$

D MEMENTOGUI-BENCH DETAILS

MEMENTOGUI-BENCH is designed to evaluate memory-dependent long-horizon GUI decision making. Unlike standard GUI benchmarks that mainly evaluate whether an agent can ground the next action from the current screenshot, MEMENTOGUI-BENCH focuses on cases where correct behavior depends on accumulated task state. Examples include remembering earlier user constraints, tracking which subgoals have already been completed, preserving values or options that may no longer be visible, and using previous interaction experience to guide later decisions. This makes the benchmark designed to stress decisions that are under-specified by the current screenshot alone.

The benchmark contains 200 trajectories with 6,953 steps in total, averaging 34.8 steps per trajectory. We split these trajectories into two parts: 80 trajectories are used for evaluation, and the remaining 120 trajectories are used to accumulate episodic memory before evaluation. This split reflects the intended use of MEMENTOGUI: during evaluation, the agent can not only maintain working memory within the current trajectory, but also retrieve reusable experience from previously completed trajectories. All reported MEMENTOGUI-BENCH results are computed on the 80 held-out evaluation trajectories, while the additional 120 trajectories serve as the episodic-memory bank for retrieval-based experiments. The raw PSAI split is made by original trajectory ID before any training examples, evaluation instances, or episodic entries are constructed. The 120 bank trajectories are excluded from controller training and evaluation. Task, app, UI, and workflow overlap is intentional, so the setting tests in-domain reuse across distinct trajectories rather than out-of-distribution generalization.

As shown in Figure 10, MEMENTOGUI-BENCH emphasizes medium and hard cases. The benchmark contains 13 easy, 73 medium, and 114 hard trajectories, so most examples require multi-step progress tracking rather than single-step visual grounding. The benchmark also covers both browser and desktop-computer tasks, with 173 browser trajectories and 27 computer trajectories. Browser tasks often involve web navigation, form filling, search, filtering, and multi-page workflows, where the agent must remember previously selected options or information observed on earlier pages. Desktop-computer tasks introduce a different set of challenges, such as application switching, file handling, local windows, menus, and document or system-level interactions.

Overall, MEMENTOGUI-BENCH complements existing GUI benchmarks by targeting the specific failure modes of long-horizon agents: forgetting prior observations, repeating completed actions, losing track of subgoals, and failing to connect current decisions with earlier task context. Its trajectory length, difficulty distribution, and episodic-memory split make it a direct testbed for studying memory-aware GUI control.

E OUR PROPOSED METRICS DETAILS

This section lists the judge prompts used for the three semantic and memory-aware metrics, which are VLM-based Action Match (VAM), Task Progress Score (TPS), and Memory Consistency Score (MCS). These metrics are designed to complement standard reference-based GUI evaluation, which can be too rigid for long-horizon tasks because multiple action sequences may be valid even when they differ from the recorded reference trajectory. Instead of judging only whether the predicted action exactly matches the annotated action, our metrics ask a VLM judge to assess whether the agent's behavior is semantically correct given the current screenshot, the reference action, and the memory state maintained by the agent.

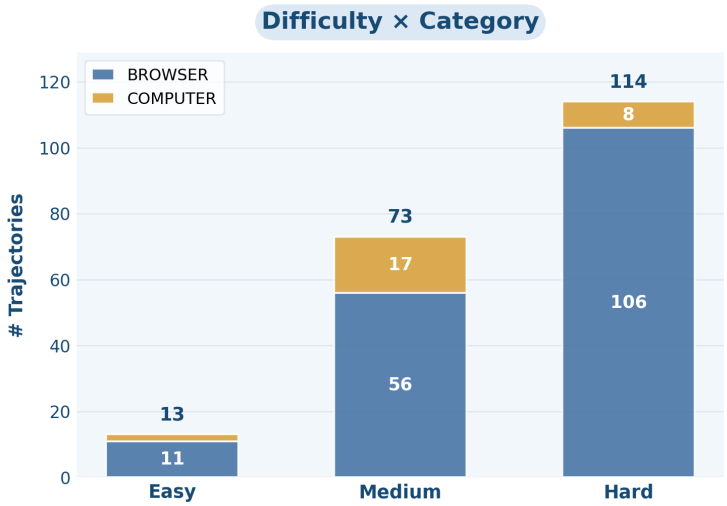


Figure 10: Task difficulty and category composition of MEMENTOGUI-BENCH. The 200 trajectories span three difficulty levels (Easy 13, Medium 73, Hard 114) and two different domains (Browser 173, Computer 27)

