VeriSeek-VL: Seeking Visual Evidence and Verifying Hypotheses in Multimodal Deep Search

Anonymous authors
Paper under double-blind review

Abstract

Multimodal deep search agents solve long-horizon information-seeking tasks by iteratively retrieving and reasoning over textual and visual evidence. However, existing methods face two limitations: insufficient active acquisition of visual evidence and premature hypothesis lock-in. First, when key visual evidence is not explicitly provided, agents tend to rely on textual evidence and rarely actively acquire external visual evidence. Second, agents often treat an insufficiently supported hypothesis as fact in subsequent reasoning, so a wrong hypothesis leads the search away from the correct answer. To address these limitations, we propose **VeriSeek-VL**, a multimodal deep search framework that integrates data synthesis, training, and inference-time control. First, to strengthen active acquisition of visual evidence, we construct multimodal hypergraphs whose visual hyperedges link multiple entities through image-derived relations. Embedding these hyperedges into multi-hop reasoning chains yields tasks with implicit visual constraints, where key intermediate entities can be identified only through external visual evidence. Through supervised fine-tuning and GRPO on these tasks, the model learns to actively acquire external visual evidence. Second, to mitigate premature hypothesis lock-in, we identify a hypothesis lock-in axis through trajectory auditing and activation analysis. This axis is a linear direction in activation space that distinguishes lock-in steps from normal reasoning steps. During inference, we project each step's mean activation onto the axis and trigger verification of key hypotheses when the projection exceeds a threshold, reducing the risk of searching under a wrong hypothesis. Extensive experiments show that VeriSeek-VL achieves state-of-the-art performance among open-source methods on all five multimodal search benchmarks, surpassing the strongest baseline by 6.32% on average. Training improves the base model by 25.5%, and axis-guided verification adds a further 3.8%.

1 Introduction

Multimodal deep search agents solve long-horizon information-seeking tasks by iteratively retrieving and reasoning over textual and visual evidence (Wu et al., 2025; Geng et al., 2025; Huang et al., 2026). In open-web settings, the required evidence is often spread across webpages, images, and scanned documents, and the question rarely indicates where it can be found. Solving such tasks therefore requires two abilities: (1) *seeking* the evidence that a question implicitly depends on, including visual evidence, and (2) *verifying* intermediate hypotheses before building further search on them. Recent work has advanced multimodal deep search through iterative textual and visual retrieval (Wu et al., 2025; Huang et al., 2026), dedicated data construction (Geng et al., 2025; Chen et al., 2026; Zhang et al., 2026a), reinforcement learning (Wu et al., 2025; Geng et al., 2025; Chen et al., 2026), and long-horizon context management (Liu et al., 2026; Du et al., 2026).

However, existing methods remain limited in both abilities. For seeking, they exhibit insufficient active acquisition of visual evidence: when key visual evidence is not explicitly provided, agents tend to rely on textual evidence and rarely make active use of external visual evidence. As shown in Figure 1, the question refers to the fruits in a painting without supplying the painting; textual descriptions mention only some of these fruits, so an agent that searches only text misses part of the answer. We attribute this largely to existing training data, where key images are either provided with the question or unnecessary for the answer, so agents are seldom required to seek them. A similar

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.

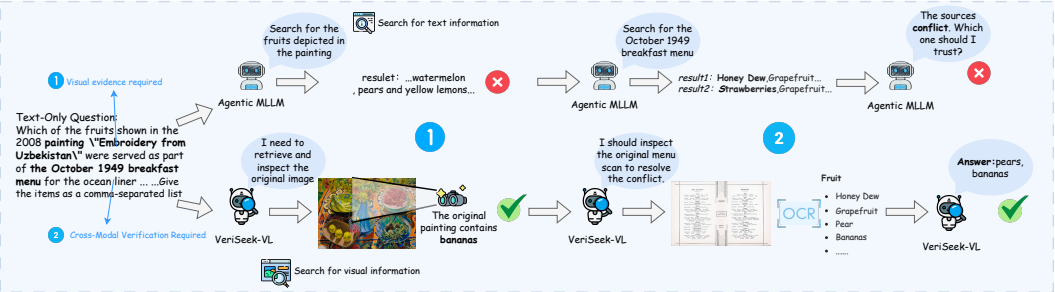


Figure 1: Comparison between an agentic MLLM (top) and VeriSeek-VL (bottom) on a text-only question. Without being given the painting or menu images, the agent must retrieve the original materials, identify visual clues, and verify conflicting evidence.

shortcut has been observed in existing multimodal search benchmarks, where answers that should require visual search can often be obtained from textual cues or prior knowledge (Zeng et al., 2026).

