HARNESS2POLICY: HARNESS INTERNALIZATION FOR LONG-HORIZON VISION-LANGUAGE AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-horizon vision-language tasks require agents to preserve visual evidence, manage memory, and coordinate tool use. External harnesses guide these decisions, but successful-trajectory distillation can omit the control supervision and failure experience needed for autonomous execution. We present **Harness2Policy**, a framework that converts harness-guided exploration into process supervision. We search task-adaptive control configurations to guide tree-based exploration, retaining successful paths with explicit planning, memory, verification, and context-supported failure reflections. Using these trajectories, we build **HarnessMind** agents through supervised learning without harness-specific control prompts, enabling control through their responses and tool calls. Repeating this process with the updated model enables recursive co-improvement of the harness and the model. Across four benchmarks, HarnessMind (4B, 9B, and 35B) outperforms its base models with and without benchmark-specific oracle harnesses. On tasks whose reference runs exceed 20 turns, the 9B model improves accuracy from 7.4% with successful-path distillation to 14.3%. It also raises aggregate accuracy from 24.4% to 25.7% over its oracle-harness baseline with 31% fewer inference tokens. This work paves the way for future research on scalable self-improvement through the co-evolution of agentic harnesses, training data, and foundation models.

1 INTRODUCTION

Vision-language agents increasingly solve tasks that combine visual perception, multi-step reasoning, and tool use (Xi et al., 2025; Surís et al., 2023). In long-horizon settings, visual evidence and intermediate findings must remain accessible to guide later decisions (Du et al., 2026; Wang et al., 2024). Matching a product photograph to a configuration diagram, for example, can require tracking component identities across successive image inspections. Such dependencies become harder to manage over extended interactions: hard AgentVista instances can exceed 25 rounds of tool use (Su et al., 2026). Reliable execution therefore requires decisions about what to remember, when to verify evidence, which tools to invoke, and when to stop (Ye et al., 2025; Hu et al., 2023).

External agentic harnesses provide this control by coordinating model interactions with tools and accumulated state (Zhang et al., 2026; Liu et al., 2026b). To adapt control to different tasks and models, automatic design methods search over programs, workflows, and architectures (Hu et al., 2025; Zhang et al., 2025b;a; Zhuge et al., 2024). Optimizing a harness improves guided execution but does not itself teach the model these control decisions. Recent work seeks to transfer this guidance into model policies through joint harness-model optimization and harness-assisted distillation (Chen et al., 2026c;a; Gao et al., 2026; Zhao et al., 2026).

Successful trajectories, however, can omit supervision needed to initiate control autonomously. An externally triggered memory update or verification step may not appear as a decision the student is trained to make. Discarded alternatives offer additional learning signals (Yuan et al., 2025; Zhang et al., 2025c), but retained failure reflections must be supported by observations available to the student. Keeping harness-specific prompts during training also creates a mismatch if they are absent at deployment (Ye et al., 2026; Lu et al., 2026). These gaps motivate explicit control targets, context-supported failure reasoning, and training inputs consistent with deployment.

We introduce **Harness2Policy**, a framework that converts harness-guided exploration into supervision for long-horizon vision-language control (Figure 1). We search task-adaptive control configu-

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

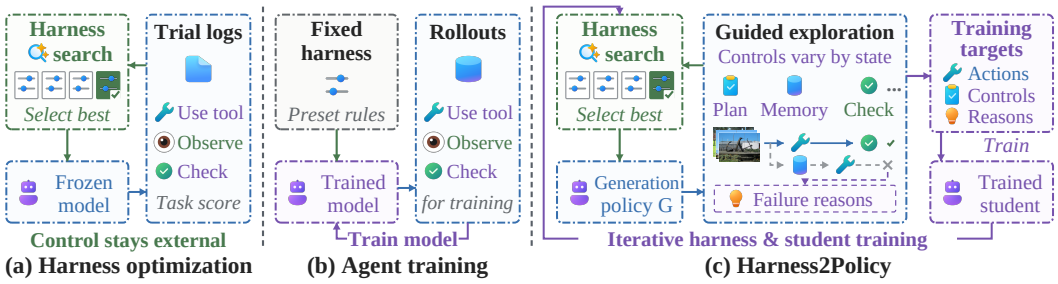


Figure 1: Comparison of agent-development paradigms. Harness2Policy uses harness-guided generation to construct process supervision for a student model; renewed harness search and fresh trajectory generation support further rounds of student training.

rations using the current student, keeping tools and the execution system fixed. Selected harnesses guide a separate generation model through tree-based exploration of training tasks. From these rollouts, we retain successful paths with planning, memory, and verification decisions and add context-supported failure reflections, without supervising failed actions.

We build the **HarnessMind** agents by supervised fine-tuning on this compiled corpus, omitting harness-specific control prompts from their inputs. The agents initiate control from the interaction context through their own responses and tool calls, retaining the deployment interface. Further rounds renew harness search for the updated student and collect fresh trajectories from the generation model, supporting teacher-assisted co-improvement of harnesses and models.

Across DISBench, AgentVista, BrowseComp-VL, and MM-BrowseComp, HarnessMind outperforms its base models with and without benchmark-specific oracle harnesses on all primary metrics at 4B, 9B, and 35B. On tasks whose reference runs exceed 20 turns, the 9B model improves accuracy from 7.4% with matched successful-path distillation to 14.3%. It also raises aggregate accuracy from 24.4% to 25.7% over its oracle-harness baseline with 31% fewer inference tokens. Further training improves accuracy as gains from both fixed and newly searched harnesses decline, providing behavioral evidence consistent with control internalization.

Our contributions are threefold:

- We introduce **Harness2Policy**, a harness-to-policy paradigm for long-horizon vision-language agents that transforms optimized external control into model-native control capabilities and enables recursive co-improvement of harnesses and models.
- We develop **task-adaptive harness search** that uses task scores and execution traces to iteratively refine and combine control components for each task family within a fixed execution system. The optimized harnesses guide subsequent tree-based exploration.
- We compile successful tree-search trajectories and context-supported failure reflections into **end-to-end supervision** for planning, memory, verification, and tool use, training HarnessMind to act without harness-specific control prompts.

2 Related Work

Vision-Language Agents. Vision-language agents combine perception with tool-mediated interaction to acquire and manipulate evidence (Gupta & Kembhavi, 2023; Surís et al., 2023). MMSearch-R1 learns multimodal search policies (Wu et al., 2026), while Visual-ARFT targets search and programmatic visual reasoning (Liu et al., 2025). WebWatcher and Vision-DeepResearch develop tool-interleaved training pipelines for visual information seeking (Huang et al., 2026; Geng et al., 2025). As interactions extend, agents must retain access to earlier visual evidence while incorporating new observations. LMM-Searcher addresses this problem through persistent image references and on-demand visual access (Du et al., 2026); WeAgent-MMSearch maintains native text and image observations and execution recovery within a shared harness for data collection, training, and inference (Liu et al., 2026b). These systems establish the importance of visual evidence management

alongside tool-use learning. We investigate how searched control configurations can provide supervision for the decisions that sustain such interactions, including context folding, memory management, evidence verification, and context-dependent tool selection.

Agentic Harness Engineering. Long-horizon execution requires memory management, failure recovery, and termination control (Wang et al., 2024; Hu et al., 2023). AgentFold uses supervised trajectories to learn proactive, multi-scale context folding, while MEM1 jointly learns memory consolidation and reasoning under bounded context (Ye et al., 2025; Zhou et al., 2025). Automatic design methods such as ADAS, AFlow, and MaAS optimize agent programs, workflows, or architecture distributions (Hu et al., 2025; Zhang et al., 2025b;a; Zhuge et al., 2024); Self-Harness optimizes the surrounding harness with model weights fixed (Zhang et al., 2026). Co-Harness alternates harness refinement and trajectory-based fine-tuning, whereas HarnessForge evolves matched harness-policy pairs (Chen et al., 2026c;a). Experience Funnel selectively consolidates validated textual-state improvements into the policy (Gao et al., 2026), and on-policy harness self-distillation trains a model to reproduce its own harness-assisted behavior without that harness (Zhao et al., 2026). We study supervision from searched control policies for autonomous long-horizon decisions by a single vision-language model, retaining a fixed execution system while removing task-specific control prompts from the student inputs used for both training and deployment.

Training Data Synthesis for Agents. Long-horizon training data combines demanding tasks with supervision for multi-step decisions. LMM-Searcher extends multimodal entity chains to construct demanding search queries, while MEM1 composes multiple objectives into memory-intensive interactions (Du et al., 2026; Zhou et al., 2025). Agent Distillation and Chain-of-Agents generate expert or multi-agent trajectories to train a single agent, including reasoning, tool use, and reflective behavior (Li et al., 2025b; Kang et al., 2025). Search and feedback provide additional supervision (Zhang et al., 2024): Agent-R constructs revision trajectories from erroneous prefixes and successful continuations (Yuan et al., 2025), and Early Experience compares the observed consequences of alternative actions to generate reflections explaining expert choices (Zhang et al., 2025c). OmniAtlas uses hindsight-guided tree exploration for supervised training and teacher-corrected prefixes for subsequent preference learning (Li et al., 2026). Our data engine compiles control events and supported reasons for rejecting failed alternatives from a generation policy's harness-guided search trees into student supervision, using both constructed and curated difficult tasks with reference answers that support task-specific final-answer verification.

3 METHOD

Harness2Policy separates harness search, trajectory generation, and student learning. A design agent searches control configurations, and a generation policy G explores training tasks under the selected harnesses. The resulting control decisions and context-supported failure reflections are compiled into supervision for a student model S_k. Fine-tuning with harness-specific prompts removed yields S_{k+1}, while the tools and execution system remain shared. Subsequent rounds renew harness search and collect fresh trajectories from G for continued student training. Figures 2 and 3 illustrate harness search and the subsequent compilation and training stages.