VLM-based Action Match is computed at the frame level. For each evaluated step, the judge receives the full screenshot, a close-up ROI around the reference element, the reference action, the predicted action, and the agent memory state before taking the action. The judge first identifies which subgoal or pending task state the reference action is intended to execute, and then determines whether the predicted action is a reasonable instantiation of the same subgoal, which allows VAM to give credit to semantically equivalent actions even when the exact coordinates or surface form differ from the reference. At the same time, the score is memory-aware: predictions receive higher scores when the relevant subgoal, target element, or typed value is explicitly tracked in the memory state, and lower scores when the action is visually plausible but unsupported by the stored state.

Task Progress Score is computed at the trajectory level. It evaluates whether the agent makes coherent progress over time. Rather than judging a single step in isolation, the judge is given a small set of evenly sampled trajectory snapshots, each containing the agent's memory state and the predicted action at that point. The judge then assesses whether the memory state evolves across the trajectory and whether the predicted actions are consistent with those state changes.

Memory Consistency Score is computed at the trajectory level. It focuses on the quality and evolution of the memory state itself. The judge is given sampled memory snapshots from the trajectory and evaluates whether the state meaningfully changes as the task progresses. The judge rewards memory states that grow and shift in response to the interaction history, and penalizes memories that remain nearly identical across steps, repeat generic task descriptions, or fail to preserve important intermediate evidence.

E.1 METRICS PROMPT

Each prompt asks the VLM judge to return a compact JSON object with an integer score and a short reason. The prompts explicitly condition the judgment on memory state so that the scores reflect not only visual action matching but also whether the agent uses accumulated task context coherently.

Judge prompt: VAM — VLM-based Action Match (per frame)

```
You grade a GUI agent's predicted action as an instantiation of a
sub-goal tracked in its memory state.
Inputs:
  - Full screenshot.
  - Close-up ROI of reference element.
```

```
  - Reference action and predicted action.
  - Agent memory state entering this step.
Procedure:
1.  Identify which SUB-GOAL (from the state) the REFERENCE action is
instantiating.  If the state does not name a sub-goal that matches the
reference, note this.
2.  Judge whether the PREDICTED action is also a reasonable
instantiation of the same sub-goal, using the ROI image as visual
evidence.
3.  Bonus:  if the state explicitly names the element or value the
reference uses, and the predicted action targets that same thing,
score higher.
Scoring (0-10):
  10:  reference sub-goal is explicit in state, predicted action
instantiates it correctly (click lands on the named element).
  9:  reference sub-goal is implied in state, predicted instantiates
it.
  8:  predicted action instantiates the right sub-goal but visually
slightly off.
  7:  state contains pending sub-goals but does not name this specific
one; predicted is visually correct.
  6:  state is compatible but lacks sub-goal structure; visual match
only.
  5:  state empty or task-level only; judge purely on visual match
(cap).
  3:  visual match off AND state does not support predicted action.
  0:  contradictory.
A state that cannot be decomposed into sub-goals (purely a task
description) caps the score at 6.  A state with tracked sub-goals that
the action instantiates can reach 9-10.
For TYPE: value-match + sub-goal anchor → 10.
For SCROLL/KEY: direction match + sub-goal anchor → 10.
Return ONLY: {"score":  int 0-10, "reason":  "name the sub-goal from
state (if any) and the instantiation match"}
```

Judge prompt: TPS — Task Progress Score (per trajectory)

```
You grade an entire GUI agent trajectory by how well the sequence
of (memory_state, predicted_action) pairs demonstrates coherent task
progress.
You are given 6 evenly-spaced snapshots of (state, action) across one
trajectory.
A strong working memory should:
  - Have a state that evolves meaningfully between snapshots (new
sub-goals completed, new values recorded).
  - Have actions that visibly tick off items from the state at each
step.
A weak memory (static state, or state that never changes between
snapshots) cannot drive action selection, so even correct-looking
actions appear disconnected.
Scoring (0-10):
  10:  state evolves at every snapshot and actions clearly correspond
to state deltas; crisp coherent trajectory.
  8:  state evolves in most snapshots with matching actions.
  6:  state evolves occasionally; actions partially tied to state.
  4:  state barely changes; actions look state-independent.
  2:  state is flat; actions and state are disconnected.
  0:  state and actions actively contradict each other.
Return ONLY: {"score":  int 0-10, "reason":  "one sentence on
state-action coherence"}
```