For verification, existing methods suffer from premature hypothesis lock-in. By analyzing search trajectories, we find that agents often accept an insufficiently supported hypothesis as fact and build subsequent reasoning on it. Because the agent regards the hypothesis as established, it seldom checks it again, and if the hypothesis is wrong, the search gradually deviates from the correct answer. These two limitations naturally raise a research question: *How can we enable an agent to actively seek visual evidence that the question does not explicitly provide, and to recognize when a hypothesis it already treats as fact should be verified?*

To answer this question, we propose **VeriSeek-VL**, a multimodal deep search framework that strengthens both abilities by integrating data synthesis, model training, and inference-time control (Figure 2). First, to strengthen seeking, we synthesize training tasks that require visual evidence. We construct multimodal hypergraphs whose visual hyperedges link multiple entities through image-derived relations, and embed them into multi-hop reasoning chains (Figure 3). The resulting tasks carry implicit visual constraints: key intermediate entities can be identified only through external visual evidence. Through supervised fine-tuning (SFT) and GRPO (Shao et al., 2024) on these tasks and public multimodal search data, the model learns to actively acquire visual evidence.

Second, to strengthen verification, we need a signal of when to verify that does not depend on the agent's own judgment, since the agent already treats its hypothesis as fact. We therefore turn to the model's internal states, motivated by prior work showing that agent failure behaviors can be characterized by directions in activation space (Sui et al., 2026). Through trajectory auditing and activation analysis, we identify a hypothesis lock-in axis, a linear direction in activation space that distinguishes lock-in steps from normal reasoning steps. Based on this axis, we design HLI-Guard for axis-guided verification: during inference, it projects each step's mean activation onto the axis and triggers verification of key hypotheses when the projection exceeds a threshold, so that the agent verifies a hypothesis before building further search on it.

Extensive experiments on five multimodal search benchmarks (MMBrowseComp, BrowseComp-VL, MMSearch, MMSearch-Plus, and VisBrowse) show that VeriSeek-VL achieves state-of-the-art performance among open-source methods on all five, reaching an average score of 56.88 and surpassing the strongest baseline on each benchmark by 6.32% on average. Training improves the base model by 25.54% on average, and axis-guided verification adds a further 3.78%.

Our main contributions are as follows:

1. We identify two limitations of existing multimodal deep search agents, insufficient active acquisition of visual evidence and premature hypothesis lock-in, which correspond to deficiencies in seeking and verification, respectively.
2. We propose VeriSeek-VL, a multimodal deep search framework that integrates data synthesis, model training, and inference-time control. It synthesizes tasks with implicit visual constraints from multimodal hypergraphs and trains agents on them with SFT and GRPO to actively acquire external visual evidence.

3. We identify a hypothesis lock-in axis in the model's activation space that distinguishes lock-in steps from normal reasoning steps, and design HLI-Guard, which uses this axis to trigger verification of key hypotheses at inference time.
4. Extensive experiments on five multimodal search benchmarks show that VeriSeek-VL achieves state-of-the-art performance among open-source methods, surpassing the strongest baselines by 6.32% on average.

2 METHOD

As shown in Figure 2, VeriSeek-VL combines data synthesis and model training with HLI-Guard, an inference-time controller guided by the **Hypothesis Lock-in Axis**. Section 2.1 describes how we synthesize question–answer pairs with implicit visual constraints, collect teacher-generated interaction trajectories, and train the model with SFT and GRPO for active visual evidence acquisition. Section 2.2 presents the axis discovered through trajectory auditing and activation analysis, and explains how HLI-Guard uses it to trigger hypothesis verification and search backtracking when needed.

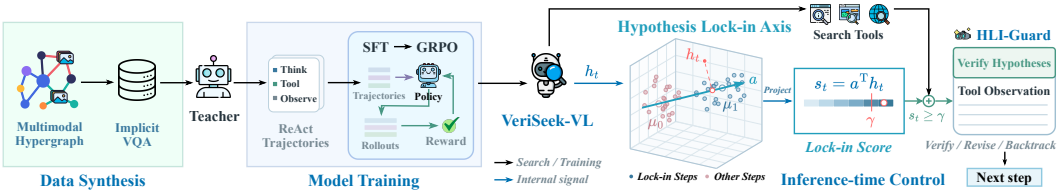


Figure 2: Overview of VeriSeek-VL. Synthetic QA pairs support teacher trajectory collection and SFT/GRPO training. The Hypothesis Lock-in Axis is extracted from audited step activations. HLI-Guard uses the resulting Lock-in Score to selectively prepend a verification prompt to tool observations. The three-dimensional activation space and score scale are schematic.

2.1 SEEKING: TRAINING WITH IMPLICIT VISUAL CONSTRAINTS

We synthesize question–answer (QA) pairs with *implicit visual constraints* to train active retrieval and use of external visual evidence. These questions describe key intermediate entities through visual relations in external images, retaining clues for locating the images and identifying the entities but providing neither the images nor the target identities. The pipeline has three stages (Figure 3): constructing a multimodal hypergraph, selecting multi-hop dependencies that traverse visual relations, and rewriting entity names into implicit visual constraints.