3.1 CONTROL TARGETS AND SUPERVISION SOURCES

We write a harness as $h = \pi_h \oplus E$. The execution system E supplies tools, state storage, observations, and runtime safeguards. Given the current context, evidence, subgoals, and budget, the control policy $\pi_h : s \mapsto a$ governs planning, verification, and tool use. Memory management and other externally executed operations are exposed as standard tools: the model chooses when and how to call them, and E executes the requests. During model-native inference, the model initiates these operations rather than relying on hand-written triggers, while planning, verification, and reflection are expressed in its responses. Control internalization therefore targets these decisions and calls, with the execution mechanisms retained in E. We denote a model equipped with a harness by $M \circ h$; the "bare model" and $\mathrm{Perf}(M)$ refer to execution with only the same E.

The additional training pool contains eight task families drawn from DeepImageSearch and RedSearcher. Each family contains multiple internal subsets for targeted harness evolution; the deployed

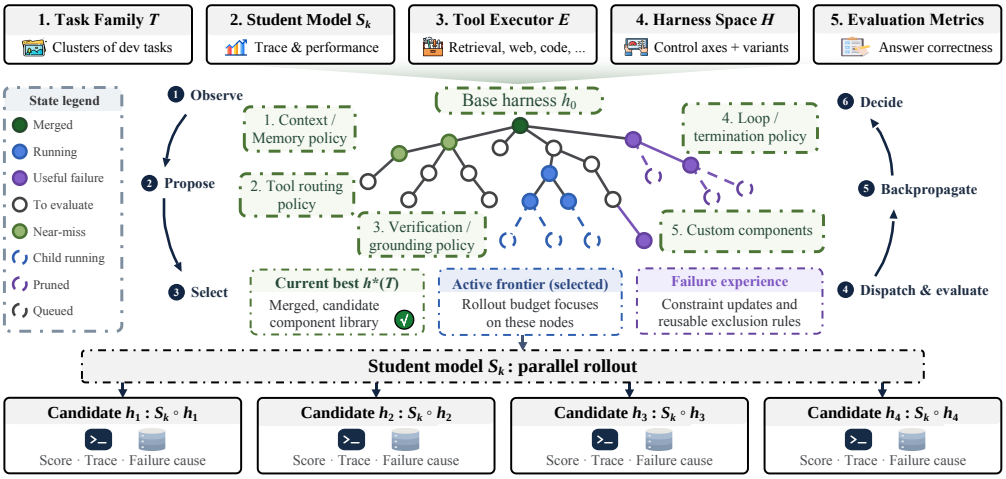


Figure 2: Harness search with a fixed model and set of tools. Each tree node represents a complete control configuration, and execution results guide further changes.

student receives no family labels. The design agent uses execution traces and scores to refine control modules, while the generation policy G supplies the actions, reasoning, and failure explanations used for training. Decoupling trajectory generation from student learning allows a shared corpus to train models at different scales. Appendix B.2 describes the task families and supervision sources.

3.2 Stage A: Composable Control Search

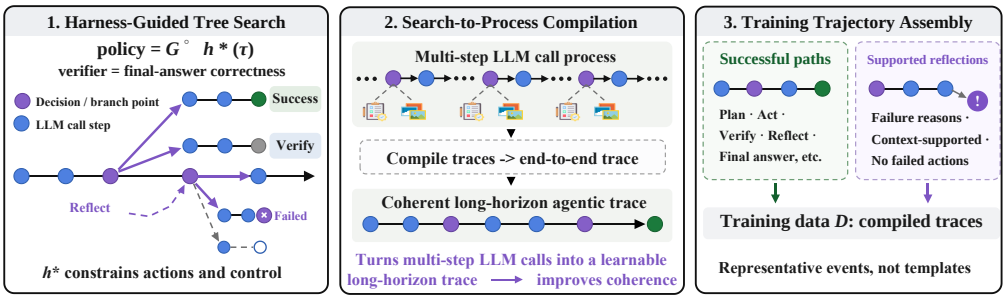
Control components and evaluation. The component inventory covers model-level decisions, execution mechanisms, and optional external controls across planning, memory, tool use, and verification. Starting from a baseline harness, the design agent can retain, remove, modify, or combine control components. At each harness-evolution round, we sample a development subset from the full training task pool to reduce evaluation cost. Candidate harnesses are evaluated with the current student S_k, keeping its checkpoint, the subset, tools, and execution limits fixed within a search run. Search uses no test tasks. For internal subset j of training-task family τ, the selected harness at round k, denoted by $h_k^\star(\tau, j)$, is the best-performing valid configuration found in the corresponding search. Selected harnesses guide subsequent trajectory generation for their subsets.

Feedback-guided refinement. The design agent uses task scores, execution traces, and failure diagnoses to propose revisions, prune unproductive configurations, and retain complementary components. The configuration tree in Figure 2 records candidate harnesses, their revisions, and their evaluation results. Independent candidates can be evaluated in parallel. This tree represents harness development rather than task execution; training trajectories are collected from separate task-action trees in the next stage. Implementation details appear in Appendix D.1.

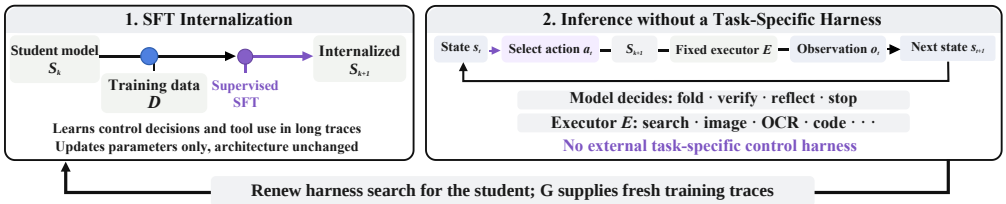
3.3 Stage B: Tree Search and Process Compilation

Controlled tree search. For every task in the full training pool, G performs two independent tree searches under the selected harness. Each search explores branching action continuations, while E returns tool observations, including images. Exploration proceeds within the configured action and search budgets, producing successful paths and failed alternatives. A task-specific verifier identifies successful leaves by comparing generated answers with the reference answer. These judgments provide selection feedback, not supervised response text.

Search-to-process supervision. The compiler selects the shortest successful path within each independent search for training-data construction; searches without a successful path contribute no supervised solution. Compilation retains **planning** responses that specify subgoals, **memory** tool calls that save candidate subsets or fold context, **verification** responses that check visual and



(a) Harness-Guided Exploration and Process Compilation



(b) Internalizing and Improving Model-Native Long-Horizon Capabilities

Figure 3: Learning control decisions from harness-guided exploration. The generation policy supplies successful paths and supported failure reflections; students learn from these trajectories with harness-specific control prompts removed from the training inputs.

cross-modal evidence, and supported **reflections** on failed alternatives, alongside ordinary actions and the final answer. Training samples preserve G's reasoning and tool calls, including memory-call arguments; tool observations remain inputs rather than prediction targets.

Failed-branch observations are not included in the compiled samples. A retained reflection must be supported by the effective context available at its insertion point. Reflections that depend on discarded observations are omitted, and failed actions are not supervised.

The compiler selects and assembles generation-policy outputs and adds fixed structural markers. The design agent's descriptions of harness changes are not substituted for the generated reasoning. Compiled samples must satisfy the formatting and context-length constraints detailed in Appendix D.2.

Trajectory assembly and task coverage. The training pool combines additional synthesized tasks with difficult tasks selected from an existing task source. All tasks undergo the same two-search exploration, verification, and compilation procedure; task definitions themselves are not SFT targets. The selected harness guides experience collection, and the compiler preserves useful control decisions and supported reflections in the retained trajectories. Source-specific task construction, filtering criteria, and compiled-corpus statistics are detailed in Appendix D.2.

3.4 Stage C: Supervised Control Internalization

We use supervised fine-tuning to train S_k on the compiled dataset $\mathcal{D}_k$ produced from G's searches. Let H_t contain the responses and tool observations visible before target token y_t, and let c_E denote the deployment tool definitions and execution interface. The training objective is

$$\mathcal{L}_k(\theta; c_E) = -\mathbb{E}_{(x,y)\sim\mathcal{D}_k}\Big[\sum_t \log p_\theta\big(y_t \mid x, c_E, H_t\big)\Big]. \tag{1}$$

Only generation-policy responses, including tool calls, are prediction targets; retained tool outputs and images provide context. Training inputs retain the deployment interface but omit harness-specific control prompts, aligning the conditioning context with model-native inference while preserving control decisions as explicit prediction targets. Context distillation and skill-internalization methods provide related approaches to reducing dependence on external guidance (Ye et al., 2026; Lu et al., 2026). Fine-tuning starts from S_k and updates its language model to obtain S_{k+1}; the tools and execution system remain available at inference. Training settings, response-token supervision, and the retained deployment inputs are detailed in Appendix D.3.

Table 1: Overall performance on complex vision-language agent tasks, reporting Pass@1 (%). Best and second-best results are bold and underlined within the proprietary and open-source groups; gray and previously reported results are excluded from ranking.