Judge prompt: MCS — Memory Consistency Score (per trajectory)

```
You grade whether a GUI agent's accumulated task state evolves
meaningfully across a trajectory.
You are given a list of (step_index, agent_state_snapshot) pairs taken
at regular intervals of one trajectory.
An agent that truly maintains a working memory should have a state
that grows and shifts as the task progresses -- new sub-goals
completed, new values observed, new decisions taken -- not a fixed
summary that repeats.
Scoring (0-10):
  10:  state changes meaningfully between almost every sampled step;
new content appears.
  8:  state changes across most steps.
  6:  state changes occasionally; some sections stay the same.
  4:  state mostly static with small additions.
  2:  state is essentially constant across the trajectory.
  0:  state is identical at every sample.
Return ONLY: {"score":  int 0-10, "reason":  "one sentence"}
```

E.2 User Study

We conduct a ranking-based user study to validate whether the LLM-judged VAM, TPS, and MCS scores align with human preferences. We recruit 10 annotators, and each annotator evaluates 50 comparison groups. Each group contains the same task context and several agent outputs produced by different history or memory settings. Annotators are asked to rank the outputs without seeing the automatic metric scores.

We compare the human rankings with the rankings induced by the LLM judge using Spearman's rank correlation ρ, Kendall's rank correlation τ, and Top-1 agreement. Spearman's ρ measures whether the overall ordering produced by the LLM judge matches the human ordering, while Kendall's τ measures pairwise ranking consistency between the two rankings. Top-1 agreement measures whether the LLM judge selects the same best output as the human annotator. All three statistics are computed directly from the ranking data, and confidence intervals are estimated by bootstrap resampling over comparison groups.

Table 9 shows that the LLM-judged metrics are well aligned with human rankings. VAM has slightly lower agreement than TPS and MCS because it is a frame-level metric and can be affected by local visual ambiguity or small coordinate differences. TPS and MCS achieve stronger correlations, indicating that humans and the LLM judge are more consistent when evaluating trajectory-level progress and memory-state evolution.

Metric	Spearman ρ	Kendall τ	Top-1 Agreement
VAM	0.68	0.52	67.6%
TPS	0.76	0.61	77.4%
MCS	0.79	0.65	79.2%

Table 9: Agreement between human rankings and LLM-judge-induced rankings in the user study. Each comparison group contains four outputs: UI-Venus-1.5-8B with Working Memory, UI-Venus-1.5-8B with Working + Episodic Memory, GUI-Owl-1.5-8B with Working Memory, and GUI-Owl-1.5-8B with Working + Episodic Memory. The statistics are computed over 500 annotated ranking instances.

E.3 Action-only Evaluation

To separate action quality from memory-state quality, we report VAM-A and TPS-A under the same evaluation setup, excluding memory states from the judge inputs. VAM-A assesses semantic action validity for the immediate intent of the reference action; TPS-A assesses progress from predicted actions over the recorded trajectory. Neither variant rewards the content or structure of generated

Adapter	Stage	Rank	Cutoff	LR	Epochs	Per-dev bs	GAS	Global bs	Steps
Step Processor	SFT	32	2048	1e-4	8	16	1	1024	4,892
WM Compressor	SFT	32	4096	1e-4	6	8	2	1024	1,146
Episodic Writer	SFT	32	8192	1e-4	6	8	1	512	1,954
Episodic Selector	SFT	32	8192	1e-4	6	2	2	256	618
Step Processor	DPO	32	2048	5e-6	2	2	1	128	297
WM Compressor	DPO	32	4096	5e-6	2	2	1	128	70

Table 10: Training hyperparameters for the four MEMENTOCORE LoRA adapters on Qwen3-VL-2B-Instruct (Bai et al., 2025). All training jobs use 64 × NVIDIA H200 GPUs, cosine learning-rate schedule, and warmup ratio 0.1. The global batch size is held fixed at the values from our released configs; with 64 GPUs we set *gradient_accumulation_steps* (GAS) so that *per-device bs* × *GAS* × 64 equals the listed global batch. DPO adds a short 2-epoch refinement with a 20× smaller LR (5e-6) and a tighter global batch (128) because preference pairs are sparser than SFT instances. Cutoff length grows with pipeline depth: Step Processor sees a single screenshot (2k tokens), WM Compressor fuses the tail-step buffer (4k), and Episodic Writer/ Episodic Selector take the full WM state plus multiple ROI crops (8k).