Multimodal hypergraph construction. We construct a multimodal hypergraph from Wikipedia page text and Wikimedia Commons images, using Wikipedia entities as nodes:

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}_T, \mathcal{E}_V), \tag{1}$$

where $\mathcal{V}$ is the set of entity nodes, $\mathcal{E}_T$ is the set of textual relation edges, and $\mathcal{E}_V$ is the set of visual hyperedges.

Each node stores the entity's name and textual description, with single-entity images as visual attributes. A textual relation edge is written as $e_T = (u, r, v)$, where r is an entity relation extracted from the page text. We use each original multi-entity image as a visual hyperedge connecting its associated entities:

$$e_V = (V_e, I_e, b_e), \tag{2}$$

where I_e is the image, $V_e \subseteq \mathcal{V}$ is the set of entities it connects, and b_e contains contextual clues for locating the image. Seed selection and node/image source metadata are detailed in Appendix B.

Starting from the seed entities, we progressively construct the hypergraph through textual and visual expansion. Textual expansion follows links on entity pages to discover new entities and extract relations. Visual expansion selects multi-entity images associated with the current entities and adds both the images and the entities they connect to the hypergraph. We retain images that can be located using contextual clues and whose associated entities can be linked to Wikipedia pages, together with the corresponding entity–image mappings.

Constructing multi-hop solution dependencies. Given the hypergraph, we sample a connected task subgraph $\mathcal{G}_q = (\mathcal{V}_q, \mathcal{E}_{T,q}, \mathcal{E}_{V,q})$ by traversing textual edges and visual hyperedges from a starting entity. We bound the subgraph's node count and expansion depth and require a minimum number of visual hyperedges (Appendix B). We then select relations that jointly determine the answer and organize them into an acyclic reference dependency structure. These relations may form sequential dependencies or constraints that must be satisfied jointly. The reference solution for each question traverses at least one visual hyperedge, so identifying a key intermediate entity depends on visual relations in an external image.

Using the selected relations, we ask a multimodal large language model to determine the reference answer and generate an initial question with explicit entity names. We then replace the names of starting and intermediate entities with relational or attribute descriptions in reverse topological order of the reference dependencies. This rewriting preserves the reference answer and the dependencies between entities. Entities involved in visual hyperedges are rewritten using the implicit visual constraints described below.

Generating implicit visual constraints. For each visual hyperedge traversed by the reference reasoning chain, we select a reference entity and a target entity that can be identified in the image. Clues in the question or preceding reasoning steps establish the reference entity, whereas the target must be identified through the current visual relation. We extract distinguishing visual information about the target from the original image and replace its name with an implicit visual constraint:

$$c_j = (b_j, \rho_j, f_j), \tag{3}$$

where b_j contains contextual clues in the question for locating the image, ρ_j specifies how the reference entity is described in the question, and f_j describes the target's distinguishing visual features, including appearance attributes, spatial relations, or interactions.

Contextual clues and the reference entity guide image retrieval; the visual description identifies the target needed for the next hop. In Figure 3, clues about a Bell Labs publicity photograph and its three Nobel laureates locate the image. The man on the right with a moustache and a patterned tie is Walter Brattain, whose identity allows the search to continue toward Whitman College.

We retain visually checkable features that distinguish the target and remove names, aliases, and other identifying text. Combining these constraints with the surrounding textual relations yields the final text-only question. Here, *implicit* means that the need for visual evidence is embedded in the entity descriptions, while the corresponding images must be retrieved externally.

Each visual hop in the reference reasoning chain produces one implicit visual constraint. Let $\mathcal{C}(q) = \{c_j\}_{j=1}^{k(q)}$ denote the constraint set for question q. We vary the number of constraints within $1 \leq k(q) \leq n$ to generate questions with different numbers of visual dependencies. For each constraint, we retain its target entity and reference image, preserving the correspondence between the question's description, the entity's identity, and the visual evidence.

Question filtering, trajectory collection, and model training. We first run a solver without external tools on the synthesized questions and remove those it answers correctly. For the retained QA pairs, we store the reference answer, the reference dependency structure, and the entities and image sources associated with each implicit visual constraint. These records support subsequent trajectory auditing.

We then use a teacher model to collect complete solution trajectories in a ReAct interaction setting (Yao et al., 2022). The teacher reasons, calls tools, and observes their outputs, gathering evidence through text search, webpage browsing, and image search until it produces a final answer. We retain trajectories with correct final answers that actually invoke image search, and discard those with excessive image-search calls or interaction lengths outside the prescribed range.

To broaden the types of questions covered during training, we combine the synthesized QA pairs with public multimodal QA and search datasets. SFT uses the filtered interaction trajectories to teach the model to retrieve external images, identify key visual clues, and continue the search. We then apply GRPO reinforcement learning (Shao et al., 2024), using the QA data to define training tasks. The reward combines final-answer correctness, format compliance across the trajectory, and a soft penalty for overly long single-turn outputs. A trajectory receives full format credit only if every step follows the required interaction format. Data sources, the teacher model, filtering thresholds, and training configurations are detailed in Appendix C.