Method	DISBench				AgentVista			BrowseComp-VL			MM-BrowseComp		
	Intra	Inter	EM	F1	Sgl.	Mul.	All	L1	L2	All	OA	SA	CS
Proprietary Agentic VLMs													
GPT-5.6-Luna	51.2	39.2	24.6	44.8	21.9	31.0	24.4	53.3	47.5	50.4	26.3	13.8	35.2
GPT-5.6-Terra	57.8	44.3	31.1	50.6	24.5	37.9	28.2	57.3	51.5	54.4	32.1	19.6	43.6
GPT-5.6-Sol	64.5	50.8	38.5	57.2	**31.1**	**41.4**	**34.0**	**63.3**	57.5	60.4	41.1	25.9	51.8
Gemini-3.1-Pro	59.3	46.4	32.0	52.4	28.5	37.9	31.1	55.3	54.5	54.9	23.7	12.5	33.1
Gemini-3.8-Flash	51.6	41.3	26.2	46.1	20.5	31.0	23.4	58.3	53.0	55.6	25.0	12.9	34.4
Claude-Sonnet-5	61.1	48.3	35.2	54.3	26.5	34.5	28.7	58.3	55.0	56.6	41.5	27.2	53.2
Claude-Opus-5	**67.2**	**53.9**	**41.8**	**60.1**	30.5	**41.4**	33.5	61.8	**59.5**	**60.7**	**45.1**	**30.4**	**57.6**
Open-Source Agentic VLMs													
DeepEyes-v2-7B	22.0	15.5	4.9	18.5	10.6	13.8	11.5	27.1	22.5	24.8	5.4	1.3	7.4
WebWatcher-7B	21.0	15.0	4.1	17.8	9.3	12.1	10.0	22.6	16.0	19.3	4.9	1.3	6.8
OpenSearch-VL-8B	23.5	17.0	6.6	20.0	13.2	19.0	14.8	40.2	35.0	37.6	8.5	2.2	9.2
SearchEyes-9B	27.0	20.0	6.6	23.3	17.2	24.1	19.1	44.7	40.0	42.4	9.8	2.7	10.0
WebWatcher-32B	29.0	21.5	9.0	25.0	9.9	10.3	10.0	27.6	24.0	25.8	7.6	1.8	8.9
Qwen3-VL-235B-A22B	32.1	22.9	11.5	27.2	11.8	15.8	12.9	40.7	35.1	37.9	12.0	3.3	11.5
Qwen3.5-397B-A17B	35.0	27.0	13.1	30.7	18.5	24.1	20.1	63.3	56.7	60.0	18.0	5.8	16.0
Qwen3.5-4B	23.1	16.9	4.1	19.8	8.6	10.3	9.1	28.6	19.0	23.8	4.5	0.9	6.7
w/ oracle harness	24.6	18.3	4.9	21.2	9.9	13.8	11.0	34.2	26.5	30.3	6.7	1.3	8.4
HarnessMind (4B)	25.4	18.8	5.7	21.9	10.6	15.5	12.0	33.7	28.0	30.8	7.6	1.8	9.0
Qwen3.5-9B	25.5	18.9	5.7	22.0	12.6	17.2	13.9	36.2	27.0	31.6	6.7	1.3	8.1
w/ oracle harness	27.9	20.7	6.6	24.1	13.9	24.1	16.7	42.7	40.5	41.6	10.7	2.2	9.9
HarnessMind (9B)	28.8	21.6	7.4	25.0	15.2	22.4	17.2	44.2	43.0	43.6	11.6	2.7	10.6
Qwen3.5-35B-A3B	30.5	23.2	8.2	26.6	18.5	25.9	20.6	48.7	41.0	44.9	10.3	3.1	9.4
w/ oracle harness	33.4	**26.9**	9.0	29.9	21.2	29.3	23.4	52.3	47.0	49.6	14.7	4.5	13.2
HarnessMind (35B)	**34.0**	26.5	**9.8**	**30.0**	**21.9**	**31.0**	**24.4**	**52.8**	**48.0**	**50.4**	**15.2**	**4.9**	**13.7**

3.5 Evaluating Control Internalization

We assess internalization through *autonomous control without task-specific external policies*. We freeze $h_0^\star$ from round 0 and report its additional gain, $\mathrm{IG}_k = \mathrm{Perf}(S_k \circ h_0^\star) - \mathrm{Perf}(S_k)$, alongside absolute student performance without external control. We also rerun harness search for the updated student to measure the remaining benefit of new external guidance. These rounds constitute teacher-assisted iterative training: G supplies new trajectories while S_k is updated. A decline in external gains provides behavioral evidence but does not by itself rule out explanations such as general capability improvement. Evaluation settings appear in Appendix D.4.

4 Experiments

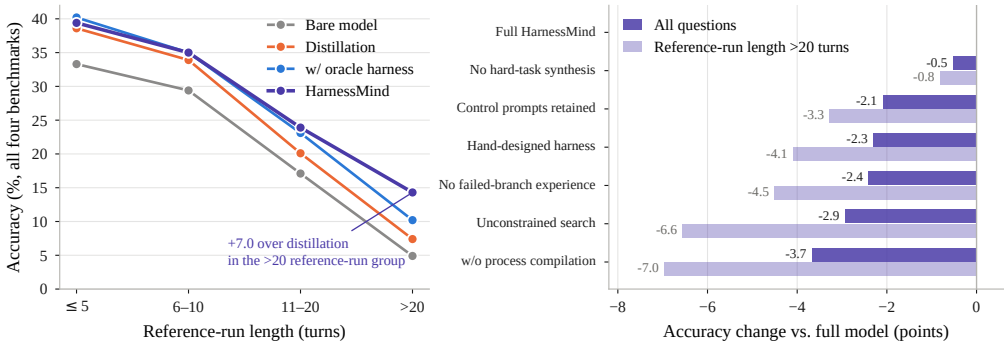
4.1 Experimental Setup

Benchmarks and metrics. We evaluate on DISBench (Deng et al., 2026), AgentVista (Su et al., 2026), BrowseComp-VL (Geng et al., 2025), and MM-BrowseComp (Li et al., 2025a). The primary metrics are exact match (EM) for DISBench and overall accuracy for the other benchmarks. DISBench additionally reports macro F1 overall and on its Intra and Inter splits; MM-BrowseComp reports overall accuracy (OA), strict accuracy (SA), and checklist score (CS). For evaluations conducted in this work, each configuration is run once per test task; aggregate accuracy pools correct answers over 954 tasks, with evaluation sizes of 122, 209, 399, and 224 tasks, respectively. We use MM-BrowseComp v1 and the 399-question version of BrowseComp-VL.

Baselines. We compare with proprietary agentic VLMs, including GPT-5.6-Luna, GPT-5.6-Terra, GPT-5.6-Sol, Gemini-3.1-Pro, Gemini-3.8-Flash, Claude-Sonnet-5, and Claude-Opus-5. Open-source baselines include larger Qwen models, DeepEyesV2 (Hong et al., 2025), WebWatcher (Geng et al., 2025), OpenSearch-VL (Chen et al., 2026b), and SearchEyes (Jiao et al., 2026). Controlled comparisons include the base model without task-specific control, ReAct (Yao et al., 2023), a fixed

Table 2: Performance across control orientations with Qwen3.5-9B. The initial-model row reports scores (%); other rows report gains in percentage points. Bold and underlined values mark the best and second-best gains among the individual control settings.

Method	DISBench				AgentVista			BrowseComp-VL			MM-BrowseComp		
	Intra	Inter	EM	F1	Sgl.	Mul.	All	L1	L2	All	OA	SA	CS
Initial model	25.5	18.9	5.7	22.0	12.6	17.2	13.9	36.2	27.0	31.6	6.7	1.3	8.1
Image Discovery Harness	+2.4	+1.8	+0.8	+2.1	+0.7	0.0	+0.5	+1.5	+1.5	+1.5	+0.4	0.0	+0.2
Vision-focused Harness	-1.5	-1.3	-0.8	-1.4	+1.3	**+6.9**	+2.9	+2.5	+4.0	+3.3	+1.3	+0.4	+0.5
Search-focused Harness	+0.2	+0.2	0.0	+0.2	+1.0	+1.7	+1.2	+5.5	+11.5	+8.5	+3.6	+0.9	+1.6
Unified Harness	+0.4	+0.3	0.0	+0.4	+2.0	0.0	+1.4	+3.5	+9.5	+6.5	+2.7	+0.4	+1.0
HarnessMind (Model-Native)	**+3.3**	**+2.7**	**+1.6**	**+3.0**	**+2.6**	+5.2	**+3.3**	**+8.0**	**+16.0**	**+12.0**	**+4.9**	**+1.3**	**+2.5**



(a) Accuracy by reference-run length

(b) Removing one core design at a time

Figure 4: Effects of training-data construction and individual components. (a) Accuracy by reference-run length. (b) Accuracy drops relative to the full model after component changes, reported over all tasks (dark) and the subset whose reference runs exceed 20 turns (light).

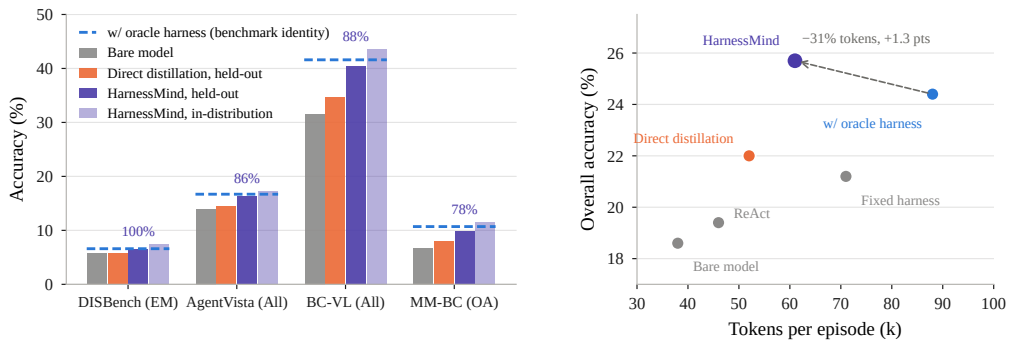
hand-designed harness, searched harnesses, and best-path distillation. The oracle baseline selects a benchmark-specific harness using benchmark identity, whereas HarnessMind requires no such selection at inference. Distillation uses the same generation-policy search trees and matches the number of training target tokens to isolate the contribution of process compilation.