Adapter	Stage	Rank	Cutoff	LR	Epochs	Per-dev bs	GAS	Global bs	Steps
Step Processor	SFT	32	2048	1e-4	8	8	2	1024	4,892
WM Compressor	SFT	32	4096	1e-4	6	4	4	1024	1,146
Episodic Writer	SFT	32	8192	1e-4	6	4	2	512	1,954
Episodic Selector	SFT	32	8192	1e-4	6	1	4	256	618
Step Processor	DPO	32	2048	5e-6	2	1	2	128	297
WM Compressor	DPO	32	4096	5e-6	2	1	2	128	70

Table 11: Training hyperparameters for scaling the four MEMENTOCORE LoRA adapters on Qwen3-VL-4B-Instruct (Bai et al., 2025). All training jobs use 64 × NVIDIA H200 GPUs, cosine learning-rate schedule, and warmup ratio 0.1. We inherit the rank, cutoff length, learning rate, number of epochs, number of examples, and global batch size from the 2B configuration in Table 10, while adjusting the per-device batch size and *gradient_accumulation_steps* (GAS) to fit the larger backbone.

memory. Table 13 shows that working memory and working-plus-episodic memory remain stronger than the non-MEMENTOGUI baselines on both backbones. On UI-Venus-1.5-8B, WM+EM achieves 1.72/2.24 VAM-A/TPS-A versus 1.45/1.88 for text-summary memory. The human agreement study in Section E.2 validates the original memory-aware metrics; it does not establish correlation with environment-verified task completion or separately validate the action-only variants.

F MEMENTOGUI WORKFLOW VISUALIZATION

To give a concrete sense of how the four adapters interact at inference time, Figure 1 visualizes one pass through the pipeline: a frame is scored and ROI-localized by the Step Processor, tail-step crops are collapsed by the WM Compressor, a completed trajectory is committed by the Episodic Writer, and retrieved candidates are re-ranked by the Episodic Selector before the merged context is handed to the action backbone. The full per-stage prompts, multi-image inputs, and raw JSON outputs, for every stage and the backbone, are provided in the supplementary.

G ADDITIONAL MEMORY-CONTROLLER COMPARISONS

Prompt-only controllers. We keep the memory pipeline, operator prompts, and frozen action backbone fixed and replace the trained operators with prompt-only inference. Table 14 compares

Adapter	Stage	Rank	Cutoff	LR	Epochs	Per-dev bs	GAS	Global bs	Steps
Step Processor	SFT	32	2048	1e-4	8	4	4	1024	4,892
WM Compressor	SFT	32	4096	1e-4	6	2	8	1024	1,146
Episodic Writer	SFT	32	8192	1e-4	6	2	4	512	1,954
Episodic Selector	SFT	32	8192	1e-4	6	1	4	256	618
Step Processor	DPO	32	2048	5e-6	2	1	2	128	297
WM Compressor	DPO	32	4096	5e-6	2	1	2	128	70

Table 12: Training hyperparameters for scaling the four MEMENTOCORE LoRA adapters on Qwen3-VL-8B-Instruct (Bai et al., 2025). All training jobs use 64 $\times$ NVIDIA H200 GPUs, cosine learning-rate schedule, and warmup ratio 0.1. We inherit the rank, cutoff length, learning rate, number of epochs, number of examples, and global batch size from the 2B configuration in Table 10, while adjusting the per-device batch size and *gradient_accumulation_steps* (GAS) to fit the larger backbone.

Table 13: Action-only evaluation on MEMENTOGUI-BENCH. Memory states are excluded from judge inputs. WM and WM+EM denote working memory and working-plus-episodic memory.