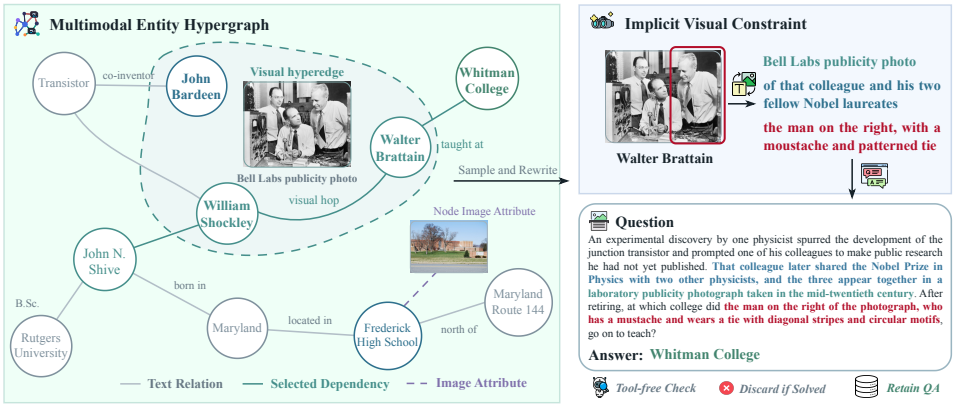


Figure 3: Synthesizing QA pairs with implicit visual constraints. Multi-entity images form visual hyperedges; single-entity images supplement node attributes. A sampled dependency identifies an intermediate entity through visual evidence. Rewriting retains image-retrieval clues and distinguishing visual features, while omitting the reference image and target identity. The example leads to Whitman College; questions answered correctly without tools are discarded.

2.2 Verifying: Hypothesis Lock-in Axis and HLI-Guard

The synthesis and training pipeline develops active visual evidence acquisition. We next ask when an intermediate hypothesis should be verified before further search is built on it. Through trajectory auditing and activation analysis, we discover a linear direction that distinguishes lock-in onset steps from other search steps, which we call the **Hypothesis Lock-in Axis**. HLI-Guard uses this signal to trigger verification and, when needed, search backtracking.

Hypothesis lock-in and complete-trajectory auditing. During long-horizon search, an agent may treat an insufficiently verified intermediate hypothesis as an established premise and narrow subsequent search around it. If the hypothesis is incorrect, further retrieval may continue to move away from the correct answer. We call this behavior *hypothesis lock-in*.

To locate the relevant steps, we audit evidence across search trajectories. For a question q, let

$$\tau = (q, r_1, o_1, \ldots, r_T, o_T), \tag{4}$$

where r_t contains the reasoning and tool call at step t, and o_t is the corresponding tool observation. The audit also consults the intermediate entities, implicit visual constraints, and reference images retained during data synthesis. These references help assess whether the evidence obtained in the trajectory resolves the relevant entity identities and verifies the constraints.

We call a conjecture about an intermediate entity a *search-critical hypothesis* if accepting it would affect subsequent retrieval or answer derivation. For a hypothesis h, let $t_0(h)$ be the step where it is first proposed. We determine its verification status from the complete trajectory:

$$z_\tau(h) = \begin{cases} 1, & \text{external evidence in } \tau \text{ sufficiently supports or refutes } h; \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

We trace each hypothesis that remains unverified at the end of the trajectory back to its first occurrence, yielding the step label

$$y_{\tau,t} = \mathbb{I}[\exists h : \; t = t_0(h) \wedge z_\tau(h) = 0]\,. \tag{6}$$

Here, $y_{\tau,t} = 1$ denotes a lock-in onset step, and $y_{\tau,t} = 0$ denotes any other search step. If a step introduces several key hypotheses, it is positive whenever at least one remains unverified.

An LLM-based trajectory auditing pipeline produces these labels, and annotation agreement is evaluated against a gold-standard set. Appendix D describes the audit procedure and gold-set setup.

Lock-in onset steps correlate with lower final-answer accuracy: trajectories with 0, 1, and 2+ onset steps achieve 28.68%, 18.24%, and 12.21%, respectively (Figure 4c).

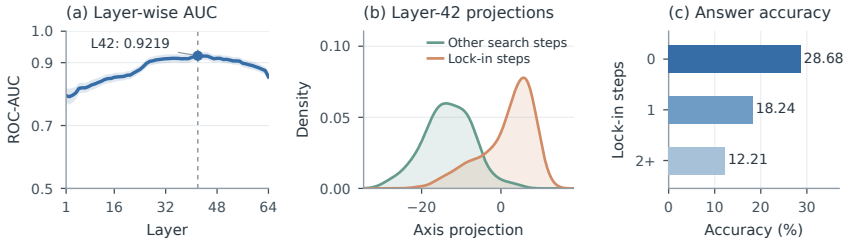


Figure 4: Hypothesis lock-in analysis on the base Qwen3-VL-32B-Thinking model. (a) Five-fold mean ROC-AUC; shading shows ± 1 SD. (b) Smoothed class-conditional distributions of held-out layer-42 projections. (c) Accuracy for trajectories with 0, 1, or 2+ lock-in onset steps.