Implementation Details. We automate harness evolution using Claude Code with Claude Opus 4.8 and train students with LLaMA-Factory. Qwen3.5-9B is the primary model for analysis; Qwen3.5-35B-A3B generates the shared training corpus used by the primary model and the 4B and 35B variants. Each harness-search round uses a newly sampled 300-task subset of the full training pool. Trajectory collection covers all training tasks with two independent tree searches per task. The first-round SFT corpus contains 1,162 samples from verified successful paths (746 from DeepImageSearch and 416 from RedSearcher), with contexts within 40,960 tokens. We train for three epochs on 8 H20-141GB GPUs with a learning rate of 10^{-5}, freezing the vision encoder and projector. Controlled comparisons share the tools and execution environment. Appendix A describes the datasets and baselines, and Appendix D details data construction and training.

4.2 Main Results

Table 1 yields three main observations. **HarnessMind improves all four primary benchmark metrics for the 4B, 9B, and 35B backbones.** For the primary 9B model, BrowseComp-VL accuracy rises from 31.6% to 43.6%, while MM-BrowseComp accuracy increases from 6.7% to 11.6%. The gains extend to both smaller models and stronger starting checkpoints: BrowseComp-VL improves from 23.8% to 30.8% at 4B and from 44.9% to 50.4% at 35B. Thus, the shared training corpus benefits all three backbones rather than only the primary model.

HarnessMind surpasses the corresponding oracle-harness baselines on all four primary metrics at every model size. The oracle uses benchmark identity to select a specialized harness, whereas



(a) Held-out benchmark: share of oracle gain recovered

(b) Accuracy vs. inference cost

Figure 5: Performance on unseen benchmarks and inference cost (9B). (a) Accuracy on excluded benchmarks; blue dashed lines show the searched harness baseline. (b) Aggregate accuracy versus mean input and output tokens per task across all model calls.

HarnessMind requires no benchmark-specific harness selection at inference. The 9B model gains 0.5–2.0 percentage points, with BrowseComp-VL improving from 41.6% to 43.6%. These gains are obtained with the same tools and execution environment, showing that harness-guided training can improve on the performance already provided by optimized external control.

The 9B HarnessMind model exceeds OpenSearch-VL on all four primary metrics and SearchEyes on three (Chen et al., 2026b; Jiao et al., 2026). Against SearchEyes, BrowseComp-VL accuracy improves from 42.4% to 43.6% and MM-BrowseComp accuracy from 9.8% to 11.6%, while SearchEyes remains stronger on AgentVista (19.1% versus 17.2%). The 35B model extends this competitiveness, leading the ranked open-source group on all four primary metrics and reaching 50.4% on BrowseComp-VL and 15.2% on MM-BrowseComp. These comparisons show that a shared model-native policy can remain competitive with specialized agents across diverse tasks.

4.3 Control Orientations Across Benchmarks

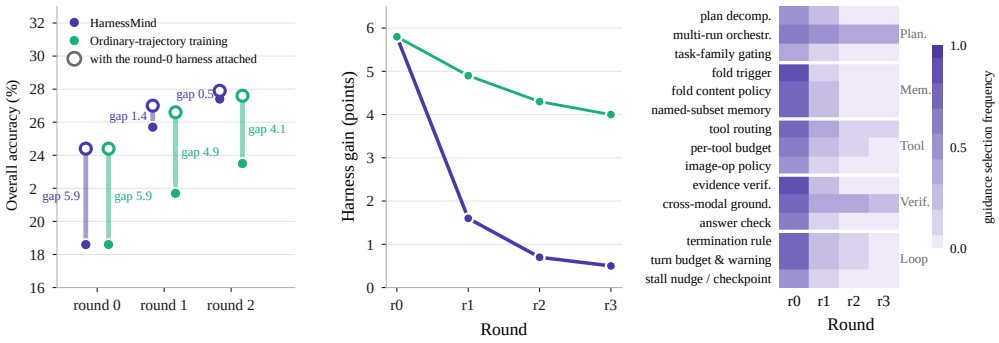
Table 2 groups harnesses by control orientation. Image Discovery emphasizes image retrieval and candidate management; Vision-focused emphasizes visual evidence inspection and cross-modal grounding; Search-focused emphasizes information retrieval, browsing, and evidence integration. Unified combines these emphases in a single control configuration. Performance gains vary across evaluation domains. The two search-focused configurations average gains of 8.5 and 3.6 percentage points on BrowseComp-VL and MM-BrowseComp, compared with 6.5 and 2.7 for Unified. By learning from complementary control experience, **HarnessMind achieves the largest gain on every primary benchmark metric without requiring task-family labels at inference.**

4.4 Long-Horizon Performance

Figure 4 groups questions by the initial model's turn count in a single reference run with its benchmark-specific harness. The groups contain 132, 180, 398, and 244 tasks and are shared across methods and ablations. Successful-path distillation uses the best SFT recipe in Table 7 and is reported as "w/o process compilation" in Table 6, with the same number of training target tokens as the full method. **The value of process supervision becomes more pronounced in longer interactions.** Its advantage over successful-path distillation is small on short runs but expands in the longest group, where accuracy rises from 7.4% to 14.3%, a gain of 7.0 percentage points. The full method also exceeds the searched-harness baseline on these longer runs. Together with the larger ablation effects in Figure 4(b), this pattern suggests that retaining control decisions and supported failure experience provides benefits beyond imitating the final successful solution.

4.5 Core Component Ablations

Figure 4(b) identifies **process compilation as a central contributor to the improvement**. Replacing it with successful-path distillation reduces overall accuracy from 25.7% to 22.0%, while removing



(a) Harness on vs. off: the gap closes (b) Gain after renewed search (c) Auxiliary-guidance selection

Figure 6: Reliance on external guidance over training rounds. (a) Test accuracy with (hollow dots) and without (filled dots) the round-0 harness; lines show its gain. (b) Additional accuracy from renewed harness search. (c) Frequency with which renewed search selects additional guidance.

failed-branch experience also causes a substantial drop. Harness search and removal of control prompts provide complementary benefits: the former improves the experience collected for training, while the latter aligns training with deployment without external guidance. Without hard-task synthesis, accuracy is 25.2%, compared with 25.7% for the full model. The ablations have stronger effects in the long-run group. Table 6 gives the detailed results.

4.6 Inference Efficiency

Figure 5(b) measures inference cost as the total input and output tokens across all model calls for a task, averaged over tasks. Relative to the initial model with its oracle harness, HarnessMind improves accuracy from 24.4% to 25.7% while reducing mean token use from approximately 88k to 61k (31%). **The student therefore achieves a better accuracy-cost tradeoff than external harness control** while using the same tools and execution environment.

4.7 Recursive Improvement and External Control

Figure 6 follows teacher-assisted training with fresh trajectories from G. Each round resamples 300 training tasks for harness search. Both training procedures are assessed on the sampled training-development subset and evaluated separately on the test sets. Figures 6(a) and 6(b) report fixed- and renewed-harness gains, while Figure 6(c) tracks auxiliary guidance selected by renewed search across the eight training-task families and their internal subsets. Appendix D.4 details the protocol. **Student accuracy increases across rounds as the original harness's added benefit falls from 5.9 to 0.5 percentage points by round 2.** After renewed search, the additional gain reaches 0.5 points for HarnessMind versus 4.0 for ordinary-trajectory training by round 3. Selection frequencies vary across components: planning and memory guidance decline, while cross-run and cross-modal guidance continue to be selected. These trends suggest reduced reliance on external guidance.

5 Conclusion

Harness2Policy advances a harness-to-policy paradigm for long-horizon vision-language agents, coupling task-adaptive harness search with process supervision from tree-based exploration. By preserving control decisions and context-supported failure reflections while removing harness-specific prompts, it trains HarnessMind to initiate control through its own responses and tool calls. Across four benchmarks and three model scales, HarnessMind outperforms its base models with and without oracle harnesses on the primary metrics. Controlled analyses show stronger long-horizon performance than successful-path distillation and higher accuracy with fewer tokens than the oracle-harness baseline. Renewed search and teacher-assisted training support further co-improvement as the benefit of external guidance declines. Together, these findings open a path toward scalable self-improvement through the co-evolution of harnesses, training data, and foundation models.

AI USE STATEMENT

Generative AI tools were used to assist with language editing, literature search and analysis, and code implementation. The research ideas, methodology, and experimental design were developed independently by the authors. Model-based harness search, training-data generation, and teacher-assisted comparisons are documented in Section 3 and Appendices B.2 and C.3. All AI-assisted manuscript content, literature-review outputs, and code were reviewed by the authors, who take responsibility for the final manuscript and its scientific content.

ETHICS STATEMENT

We identify no additional ethical concerns specific to this study.

REPRODUCIBILITY STATEMENT

Sections 3 and 4.1 describe the framework and experimental settings. Appendices A, B, and D provide dataset and baseline descriptions, control components, supervision sources, and implementation details for harness search, trajectory compilation, and student training.

REFERENCES

Mingju Chen, Can Lv, Guibin Zhang, Heng Chang, and Shiji Zhou. Harnessforge: Joint harness and policy evolution for adaptive agent systems, 2026a. URL https://arxiv.org/abs/2606.01779.

Shuang Chen, Kaituo Feng, Hangting Chen, Wenxuan Huang, Dasen Dai, Quanxin Shou, Yunlong Lin, Xiangyu Yue, Shenghua Gao, and Tianyu Pang. Opensearch-vl: An open recipe for frontier multimodal search agents, 2026b. URL https://arxiv.org/abs/2605.05185.

Zhengyu Chen, Teng Xiao, Huaisheng Zhu, Yige Yuan, Luan Zhang, and Jingang Wang. Co-harness: Co-evolving harnesses and model weights for llm agents, 2026c. URL https://arxiv.org/abs/2607.22688.

Chenlong Deng, Mengjie Deng, Junjie Wu, Dun Zeng, Teng Wang, Qingsong Xie, Jiadeng Huang, Shengjie Ma, Changwang Zhang, Zhaoxiang Wang, Jun Wang, Yutao Zhu, and Zhicheng Dou. Deepimagesearch: Benchmarking multimodal agents for context-aware image retrieval in visual histories. *CoRR*, abs/2602.10809, 2026. doi: 10.48550/ARXIV.2602.10809. URL https://doi.org/10.48550/arXiv.2602.10809.