Backbone	Method	VAM-A ↑	TPS-A ↑
UI-Venus-1.5-8B	No History	1.27	1.51
	Pred. Hist. All	1.40	1.69
	Text Summary Memory	1.45	1.88
	Working Memory	1.62	2.10
	Working + Episodic Memory	1.72	2.24
GUI-Owl-1.5-8B	No History	0.79	1.30
	Pred. Hist. All	0.85	1.44
	Text Summary Memory	0.84	1.61
	Working Memory	0.93	1.99
	Working + Episodic Memory	1.06	2.13

untrained Qwen3-VL-2B/4B/8B controllers with the corresponding trained MEMENTOCORE variants. Training improves the reported metrics across the tested settings. For the primary 2B WM+EM configuration, GUI-Odyssey AMS/Traj. SR improves from 58.31/1.81 to 68.32/3.57 on UI-Venus-1.5-8B and from 39.09/0.72 to 49.45/1.71 on GUI-Owl-1.5-8B. The gap is generally smaller for larger controllers, for which the untrained memory harness itself is more competitive.

Strong general VLM controller. Table 15 uses Gemini-3.1-Pro as a prompt-only memory controller with the same frozen UI-Venus-1.5-8B action backbone. Its WM and WM+EM AMS/Traj. SR are 67.93/4.50 and 68.81/5.48, versus 67.69/2.69 and 68.32/3.57 for trained 2B MEMENTOCORE. Thus, task-specific training lets the compact controller approach Gemini's action-matching performance, but Gemini has higher trajectory success. This comparison supports the effectiveness of the memory harness with a strong general VLM; it does not show that the trained 2B controller outperforms it.

Shared versus role-specific adapters. Table 16 compares a single shared rank-64 LoRA with four role-specific LoRAs for 2B and 4B controllers, with the controller backbone, memory pipeline, training data, and action model fixed. Role specialization improves trajectory success in the primary 2B configuration, but does not dominate every metric. The shared adapter becomes more competitive at 4B; for example, GUI-Owl-1.5-8B with WM obtains 1.71 Traj. SR with the shared adapter versus 1.60 with role-specific adapters. A shared adapter is therefore a simpler design option for larger controllers.

The four role-specific adapters share one frozen controller backbone. Training examples are partitioned by memory-control operation, and each adapter is trained only on its corresponding supervision; four adapters therefore do not imply four independent full-data training runs. Only the adapter for

Table 14: Prompt-only (harness) versus trained (4 LoRA) memory controllers. WM denotes working memory; WM+EM additionally uses episodic memory. The memory pipeline and frozen action backbone are fixed.

Backbone	#Param	Method	GUI-Odyssey		MementoGUI-Bench		
			AMS	Traj. SR	VAM	TPS	MCS
UI-Venus-1.5-8B	2B	WM (harness)	60.28	2.12	1.31	3.18	6.33
		WM (4 LoRA)	67.69	2.69	1.41	4.63	7.00
		WM+EM (harness)	58.31	1.81	1.36	2.30	6.30
		WM+EM (4 LoRA)	**68.32**	**3.57**	**1.67**	**5.16**	**7.14**
	4B	WM (harness)	63.52	2.07	1.21	3.29	6.24
		WM (4 LoRA)	67.82	2.74	**1.53**	4.84	7.36
		WM+EM (harness)	64.08	1.97	1.22	3.28	6.13
		WM+EM (4 LoRA)	**68.84**	**3.20**	1.44	**4.90**	**7.71**
	8B	WM (harness)	65.48	2.28	1.20	3.41	6.41
		WM (4 LoRA)	67.38	2.85	1.60	4.83	7.39
		WM+EM (harness)	65.57	2.38	1.25	3.29	6.30
		WM+EM (4 LoRA)	**69.83**	**3.36**	**1.63**	**4.96**	**7.96**
GUI-Owl-1.5-8B	2B	WM (harness)	40.97	0.62	1.51	3.28	5.43
		WM (4 LoRA)	48.25	1.40	1.79	3.38	7.39
		WM+EM (harness)	39.09	0.72	1.66	4.05	6.00
		WM+EM (4 LoRA)	**49.45**	**1.71**	**1.82**	**4.18**	**7.73**
	4B	WM (harness)	46.49	1.14	1.28	3.29	6.44
		WM (4 LoRA)	48.49	1.60	1.64	**3.90**	**7.58**
		WM+EM (harness)	46.71	1.24	1.31	3.23	6.41
		WM+EM (4 LoRA)	**49.05**	**1.71**	**1.87**	3.68	7.53
	8B	WM (harness)	45.84	1.14	1.27	3.43	6.04
		WM (4 LoRA)	49.47	**1.81**	1.65	4.14	7.50
		WM+EM (harness)	46.36	1.45	1.22	3.31	6.18
		WM+EM (4 LoRA)	**50.70**	**1.81**	**2.18**	**4.58**	**7.75**

Table 15: Strong prompt-only memory controller on GUI-Odyssey. Both rows use the same frozen UI-Venus-1.5-8B action backbone; Gemini-3.1-Pro is the memory controller, not the action backbone.