Step activations and the Hypothesis Lock-in Axis. The Assistant Axis characterizes the assistant persona through a linear activation direction (Lu et al., 2026). Inspired by this finding, we examine whether hypothesis lock-in is also linearly decodable from model activations. We pair the audit labels with mean step activations and analyze each layer. For each step, we average the residual-stream activations over its thinking tokens, whose positions are denoted by $\mathcal{I}^{\text{think}}_{\tau,t}$. The mean activation at layer ℓ is

$$\mathbf{h}^{(\ell)}_{\tau,t} = \frac{1}{|\mathcal{I}^{\text{think}}_{\tau,t}|} \sum_{j \in \mathcal{I}^{\text{think}}_{\tau,t}} \mathbf{h}^{(\ell)}_{\tau,t,j} \in \mathbb{R}^d. \tag{7}$$

At each layer, let $\boldsymbol{\mu}^{(\ell)}_c$ be the mean activation of fitting-set steps with label $c \in \{0, 1\}$. We normalize the difference between the class means to obtain the axis and project each step onto it to obtain its **Lock-in Score**:

$$\mathbf{a}^{(\ell)} = \frac{\boldsymbol{\mu}^{(\ell)}_1 - \boldsymbol{\mu}^{(\ell)}_0}{\|\boldsymbol{\mu}^{(\ell)}_1 - \boldsymbol{\mu}^{(\ell)}_0\|_2}, \tag{8}$$

$$s^{(\ell)}_{\tau,t} = \left(\mathbf{a}^{(\ell)}\right)^\top \mathbf{h}^{(\ell)}_{\tau,t}. \tag{9}$$

The axis points from other search steps toward lock-in onset steps. Higher scores place activations farther toward its lock-in side, providing an internal risk signal for triggering verification. Class-mean estimation is detailed in Appendix D.

On the base Qwen3-VL-32B-Thinking model, five-fold evaluation yields the highest mean axis ROC-AUC at layer 42 (0.9219; Figure 4a). Figure 4b shows held-out projection distributions. Axes are estimated separately for each checkpoint; further analyses are in Appendix F.

Selective verification using axis projections. HLI-Guard scores each search step after it is generated, using the selected readout layer $\ell^\star$. We write its projection score as s_t. The trigger threshold is set in the upper tail of the score distribution over all calibration steps:

$$\gamma_\alpha = Q_{1-\alpha}\left(\left\{s^{(\ell^\star)}_{\tau,t} : (\tau, t) \in \mathcal{D}_{\text{cal}}\right\}\right), \qquad g_t = \mathbb{I}[s_t \geq \gamma_\alpha], \tag{10}$$

where $Q_{1-\alpha}$ is the $(1-\alpha)$-quantile and α controls the size of the upper tail in the calibration distribution. Readout layers and threshold settings are specified in Appendix E.

When $g_t = 1$, HLI-Guard uses an auxiliary model to organize the available evidence and extract key entity hypotheses in the triggering step that remain unverified. The step's tool call executes normally, and the controller prepends a verification prompt to its returned content. This prompt guides the agent to seek evidence that supports or refutes the hypotheses in subsequent search.

Hypothesis tracking, context compression, and backtracking. HLI-Guard maintains a set of pending hypotheses $\mathcal{U}_t = \{(h, t_0(h))\}$, recording each unverified hypothesis and its first-occurrence step. The controller updates this set only when HLI-Guard is triggered. It reevaluates existing hypotheses using newly acquired tool evidence, removes those that are sufficiently supported or refuted, and adds newly identified unverified hypotheses. The verification outcomes and associated evidence are retained.

The pending hypotheses determine how much history can be compressed. When $\mathcal{U}_t$ is nonempty, let

$$b_t = \min_{(h,j)\in\mathcal{U}_t} j, \tag{11}$$

be the earliest step at which a still-pending hypothesis was introduced. The controller compresses the history before b_t into an evidence summary, retaining relevant facts, verification outcomes, and source identifiers. It preserves the full interaction from step b_t onward. Further compression occurs only when earlier hypotheses are resolved and the boundary advances. This keeps unresolved hypotheses and the evidence collected after them in the full context.

When new evidence refutes a candidate hypothesis underlying the current search, HLI-Guard marks it as excluded. It then reconstructs the context from the evidence and hypothesis state stored at the corresponding step, guiding the agent toward alternative candidates. If conflicting evidence affects multiple hypotheses, it backtracks to the earliest affected step. Candidate update rules, the complete control procedure, and prompts are provided in Appendix E.

3 Experiments

We evaluate VeriSeek-VL on five multimodal search benchmarks and assess the effects of model training and inference-time control.