Yifan Du, Zikang Liu, Jinbiao Peng, Jie Wu, Junyi Li, Jinyang Li, Wayne Xin Zhao, and Ji-Rong Wen. Towards long-horizon agentic multimodal search, 2026. URL https://arxiv.org/abs/2604.12890.

Wenbo Gao, Zhaomou Song, Zhiyuan Ji, Renxi Liu, Xing Li, Xianzhi Yu, Xiaoguang Li, James Chung wai Cheung, Weizhe Lin, and Yaoyuan Wang. Experience funnel: A state-policy alternating loop for self-evolving agents, 2026. URL https://arxiv.org/abs/2609.08919.

Xinyu Geng, Peng Xia, Zhen Zhang, Xinyu Wang, Qiuchen Wang, Ruixue Ding, Chenxi Wang, Jialong Wu, Yida Zhao, Kuan Li, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. Webwatcher: Breaking new frontier of vision-language deep research agent. *CoRR*, abs/2508.05748, 2025. doi: 10.48550/ARXIV.2508.05748. URL https://doi.org/10.48550/arXiv.2508.05748.

Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pp. 14953–14962. IEEE, 2023. doi: 10.1109/CVPR52729.2023.01436. URL https://doi.org/10.1109/CVPR52729.2023.01436.

Jack Hong, Chenxiao Zhao, ChengLin Zhu, Weiheng Lu, Guohai Xu, and Xing Yu. Deepeyesv2: Toward agentic multimodal model. *CoRR*, abs/2511.05271, 2025. doi: 10.48550/ARXIV.2511.05271. URL https://doi.org/10.48550/arXiv.2511.05271.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=t9U3LW7JVX.

Ziniu Hu, Ahmet Iscen, Chen Sun, Kai-Wei Chang, Yizhou Sun, David Ross, Cordelia Schmid, and Alireza Fathi. AVIS: autonomous visual information seeking with large language model agent. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/029df12a9363313c3e41047844ecad94-Abstract-Conference.html.

Wenxuan Huang, Yu Zeng, Qiuchen Wang, Zhen Fang, Shaosheng Cao, Zheng Chu, Qingyu Yin, Shuang Chen, Zhenfei Yin, Lin Chen, Zehui Chen, Xu Tang, Yao Hu, Shaohui Lin, Philip Torr, Feng Zhao, and Wanli Ouyang. Vision-deepresearch: Incentivizing deepresearch capability in multimodal large language models, 2026. URL https://arxiv.org/abs/2601.22060.

Zhengbo Jiao, Yiming Cheng, Yilei Jiang, Kaituo Feng, Rui Huang, Tianyi Jiang, Juanxi Tian, Jiapeng li, Qunzhong Wang, Tailai Chen, Qianshan Wei, Chuan Xiao, Shanyu Rong, Yangfu Li, Yanhan Zhou, Yunpu Ma, Yifan Zhang, and Xiangyu Yue. Searcheyes: Towards frontier multimodal deep search intelligence via search world simulation, 2026. URL https://arxiv.org/abs/2607.05943.

Minki Kang, Jongwon Jeong, Seanie Lee, Jaewoong Cho, and Sung Ju Hwang. Distilling LLM agent into small models with retrieval and code tools. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/99263f9bf46874e2b8b7f104b5063864-Abstract-Conference.html.

Shilong Li, Xingyuan Bu, Wenjie Wang, Jiaheng Liu, Jun Dong, Haoyang He, Hao Lu, Haozhe Zhang, Chenchen Jing, Zhen Li, Chuanhao Li, Jiayi Tian, Chenchen Zhang, Tianhao Peng, Yancheng He, Jihao Gu, Yuanxing Zhang, Jian Yang, Ge Zhang, Wenhao Huang, Wangchunshu Zhou, Zhaoxiang Zhang, Ruizhe Ding, and Shilei Wen. Mm-browsecomp: A comprehensive benchmark for multimodal browsing agents. *CoRR*, abs/2508.13186, 2025a. doi: 10.48550/ARXIV.2508.13186. URL https://doi.org/10.48550/arXiv.2508.13186.

Weizhen Li, Jianbo Lin, Zhuosong Jiang, Jingyi Cao, Xinpeng Liu, Jiayu Zhang, Zhenqiang Huang, Qianben Chen, Weichen Sun, Qiexiang Wang, Hongxuan Lu, Tianrui Qin, Chenghao Zhu, Yi Yao, Shuying Fan, Xiaowan Li, Tiannan Wang, Pai Liu, King Zhu, He Zhu, Dingfeng Shi, Piaohong Wang, Yeyi Guan, Xiangru Tang, Minghao Liu, Yuchen Eleanor Jiang, Jian Yang, Jiaheng Liu, Ge Zhang, and Wangchunshu Zhou. Chain-of-agents: End-to-end agent foundation models via multi-agent distillation and agentic rl, 2025b. URL https://arxiv.org/abs/2508.13167.

Xiaoxi Li, Wenxiang Jiao, Jiarui Jin, Haoxuan Li, Hao Wang, Shijian Wang, Guanting Dong, Jiajie Jin, Yinuo Wang, Yuan Lu, Ji-Rong Wen, Zhicheng Dou, and Zhouchen Lin. Omnigaia: Towards native omni-modal ai agents, 2026. URL https://arxiv.org/abs/2602.22897.

Jun Liu, Zhenglun Kong, Peiyan Dong, Changdi Yang, Tianqin Li, Yanyue Xie, Yifan Gong, Xuan Shen, Pu Zhao, Hao Tang, Geng Yuan, Wei Niu, Wenbin Zhang, Xue Lin, Dong Huang, and Yanzhi Wang. Structured agent distillation for large language model agents. In Viviana Mascardi, John Thangarajah, Chris Amato, and Louise A. Dennis (eds.), *Proceedings of the 25th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2026, Paphos, Cyprus, May 25-29, 2026*, pp. 3676–3685. ACM, 2026a. doi: 10.65109/OLHJ8062. URL https://dl.acm.org/doi/10.65109/OLHJ8062.

Ziyu Liu, Yuhang Zang, Yushan Zou, Zijian Liang, Xiaoyi Dong, Yuhang Cao, Haodong Duan, Dahua Lin, and Jiaqi Wang. Visual agentic reinforcement fine-tuning, 2025. URL https://arxiv.org/abs/2505.14246.

Zongkai Liu, Hui Zhang, Liqiang Niu, Zhen Cao, Han Li, Juntao Liu, Wenchao Chen, Chengduo Zhao, Chao Yu, and Fandong Meng. Weagent-mmsearch: Native text-vision interaction for multimodal search agents, 2026b. URL https://arxiv.org/abs/2608.28062.

Zhengxi Lu, Zhiyuan Yao, Jinyang Wu, Chengcheng Han, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Skill0: In-context agentic reinforcement learning for skill internalization, 2026. URL https://arxiv.org/abs/2604.02268.

Yinyi Luo, Yiqiao Jin, Weichen Yu, Mengqi Zhang, Srijan Kumar, Xiaoxiao Li, Weijie Xu, Xin Chen, and Jindong Wang. Agentark: Distilling multi-agent intelligence into a single llm agent, 2026. URL https://arxiv.org/abs/2602.03955.

Zhaochen Su, Jincheng Gao, Hangyu Guo, Zhenhua Liu, Lueyang Zhang, Xinyu Geng, Shijue Huang, Peng Xia, Guanyu Jiang, Cheng Wang, Yue Zhang, Yi R. Fung, and Junxian He. Agentvista: Evaluating multimodal agents in ultra-challenging realistic visual scenarios, 2026. URL https://arxiv.org/abs/2602.23166.

Dídac Surís, Sachit Menon, and Carl Vondrick. Vipergpt: Visual inference via python execution for reasoning. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pp. 11854–11864. IEEE, 2023. doi: 10.1109/ICCV51070.2023.01092. URL https://doi.org/10.1109/ICCV51070.2023.01092.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. A survey on large language model based autonomous agents. *Frontiers Comput. Sci.*, 18(6):186345, 2024. doi: 10.1007/S11704-024-40231-1. URL https://doi.org/10.1007/s11704-024-40231-1.

Jinming Wu, Zihao Deng, Wei Li, Yiding Liu, Bo You, Bo Li, Zejun Ma, and Ziwei Liu. Mmsearch-r1: Incentivizing lmms to search. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2026, San Diego, California, United States, July 2-7, 2026*, pp. 2456–2487. Association for Computational Linguistics, 2026. doi: 10.18653/V1/2026.ACL-LONG.114. URL https://doi.org/10.18653/v1/2026.acl-long.114.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, Wenjuan Qin, Yongyan Zheng, Xipeng Qiu, Xuanjing Huang, Qi Zhang, and Tao Gui. The rise and potential of large language model based agents: a survey. *Sci. China Inf. Sci.*, 68 (2), 2025. doi: 10.1007/S11432-024-4222-0. URL https://doi.org/10.1007/s11432-024-4222-0.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Rui Ye, Zhongwang Zhang, Kuan Li, Huifeng Yin, Zhengwei Tao, Yida Zhao, Liangcai Su, Liwen Zhang, Zile Qiao, Xinyu Wang, Pengjun Xie, Fei Huang, Siheng Chen, Jingren Zhou, and Yong Jiang. Agentfold: Long-horizon web agents with proactive context management. *CoRR*, abs/2510.24699, 2025. doi: 10.48550/ARXIV.2510.24699. URL https://doi.org/10.48550/arXiv.2510.24699.

Tianzhu Ye, Li Dong, Xun Wu, Shaohan Huang, and Furu Wei. On-policy context distillation for language models, 2026. URL https://arxiv.org/abs/2602.12275.