Memory Controller	**Working Memory**		**Working + Episodic Memory**	
	AMS ↑	**Traj. SR ↑**	**AMS ↑**	**Traj. SR ↑**
MementoCore-2B (4LoRA)	67.69	2.69	68.32	3.57
Gemini-3.1-Pro (harness)	67.93	4.50	68.81	5.48

the current operation is active at inference. This avoids replicating the controller backbone, while retaining adapter storage and routing overhead.

Table 16: Shared versus role-specific LoRA controllers at 2B and 4B. The controller backbone, memory pipeline, training data, and action model are fixed. Results include metrics where the shared adapter is stronger.

Backbone	#Param	Memory	Adapter	GUI-Odyssey		MementoGUI-Bench		
				AMS ↑	Traj. SR ↑	VAM ↑	TPS ↑	MCS ↑
UI-Venus-1.5-8B	2B	WM	1 shared LoRA	55.14	1.55	1.37	4.20	**7.03**
			4 role-specific LoRAs	**67.69**	**2.69**	**1.41**	**4.63**	7.00
		WM+EM	1 shared LoRA	60.83	1.81	1.54	4.90	7.00
			4 role-specific LoRAs	**68.32**	**3.57**	**1.67**	**5.16**	**7.14**
	4B	WM	1 shared LoRA	65.26	2.22	1.41	4.03	6.91
			4 role-specific LoRAs	**67.82**	**2.74**	**1.53**	**4.84**	**7.36**
		WM+EM	1 shared LoRA	65.78	2.43	**1.51**	4.66	7.20
			4 role-specific LoRAs	**68.84**	**3.20**	1.44	**4.90**	**7.71**
GUI-Owl-1.5-8B	2B	WM	1 shared LoRA	40.79	0.52	**1.87**	3.30	6.73
			4 role-specific LoRAs	**48.25**	**1.40**	1.79	**3.38**	**7.39**
		WM+EM	1 shared LoRA	44.73	0.67	1.67	2.98	7.62
			4 role-specific LoRAs	**49.45**	**1.71**	**1.82**	**4.18**	**7.73**
	4B	WM	1 shared LoRA	47.99	**1.71**	**1.87**	3.41	6.90
			4 role-specific LoRAs	**48.49**	1.60	1.64	**3.90**	**7.58**
		WM+EM	1 shared LoRA	46.65	1.60	1.83	3.33	7.23
			4 role-specific LoRAs	**49.05**	**1.71**	**1.87**	**3.68**	**7.53**

H LIMITATIONS

MEMENTOGUI is designed as a plug-in memory layer for frozen GUI backbones, and its effectiveness therefore depends on the underlying action model's ability to interpret the injected textual summaries and ROI images. When the backbone fails at basic visual grounding or action-format following, memory augmentation alone may not fully recover correct behavior. In addition, our memory controller is trained from automatically curated computer-use trajectories, which enables scalability but may inherit errors or biases from the preprocessing and VLM-based annotation pipeline. Our benchmark and experiments are primarily offline, so they do not fully capture interactive recovery, environmental stochasticity, or failures caused by irreversible actions in live GUI settings. VLM-judged process scores and their agreement with human rankings do not establish environment-verified completion; reference-path bias remains for alternative next-subgoal branches. MEMENTOGUI-BENCH is trajectory-disjoint but in-domain, allowing task/app/UI/workflow overlap. Role-specific adapters add storage and routing complexity, and the shared-adapter results become more competitive at larger controller sizes. Finally, episodic memory introduces extra retrieval and controller-inference cost, and its usefulness depends on relevant prior experience; gains are not uniform across metrics. Future work should study online adaptation, privacy-preserving episodic memory, and more robust memory control under distribution shifts across devices, applications, and user workflows.