3.1 Experimental Setup

Training data and optimization. We initialize from Qwen3-VL-32B-Thinking (Bai et al., 2025). The filtered SFT corpus contains 14,203 trajectories, split into 13,919 training and 284 validation trajectories; RL uses 6,696 training questions. Our QA pairs with implicit visual constraints account for 46.95% of the pre-split SFT corpus and 32.21% of the RL questions. Other sources are REDMM (Chu et al., 2026), LiveVQA (Fu et al., 2025), and FVQA, with VDRBench (Zeng et al., 2026) added for RL. Appendix C provides the synthesis and filtering settings, source counts, and tool specifications.

SFT updates the language model while freezing the vision encoder and alignment module, using learning rate 2×10^{-5}, batch size 128, and a 49,152-token sequence limit; we select checkpoint 320. RL starts from this checkpoint, keeps the vision encoder frozen, and uses a GRPO-style clipped policy objective (Shao et al., 2024) with RLOO advantages (Ahmadian et al., 2024). We sample eight trajectories per question and use the mean reward of the other seven as the baseline, without group standard-deviation normalization or KL regularization. The learning rate is 10^{-6}, the rollout batch contains 64 questions, and we select checkpoint 100. Full optimizer, rollout, and hardware settings are in Appendix C.

Confidence-based routing. Only trajectories whose outcomes can be reliably attributed to model decisions enter policy updates. These high-confidence trajectories complete normal tool interactions, submit an answer, and pass reliability checks; answers may be correct or incorrect. Their reward combines correctness, format validity, and a length term with weights 0.8, 0.1, and 0.1. Low-confidence trajectories, such as tool-service failures or degenerate repeated searches, receive zero reward and are fully masked from the loss, but remain in the RLOO baseline. This prevents reward hacking through format or length bonuses on unreliable trajectories. Reward formulas and turn-resampling rules are given in Appendix C.

Benchmarks and baselines. We evaluate on MMBrowseComp (Li et al., 2025), BrowseComp-VL (Geng et al., 2025), MMSearch (Jiang et al., 2024), MMSearch-Plus (Tao et al., 2025), and VisBrowse (Zhang et al., 2026b), using official scoring protocols. Following prior work, MMSearch-Plus uses its single-image subset; the other benchmarks use their full public test sets. Our main results average four independent runs per question; timeouts and missing answers count as incorrect. Overall scores average accuracy across the five benchmarks.

We compare direct answering, general-purpose models in ReAct workflows, and specialized multimodal deep search agents. ReAct baselines adaptively select actions from the same tools as VeriSeek-VL based on accumulated observations.

Inference settings. Our SFT and RL checkpoints allow 40 interaction turns; FULL adds HLI-Guard to the RL checkpoint and allows 100. HLI-Guard reads layer 42 of the final RL checkpoint using

Table 1: Main results on five multimodal search benchmarks (accuracy, %). Bold and underlining indicate the best and second-best scores within each category. A slash denotes an unreported result.

Model	MMBrowseComp	BrowseComp-VL	MMSearch	MMSearch-Plus	VisBrowse
Direct Answer					
GPT-5	<u>10.3</u>	**47.2**	33.3	**19.1**	**26.0**
Gemini-2.5-Flash	5.4	37.1	30.4	8.1	16.0
Gemini-2.5-Pro	<u>10.3</u>	<u>43.1</u>	<u>39.8</u>	<u>14.5</u>	<u>19.5</u>
Claude-4-Sonnet	5.3	29.3	18.7	4.0	8.3
Qwen3-VL-8B	4.9	25.1	15.2	3.2	10.7
Qwen3-VL-32B-Thinking	8.0	30.8	19.8	4.0	14.0
Qwen3-VL-30B-A3B-Thinking	7.1	34.6	17.7	2.7	13.0
Seed-1.8	**13.0**	40.0	31.0	8.6	18.9
Kimi-K2.5	2.7	38.0	**47.0**	12.6	18.3
ReAct Workflows					
GPT-5	9.4	49.1	65.7	34.5	29.0
Gemini-2.5-Flash	10.4	43.1	67.0	23.3	20.7
Gemini-2.5-Pro	14.8	45.2	67.8	30.9	26.6
Claude-4-Sonnet	6.3	44.1	70.3	20.9	19.5
Qwen3-VL-8B	6.7	28.4	53.8	10.9	15.4
Qwen3-VL-32B-Thinking	15.3	35.1	53.9	13.5	20.0
Qwen3-VL-30B-A3B-Thinking	9.8	44.1	62.0	14.4	16.0
Seed-1.8	25.5	<u>54.1</u>	73.2	**46.7**	**58.0**
Kimi-K2.5	<u>25.9</u>	50.3	72.3	<u>39.2</u>	<u>50.3</u>
Gemini-3.1-Pro	**28.5**	**64.1**	**86.1**	38.1	40.2
Claude-4.6-Opus	20.0	48.3	<u>76.2</u>	30.0	47.6
Multimodal Deep Search Agents					
Visual-Seeker-8B	16.1	47.6	72.2	27.3	34.7
Vision-DeepResearch-8B*	–	42.6	69.6	20.4	–
Vision-DeepResearch-30B-A3B*	–	53.7	69.6	28.5	–
POINTS-Seeker-8B*	–	44.4	70.8	25.2	–
MM-DeepResearch-8B*	–	37.9	67.8	–	–
MM-DeepResearch-32B*	–	43.0	69.0	–	–
OpenSearch-VL-8B*	–	37.6	64.5	–	–
OpenSearch-VL-30B-A3B*	–	41.1	68.7	–	–
OpenSearch-VL-32B*	–	43.8	72.3	–	–
REDSearcher-MM-SFT-30B*	25.3	55.3	70.3	20.2	–
REDSearcher-MM-RL-30B*	23.5	57.2	72.9	26.6	–
SearchEyes-9B*	–	42.3	69.2	–	–
SearchEyes-27B*	–	49.3	<u>82.4</u>	–	–
LMM-Searcher-30B*†	30.1	–	72.3	34.8	<u>48.3</u>
VeriSeek-VL-SFT-32B (Ours)	33.9	62.2	80.3	35.9	45.6
VeriSeek-VL-RL-32B (Ours)	<u>34.4</u>	<u>65.9</u>	81.3	<u>40.1</u>	43.8
VeriSeek-VL-FULL-32B (Ours)	**40.1**	**67.6**	**82.7**	**44.3**	**49.7**