Siyu Yuan, Zehui Chen, Zhiheng Xi, Junjie Ye, Zhengyin Du, and Jiecao Chen. Agent-r: Training language model agents to reflect via iterative self-training, 2025. URL https://arxiv.org/abs/2501.11425.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. Rest-mcts*: LLM self-training via process reward guided tree search. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/76ec4dc30e9faaf0e4b6093eaa377218-Abstract-Conference.html.

Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, Lei Bai, and Xiang Wang. Multi-agent architecture search via agentic supernet. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025a. URL https://proceedings.mlr.press/v267/zhang25bi.html.

Hangfan Zhang, Shao Zhang, Kangcong Li, Chen Zhang, Yang Chen, Yiqun Zhang, Lei Bai, and Shuyue Hu. Self-harness: Harnesses that improve themselves, 2026. URL https://arxiv.org/abs/2606.09498.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. Aflow: Automating agentic workflow generation. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025b. URL https://openreview.net/forum?id=z5uVAKwmjf.

Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, Jian Xie, Yuxuan Sun, Boyu Gou, Qi Qi, Zihang Meng, Jianwei Yang, Ning Zhang, Xian Li, Ashish Shah, Dat Huynh, Hengduo Li, Zi Yang, Sara Cao, Lawrence Jang, Shuyan Zhou, Jiacheng Zhu, Huan Sun, Jason Weston, Yu Su, and Yifan Wu. Agent learning via early experience. *CoRR*, abs/2510.08558, 2025c. doi: 10.48550/ARXIV.2510.08558. URL https://doi.org/10.48550/arXiv.2510.08558.

Zhengyang Zhao, Lu Ma, and Wentao Zhang. Training with harnesses: On-policy harness self-distillation for complex reasoning, 2026. URL https://arxiv.org/abs/2605.08741.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Jinhua Zhao, Bryan Kian Hsiang Low, and Paul Pu Liang. MEM1: learning to synergize memory and reasoning for efficient long-horizon agents. *CoRR*, abs/2506.15841, 2025. doi: 10.48550/ARXIV.2506.15841. URL https://doi.org/10.48550/arXiv.2506.15841.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. Gptswarm: Language agents as optimizable graphs. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62743–62767. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/zhuge24a.html.

APPENDIX CONTENTS

A Datasets and Baselines **15**

A.1 Datasets 15

A.2 Baselines 15

B Harness Components and Attribution Protocol **16**

B.1 Harness Component Inventory 16

B.2 Attribution Protocol and Task Families 16

C Additional Experiments **17**

C.1 Detailed Ablation Results 17

C.2 Model and Data Scaling 17

C.3 Distillation and Harness Search 18

D Method Implementation Details **19**

D.1 Stage A: Harness Search with a Design Agent 19

D.2 Stage B: Exploration and Trajectory Compilation 20

D.3 Stage C: Training and Removal of Control Prompts 21

D.4 External-Control Evaluation across Rounds 21

E Limitations **22**

F Broader Impacts and Future Directions **22**

A Datasets and Baselines

A.1 Datasets

- **DISBench** (Deng et al., 2026) evaluates context-aware image retrieval from personal visual histories through 122 queries. Each query targets a set of images whose relevance depends on event context and spatiotemporal relationships, rather than isolated query-image similarity. Intra-Event queries retrieve images within a specific event, whereas Inter-Event queries require evidence across events. We report set-level exact match and macro F1, with F1 also broken down by event type.
- **AgentVista** (Su et al., 2026) comprises 209 manually authored requests grounded in realistic visual inputs, including photographs, screenshots, and technical diagrams. Solving these tasks requires interleaving visual reasoning with web and image search, page navigation, and code execution. We report final-answer accuracy overall and for the single-image and multi-image subsets.
- **BrowseComp-VL** (Geng et al., 2025) extends BrowseComp-style information seeking to questions that combine visual evidence with multi-hop web research. We use its 399-question evaluation set. Level 1 retains explicit entities while requiring multi-step retrieval; Level 2 obscures entities or attributes, increasing the need to compare and integrate evidence across sources. Accuracy is reported for each difficulty level and over the full evaluation set.
- **MM-BrowseComp** (Li et al., 2025a) evaluates browsing tasks whose decisive evidence may appear in online images or videos, even when the initial prompt is text-only. We use the 224-question v1 release, which provides a verified reasoning checklist for each question. Overall accuracy measures final-answer correctness; strict accuracy additionally requires full checklist completion; and checklist score averages per-question checklist completion rates.

A.2 Baselines

Reference models and specialized agents.

- **Proprietary VLMs.** The GPT, Gemini, and Claude variants in Table 1 provide general-purpose baselines for multimodal reasoning and tool use alongside specialized open-source agents.
- **Qwen backbones.** Qwen3.5-4B, Qwen3.5-9B, and Qwen3.5-35B-A3B support comparisons among the starting checkpoint, its oracle-harness configuration, and HarnessMind. Qwen3-VL-235B-A22B and Qwen3.5-397B-A17B provide larger-model reference results.
- **DeepEyesV2** (Hong et al., 2025) integrates visual reasoning, code execution, and web search within a tool-augmented reasoning process. Its training pipeline first establishes tool-use patterns through cold-start supervised fine-tuning, then uses reinforcement learning to refine context-dependent tool invocation and the coordination of complementary tools within a reasoning trajectory.
- **WebWatcher** (Geng et al., 2025) is a multimodal deep-research agent that combines text and image search, webpage navigation, and code execution. Supervised training on synthetic multimodal tool-use trajectories precedes reinforcement learning. We include its 7B and 32B variants.
- **OpenSearch-VL** (Chen et al., 2026b) provides an open training recipe for multimodal search agents. Its data pipeline combines multi-hop Wikipedia paths, entity rewriting, and visual grounding, while its tool environment integrates retrieval with image processing. Supervised initialization is followed by failure-aware multi-turn reinforcement learning to handle cascading tool failures.
- **SearchEyes** (Jiao et al., 2026) trains multimodal deep-search agents in a simulated search world built from a typed knowledge graph. Perception-Knowledge Chains retain hop-level entity information for task construction and environment simulation; Hop-Anchored Policy Optimization reuses these anchors for step-level credit assignment without a separate process reward model.

Controlled comparisons.

- **Base model.** The starting checkpoint uses the shared tools and execution system E without a task-specific external control policy. It establishes the starting performance before either external harness optimization or supervised student training.

- **ReAct** (Yao et al., 2023) interleaves explicit reasoning, tool actions, and environment observations in an iterative interaction loop. It provides a prompting-based comparison for tool use without the harness-derived process training of Harness2Policy.
- **Hand-designed harness.** This setting replaces automated harness search with a fixed, manually specified control configuration, isolating the contribution of the search procedure.
- **Searched and oracle harnesses.** Searched harnesses use control configurations selected on development tasks. The oracle baseline selects a benchmark-specific configuration using benchmark identity, whereas HarnessMind requires no such selection at inference.
- **Successful-path distillation.** This comparison uses successful paths from the same generation-policy search trees and matches the number of training target tokens. The best SFT recipe in Table 7 serves as the "w/o process compilation" baseline; the full method additionally preserves control decisions and supported failure reflections.

B Harness Components and Attribution Protocol

B.1 Harness Component Inventory

Source harnesses. Our component library draws on four independent source harnesses: **A** ImageSeeker (Deng et al., 2026) (the official DISBench harness, a LangGraph state graph with 7 nodes, 7 tools, conditional edges, turn budgets, and a fold node); **B** PACE-VL (a fixed unidirectional DAG with 13 sequential stages); **C** the BrowseComp/REDSearcher harness (a native tool-calling ReAct loop with 13 tools and context compaction); and **D** min-agent (a self-contained ReAct loop with 5 tools and per-tool call budgets). Cascaded fallback (A→B), dual-proposer fusion, and repeated-sampling voting form an external orchestration layer. These source-harness names identify component provenance, not the training task pools. The harness used by AgentVista is not part of this inventory.

Boundary between E and π_h. Tables 3 and 4 record source-harness implementations and their historical triggers. In model-native deployment, memory management and other retained control operations are exposed through a standard function-calling interface. The model chooses when to invoke them, while E executes the requests, presents observations, and enforces runtime constraints. Planning, verification, and reflection are expressed in model responses. Control internalization concerns the choice and timing of these actions; operations requiring external execution remain tools. Only external-harness settings use rule-driven scheduling and automatic prompt injection.

Historical trajectory statistics. In the Source trigger column, hard-coded means enforced by fixed code rules; configurable means set through runtime options; model-decided means chosen through tool calls or structured outputs; and prompt-only means requested in text without code enforcement. The descriptive statistics use 597 legacy DISBench trajectories from harness A and 366 from harness D, rather than the SFT corpus. For harness A, the mean, median, and 90th-percentile assistant turn counts are 11.7, 8, and 30 (the full budget), respectively, with an average of 96.3 rendered images per trajectory. In the legacy harness-D runs, the primary and larger models reach the full 14-turn budget in 52%/24% of episodes, stop producing tool calls without answering in 29%/44%, and finish with an explicit termination action in 20%/32%. The zero model-initiated fold calls in Table 4 describe the earlier harness, not the current interface, which exposes folding as a model-invoked tool and includes its calls in training supervision. These historical statistics do not define the reference-run groups in Figure 4. The removed visual-expert subagent and the wall-clock measurement that did not terminate episodes are not counted as active control components.

B.2 Attribution Protocol and Task Families

Attribution protocol. Table 5 separates the roles of task construction, harness design, trajectory generation, and answer verification. All student sizes use the same first-round corpus from G.

Generation and evaluation roles. The main route uses final-answer verification to select successful paths from G's searches. The design agent manages harness development and task synthesis, while G produces the supervised trajectory responses. Neither design diagnostics nor the answer verifier's

Table 3: Execution mechanisms supplied by the source harnesses. Control decisions and their roles in the learning interface are distinguished from these implementations in Table 4.