* Results reported in the original papers. † LMM-Searcher uses its reported 100-turn setting with context management. Reference papers are listed in Appendix G.

its separately estimated axis, with $\alpha = 0.2$ defining the highest-scoring 20% of calibration steps as the trigger region. GPT-5.6-Luna handles evidence organization, hypothesis extraction, and conflict assessment. Decoding settings are in Appendix C, and controller settings are in Appendix E.

3.2 Main Results

Overall performance. Table 1 shows that VeriSeek-VL achieves the highest accuracy among the compared open-source multimodal deep search agents on all five benchmarks, averaging **56.88**. The average gain over the strongest competing agent on each benchmark is **6.32 percentage points**.

Performance across benchmarks. The largest gains over competing search agents occur on MMBrowseComp, BrowseComp-VL, and MMSearch-Plus: 10.0, 10.4, and 9.5 points, respectively. These results support the framework's effectiveness in multi-step information seeking and cross-modal evidence integration. On MMSearch, FULL scores 82.7, exceeding the strongest competing search agent by 0.3 points and both its SFT and RL configurations.

Effect of model training. Under the same ReAct workflow, SFT improves the base model's average accuracy from 27.56 to 51.58, with gains on all five benchmarks. RL adds 1.52 points, reaching 53.10, for a total gain of 25.54 points over the base model. Improvements are not uniform: MMSearch-Plus and BrowseComp-VL gain 4.2 and 3.7 points, while VisBrowse drops from 45.6 to 43.8.

Effect of inference-time control. With RL weights unchanged, FULL adds HLI-Guard and raises the turn limit from 40 to 100. It improves all five benchmarks by an average of 3.78 points, with the largest gains on MMBrowseComp (+5.7) and VisBrowse (+5.9). The ablations below examine the effects of the additional search budget and triggering strategy.

3.3 Ablation Study of HLI-Guard

Settings. Using the final RL checkpoint, we compare two no-Guard budgets (40 and 100 turns) and three Guard variants with 100 turns on three benchmarks (Table 2). Random Guard triggers independently with probability 0.2 per step; Periodic Guard triggers every five turns; HLI-Guard uses the axis-based trigger. All Guard variants share the same verification, hypothesis tracking, compression, and backtracking procedures.

Table 2: HLI-Guard ablations (accuracy, %). MMS, BC-VL, and MMBC denote MMSearch, BrowseComp-VL, and MMBrowseComp. Avg. is the arithmetic mean across these three benchmarks.

Setting	Max. turns	MMS	BC-VL	MMBC	Avg.
No Guard	40	81.3	65.9	34.4	60.54
No Guard	100	81.0	66.2	34.8	60.66
Random Guard	100	79.3	64.7	31.7	58.56
Periodic Guard	100	82.0	65.4	35.7	61.04
HLI-Guard	100	**82.7**	**67.6**	**40.1**	**63.47**

The first four rows report single-run results, with averages computed from unrounded accuracies. HLI-Guard reuses the four-run mean accuracies from Table 1; its three-benchmark average is computed from the reported scores.

Results. Increasing the no-Guard budget from 40 to 100 turns improves average accuracy by only 0.12 points. At 100 turns, random triggering reduces the average by 2.10 points relative to no Guard, while periodic triggering yields mixed results and a 0.38-point gain. HLI-Guard achieves the highest reported scores on all three benchmarks, averaging 63.47; MMBrowseComp reaches 40.1 versus 35.7 with periodic triggering. These results support using the internal lock-in signal to select when to verify hypotheses.