Component	Source	Mechanism
Graph topology and action dispatch	A	Source graph: assistant → tools → observation rendering → [fold] → turn check → assistant / forced finalization; 7 fixed nodes
Recursion limit and state schema	A	Execution-graph step limit of $3{,}000$ (never reached); fields include messages / photo-subsets / turns / max-turns / max-context-tokens
Tool exception recovery	A	Converts exceptions into observations that allow reasoning to continue (2.3% of episodes)
Observation rendering and image quotas	A	Reads tool-returned images → deduplicates → base64-encodes → combines into one message; drops and reports excess images (rendering occurs in 96.6% of episodes)
Failure-category reporting	A	Reports four categories to the model: invalid ID / missing file / encoding error / limit exceeded
Transport-layer retries	A/C/D	Whole-task retries (only for 429 / timeout / 522, with exponential backoff), model-call retries, and retrieval retries
Retrieval backend	A/D	Out-of-process HTTP retrieval service (FAISS) or in-process numpy inner-product retrieval
Image downsampling and re-encoding	D	Longest edge 896px; JPEG quality 82
Safety sandbox	A/C	Restricted eval for metadata filtering (the most frequent tool, $1{,}637$ calls); AST allowlist sandbox for code execution ($\leq 1{,}200$ nodes, 20s); URL SSRF guard
Context-overflow recovery	C	Catches overflow errors → halves the output limit + forces compaction before retrying
Global image index	C	Assigns each returned image a stable index for cropping and OCR references
Trace persistence	A/C	Writes traces per message or as an event stream

explanations are substituted for those responses. The branch-assessment comparison in Table 7 adds stronger-model reranking and verification of intermediate branches beyond final-answer checking. The remaining comparisons vary harness search or student-training procedures.

Task success and data validity. The generation policy executes from the task input and returned observations, without access to hidden answer fields. The task-specific leaf verifier $\mathcal{V}$ assesses final-answer correctness, generally by asking DeepSeek V4 Flash to judge equivalence between the generated answer and the reference answer. Image-reference, tool-call/observation, formatting, and context-length checks govern the SFT samples derived from successful paths. The verifier's judgment is used as selection feedback and is not supervised as response text.

Pilot study of trajectory rewriting. In a separate pilot, rewriting trajectory reasoning with a stronger model reduced explicit image inspection: ViewPhotos calls fell from 1.01 to 0.02 per trajectory, and performance declined relative to the pilot's base model. This result illustrates how trajectory rewriting can alter the tool-use behavior conveyed by supervision.

Task-family partitioning. The training pool contains eight task families from DeepImageSearch and RedSearcher. Each family contains multiple internal subsets for targeted harness evolution. Each evolution round samples 300 tasks from the combined training pool.

C Additional Experiments

C.1 Detailed Ablation Results

Table 6 reports the trained students' accuracies underlying Figure 4(b). Each ablation changes one design choice in data generation, compilation, or student training.

C.2 Model and Data Scaling

Figure 7 examines student model size and training-data volume. The model-scale comparison uses the same first-round corpus for all students, and each trained model exceeds its starting checkpoint with the searched harness. For data scaling, the horizontal axis gives the fraction of each variant's training corpus; the full method uses 1,162 SFT samples at 100%. Increasing the amount of compiled data improves accuracy, with an advantage over successful-path distillation at each evaluated data level. The same trend holds without hard-task synthesis.

Table 4: Source-harness component inventory, historical triggers, and roles in the learning and execution interface. Historical counts are not measurements of the trained students.

Component	Implementation in source harness	Source trigger	Historical statistics	Learning / execution role
Planning and orchestration				
Plan decomposition	B: compiles queries into constraint programs (anchor/target/bridge retrieval expressions + metadata filters + must-find-all); A/C/D: prompts only	Model-decided (schema-constrained) / prompt-only	—	Subgoals and constraints in model responses
Multi-run orchestration	Orchestration layer: A→B cascaded fallback, dual-proposer fusion and arbitration, repeated-sampling voting	Hard-coded	—	If retained, invoked by the model as a tool
Task-family gating	B: 11 functions that enable or disable entire branches by query keywords (handwritten if/else)	Hard-coded keywords	—	Removed: the model conditions directly on the query
Context and memory				
Fold trigger	A: token threshold (limit 65,536, buffer 20,000) or accumulated images > 80	Hard-coded thresholds	Legacy A: 26.8% of episodes (336 events); 0 model-initiated calls	Model invokes the folding tool; E updates context
Fold content policy	C: retains the first 2 and last 12 messages, truncates old tool results to 1,500 characters, and replaces old image messages with placeholder tags	Hard-coded	—	Context transformation executed on tool call
Named-subset memory	A/D: save-as / search-within / filter-within	Model-decided	806 metadata-filter save-as calls; 759 search-within calls	The model decides when to save subsets and search within them
Tool policy				
Tool routing	A: post-assistant router (terminate / tools / stall); C: tool aliases and rerouting (URL-based routing of visit); code execution available in C	Hard-coded rules	Every turn containing a tool call	Model selects tools; E dispatches calls
Per-tool call budgets	D: ImageSearch ≤ 6, ViewPhotos ≤ 4; excess calls return "budget exhausted" and urge convergence	Configurable	Legacy primary model exhausts budgets in 65% of episodes	Hard limits in E; model selects calls
Image-operation policy	A: quotas of 20 images per turn / 80 accumulated; D: 896px downsampling; C: zoom-in, zoom-in-search, OCR	Hard-coded (quota values searchable) / model-decided	Per-turn limit reached in 30.3% of episodes	Model selects operations; E enforces limits
Verification and anchoring				
Evidence verification	B: explicit verify stage (valid / maybe labels), three-level recovery after failure, strict rechecking under high support; A/C/D: prompts only	Model-decided (structured output) / prompt-only	—	Verification behavior in model responses
Cross-modal anchoring	B: derives target visual signatures and hard/soft constraints from anchor images, rewrites retrieval expressions, and injects them into every subsequent verification prompt; prototype-image selection (≤ 2 images)	Model responses / external injection	—	Model-led anchoring; retained external operations use tools
Answer checking	B: global final selection over ≤ 48 candidates; A/C/D: "verify before answering" appears only in prompts	Model-decided / prompt-only	—	A checking segment before the final answer
Execution protocol and auxiliary controls				
Termination rules	A/C/D: FinalAnswerTool or text-based termination conventions (regex fallback over 6 marker types); A: forced-finalization node; C: disables tools and forces an answer when the budget is exhausted	Model response + hard-coded fallback	100% text termination in A; legacy primary model: 20% FinalAnswerTool use, 29% silent stalls	Answer protocol and fallback, not a separate learned control
Turn budgets and warnings	A: max-turns 30, with a "must finish" warning at turn 29; D: max-turns 14	Configurable	A warns in 14.1% of episodes, p90 turns $= 30$; legacy primary model: 52% use all 14 turns	Runtime constraints in E
Stall nudges and periodic self-checks	A: injects SYSTEM ALERT when neither a tool call nor a termination marker appears; C: every 6 turns, prompts a "summary of confirmed / unresolved / contradictory evidence"	Hard-coded	Stall nudges in 5.0% of A episodes	Self-checks initiated by the model

C.3 Distillation and Harness Search

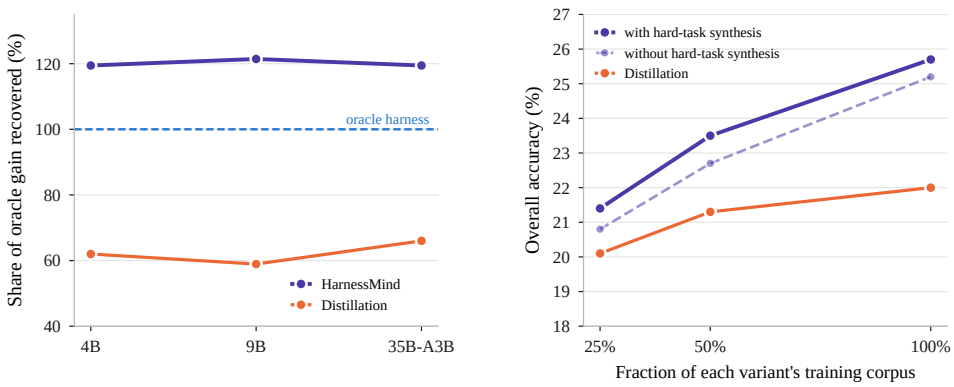
Table 7 compares SFT recipes on successful paths from the same search trees, with matched training target tokens. The best recipe reaches 22.0% and serves as the "w/o process compilation" baseline; direct best-path distillation scores 20.9% without the added recipe refinements.

Table 5: Roles in task construction, harness search, and supervised data generation.

Factor	What it determines	Source
Training tasks	Synthesized tasks and difficult candidates for exploration	Claude Code synthesizes DeepImageSearch tasks; generation-policy rollouts guide RedSearcher task selection
Harness structure	External control configurations used to guide execution	The Claude Code design agent selects and refines components from task scores and execution traces
Training responses	Actions, reasoning, and supported reflections retained for SFT	The generation policy G produces the response text and tool calls through tree search
Answer verification	Whether a completed path is accepted as successful	A task-specific verifier, generally DeepSeek V4 Flash, compares the generated answer with the reference answer

Table 6: Ablation accuracy (%). The final column reports the reference-run group above 20 turns.

Setting	DISBench	AgentVista	BC-VL	MM-BC	Overall	Over 20 turns
Hand-designed harness	6.6	15.8	40.1	9.8	23.4	10.2
Unconstrained search	5.7	14.8	39.3	9.8	22.7	7.8
w/o process compilation	6.6	15.8	37.3	8.9	22.0	7.4
No failed-branch experience	6.6	15.8	39.8	9.8	23.3	9.8
No hard-task synthesis	7.4	16.7	42.9	11.2	25.2	13.5
Control prompts retained	6.6	16.3	40.1	10.3	23.6	11.1
HarnessMind (full)	**7.4**	**17.2**	**43.6**	**11.6**	**25.7**	**14.3**



(a) Model scale (b) Data scale

Figure 7: Effects of student model size and training-data volume. (a) Accuracy gain after training as a percentage of the gain from adding the searched harness. (b) Primary-model accuracy as the fraction of each variant's training corpus increases.

The advantage of process compilation persists under stronger distillation recipes motivated by prior agent-training methods (Luo et al., 2026; Kang et al., 2025; Liu et al., 2026a). Alternative harness proposal strategies also retain much of the improvement. Free-form harness instructions achieve 26.9%, versus 25.7% for our component-based search with its explicit control structure.

D Method Implementation Details

D.1 Stage A: Harness Search with a Design Agent

Design agent and feedback. We automate harness development with Claude Code backed by Claude Opus 4.8. The design agent inspects the available control modules and evaluates candidate configurations, using task scores, execution traces, and failure diagnoses to retain, remove, revise, or combine components. Its diagnostic text guides subsequent proposals and evaluations rather than replacing the generation policy's responses in the supervised training corpus.

Table 7: Primary-model accuracy (%) across training, verification, and harness search settings.

Setting	Overall accuracy
Direct distillation of best leaf paths	20.9
+ first-thought	21.3
+ segment-wise loss	21.6
+ process-aware SFT adaptation	22.0
Route A (task-specific final-answer verification)	25.7
Route B (additional branch reranking and verification)	26.6
Weaker harness proposal model (same search budget)	25.2
Random harness search (same search budget)	23.9
Strong model writes free-form instructions	26.9

Candidate records. Each node in the configuration tree represents a complete, executable harness. The record stores its parent configuration, the changes made, the evaluation results, and relevant failure observations. The agent uses these records to guide subsequent proposals and select the candidate configurations to evaluate next, as illustrated in Figure 2.

Evaluation and selection. Each harness-evolution round resamples 300 tasks from the full training pool as a development subset. Candidate harnesses are evaluated with the current student S_k, keeping its checkpoint, sampled tasks, tool interfaces, and execution limits fixed within that round's search. Independent evaluations can run in parallel with separate execution states. Invalid configurations and infrastructure failures are distinguished from task failures. The design agent proposes further changes from completed evaluations, and the best-performing valid configuration is retained. Comparisons between proposal strategies use the same search budget; search stops when that budget is exhausted or no further candidate is pursued. Neither harness search nor data construction uses test tasks.

Reuse across training rounds. Selected components and their execution records support subsequent search and training-data collection. In iterative experiments, harness evaluation is repeated for the updated student, and the generation policy G supplies fresh trajectories for continued student training. Deployment uses the shared tools and execution system without selecting a harness.

D.2 Stage B: Exploration and Trajectory Compilation

Training task construction. We build an additional training task pool from DeepImageSearch and RedSearcher sources without using test tasks. For DeepImageSearch, Claude Code synthesizes 500 difficult tasks from 100 original user photo-album collections. For RedSearcher, we start from a pool of 5,000 training tasks and obtain rollouts using the generation model. We rank tasks by the number of tool calls in these rollouts and retain the top 600 as difficult candidates. Candidate-data cleaning applies `correct=true` and high-confidence filtering, repairs formatting, and removes hedged answers. These steps select and clean the task pool; they are distinct from the final-answer verification used for subsequently generated SFT trajectories. The RedSearcher source provides selected tasks rather than newly synthesized tasks.

Exploration and verification. Trajectory collection covers all training tasks, not only the 300-task development subset. For each task, G conducts two independent tree searches under the selected harness. Each search starts from the task input, branches over alternative action continuations, and records tool calls together with returned observations. A task-specific verifier assesses the final answer at a completed leaf, generally using DeepSeek V4 Flash to judge equivalence to the reference answer. Each search independently selects its shortest successful path for training-data construction; a search without a successful path contributes no supervised solution. The accepted paths supply reasoning and tool-call supervision for rejection fine-tuning.

Compiling control experience. Planning, memory, and verification segments retain the generation policy's outputs together with the action sequence they support. Context-folding tool calls are prediction targets, while the executor performs the resulting context update. Memory summaries and saved-subset operations remain linked to the observations they summarize or organize. Candidate reflections come from G's diagnoses of failed alternatives; when a diagnosis is absent, G generates

one from the corresponding exploration record. The compiler assembles these outputs without substituting the design agent's reasoning or the answer verifier's comments. Planning, verification, and reflection appear in responses and tool-use decisions rather than dedicated tools.

A reflection is retained only when its factual premises are supported by the actual training prefix at its insertion point, after removing harness-specific prompts and accounting for preceding context folds. Reference answers, future observations, and discarded branch observations are not admissible evidence. Training omits failed-branch observations from inputs and failed actions from targets.

Data processing and corpus size. Final-answer correctness is verified on each complete selected path. When a path contains a context fold, the pre-fold and post-fold executions are exported as separate SFT samples, and both enter training. The pre-fold sample supervises actions through the fold call; the post-fold sample starts from the executor's folded context, without restoring discarded observations. Each sample preserves its effective input context and generated response targets. Image-reference, tool-call/observation, and formatting checks and the 40,960-token context limit apply to individual samples. The first-round corpus contains 1,162 SFT samples: 746 from DeepImageSearch and 416 from RedSearcher. All evaluated students share these samples from G.

D.3 Stage C: Training and Removal of Control Prompts

Student training and optimization. We train the students with LLaMA-Factory using the shared generation-policy corpus. Fine-tuning updates the language model while freezing the vision encoder and multimodal projector. Training uses eight H20-141GB GPUs, a maximum sequence length of 40,960 tokens, a learning rate of 0.00001, and three epochs. The supervised objective in Equation 1 applies to the generation-policy responses, including reasoning and tool calls. Retained tool outputs and image observations remain in the input context and are excluded from the prediction loss. Subsequent rounds continue training the primary student with newly generated trajectories from G.

Control prompts and deployment. We remove harness-specific control prompts from supervised inputs while retaining the deployment tool definitions and execution interface. The student learns planning and verification responses and autonomously invokes memory management and other externally executed operations through standard function calls. E carries out the requested operations and enforces runtime constraints; no external rule schedules control actions or injects reminders in model-native deployment. The student uses no family labels or task-specific harnesses at inference.

Controlled comparisons. Successful-path distillation and process compilation use the same generation-policy search trees and matched numbers of training target tokens. The former retains successful action sequences, while the latter also preserves control decisions and supported failure reflections. Comparisons use the same tools and execution interface. The additional training pool supports harness development and trajectory collection; neither stage uses benchmark test tasks.

D.4 External-Control Evaluation across Rounds

Fixed and renewed harnesses. Figures 6(a) and 6(b) distinguish the original harness from newly searched guidance. Figure 6(a) keeps the round-0 harness fixed. Figure 6(b) renews harness search for each student using that round's newly sampled training-development subset. Gains are test-accuracy differences, in percentage points, between each student with and without the evaluated harness. Figure 6(c) reports auxiliary guidance selected across the eight training-task families and their internal subsets; shared tools and runtime safeguards remain available in E.

Repeated training and the comparison model. Each teacher-assisted round renews harness search and trains S_{k+1} from S_k using new trajectories from G. The ordinary-trajectory comparison student receives an equal amount of data and undergoes the same harness evaluation. Both procedures are assessed on the sampled training-development subset and evaluated separately on test sets.

E LIMITATIONS

Harness2Policy is evaluated on four benchmarks with three Qwen3.5 backbones under a shared tool and execution interface. These controlled settings do not establish robustness to other model families, substantially different tool ecosystems, or environments that change over time. Harness search is also bounded by a predefined component inventory and a finite evaluation budget, so strategies outside this space may require additional components and training experience.

Supervision depends on the generation policy and the task-specific verifier. Only searches with a verified successful path contribute solution targets, leaving tasks that remain unsolved with limited training coverage. A correct final answer does not ensure that every intermediate decision or reflection is reliable. Requiring reflections to be supported by the student's available context restricts their evidence sources, but does not fully validate the reasoning.

The iterative experiments use a separate generation policy to supply fresh trajectories for student training. Whether their gains persist over many rounds or extend to a fully student-driven loop remains open. A declining benefit from external guidance also does not by itself distinguish control internalization from broader capability gains. In addition, inference-token savings exclude the offline costs of harness search, tree exploration, and fine-tuning.

F BROADER IMPACTS AND FUTURE DIRECTIONS

Scope and potential benefits. Harness2Policy treats harness optimization as a source of reusable learning experience rather than only an inference-time control mechanism. This could reduce dependence on hand-designed orchestration in visual search and other extended tool-use workflows. The observed gains in long-horizon accuracy and inference efficiency suggest practical value, while the separation of search, trajectory construction, and student training leaves room to improve each stage while preserving the shared deployment interface.

Future directions. A central direction is to connect harness search with student-generated exploration and continually refreshed training data, moving toward recursive co-improvement of harnesses, data, and foundation models. Reliable verification and diverse tasks will be essential to avoid reinforcing earlier errors or overfitting to a narrow task distribution. Further work can expand the control inventory, assess transfer across model families and changing tool interfaces, and strengthen verification of intermediate decisions alongside final answers. Budget-aware search and selective trajectory collection may reduce offline costs while preserving coverage of difficult cases.

Broader impacts. Agents that sustain longer workflows may reduce user effort, but errors can also propagate through more tool actions before they are detected. Model-native control should therefore complement rather than replace runtime safeguards. Practical deployments should enforce permission limits, maintain auditable action records, and require human approval for consequential or irreversible actions. Training and evaluation data also require appropriate authorization and privacy protection, particularly when they contain personal images or browsing traces. These safeguards remain necessary even when benchmark accuracy improves.