4 Conclusion

We presented VeriSeek-VL, a multimodal deep search framework that strengthens two abilities required for long-horizon information seeking: seeking the evidence a question implicitly depends on, and verifying intermediate hypotheses before building further search on them. For seeking, we train the model with SFT and GRPO on tasks with implicit visual constraints synthesized from multimodal hypergraphs. For verification, we find that hypothesis lock-in is linearly decodable from model activations, and design HLI-Guard to use the resulting Hypothesis Lock-in Axis to decide when to verify key hypotheses. VeriSeek-VL achieves state-of-the-art performance among open-source methods on five multimodal search benchmarks, surpassing the strongest baseline by 6.32% on average. Training improves the base model by 25.5%, and axis-guided verification adds a further 3.8%. Future work could incorporate the lock-in signal into training, so that agents learn to verify their own hypotheses without inference-time control.

AI Use Statement

LLMs are components of our method, as described in Sections 2 and 3: they synthesize and filter questions, generate teacher trajectories, produce audit labels validated against human annotations, judge answers during RL, and serve as HLI-Guard's auxiliary model and as evaluated baselines. During manuscript preparation, we used LLMs for language polishing and for locating and cross-checking references; every citation was verified against its original source. LLMs did not originate research ideas, claims, or results. The authors take full responsibility for the paper.

Ethics Statement

This work uses only public resources: Wikipedia text, openly licensed Wikimedia Commons images with retained attribution, and public datasets and benchmarks used under their licenses. Some questions require identifying people in public photographs; we restrict them to entities with Wikipedia pages, describe them only through context and visible attributes, and provide no face recognition tools. Like other search agents, VeriSeek-VL may give incorrect answers or be misused to gather information about individuals, so we do not recommend using it to identify private persons or for high-stakes decisions without human oversight.

References

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to Basics: Revisiting REINFORCE Style Optimization for Learning from Human Feedback in LLMs. *arXiv preprint arXiv:2402.14740*, 2024. URL https://arxiv.org/abs/2402.14740.

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. URL https://arxiv.org/abs/2511.21631.

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. *arXiv preprint arXiv:2605.05185*, 2026. URL https://arxiv.org/abs/2605.05185.

Zheng Chu, Xiao Wang, Jack Hong, Huiming Fan, Yuqi Huang, Yue Yang, Guohai Xu, Chenxiao Zhao, Cheng Xiang, Shengchao Hu, Dongdong Kuang, Ming Liu, Bing Qin, and Xing Yu. REDSearcher: A Scalable and Cost-Efficient Framework for Long-Horizon Search Agents. *arXiv preprint arXiv:2602.14234*, 2026. URL https://arxiv.org/abs/2602.14234.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the Frontier with Advanced Reasoning, Multimodality, Long Context, and Next Generation Agentic Capabilities. *arXiv preprint arXiv:2507.06261*, 2025. URL https://arxiv.org/abs/2507.06261.

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. *arXiv preprint arXiv:2604.12890*, 2026. URL https://arxiv.org/abs/2604.12890.

Mingyang Fu, Yuyang Peng, Dongping Chen, Zetong Zhou, Benlin Liu, Yao Wan, Zhou Zhao, Philip S. Yu, and Ranjay Krishna. Seeking and Updating with Live Visual Knowledge. *arXiv preprint arXiv:2504.05288*, 2025. URL https://arxiv.org/abs/2504.05288.

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. *arXiv preprint arXiv:2508.05748*, 2025. URL https://arxiv.org/abs/2508.05748.

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. *arXiv preprint arXiv:2601.22060*, 2026. URL https://arxiv.org/abs/2601.22060.

Dongzhi Jiang, Renrui Zhang, Ziyu Guo, Yanmin Wu, Jiayi Lei, Pengshuo Qiu, Pan Lu, Zehui Chen, Chaoyou Fu, Guanglu Song, Peng Gao, Yu Liu, Chunyuan Li, and Hongsheng Li. MMSearch: Benchmarking the Potential of Large Models as Multi-modal Search Engines. *arXiv preprint arXiv:2409.12959*, 2024. URL https://arxiv.org/abs/2409.12959.

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. *arXiv preprint arXiv:2607.05943*, 2026. URL https://arxiv.org/abs/2607.05943.

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. *arXiv preprint arXiv:2508.13186*, 2025. URL https://arxiv.org/abs/2508.13186v1.

Yikun Liu, Yuan Liu, Haicheng Wang, Zhongyin Zhao, Le Tian, Xiao Zhou, Jiangchao Yao, Yanfeng Wang, and Weidi Xie. POINTS-Seeker: An Open Recipe for Multimodal Search Agents with Visual Memory Management. *arXiv preprint arXiv:2604.14029*, 2026. URL https://arxiv.org/abs/2604.14029.

Christina Lu, Jack Gallagher, Jonathan Michala, Kyle Fish, and Jack Lindsey. The Assistant Axis: Situating and Stabilizing the Default Persona of Language Models. *arXiv preprint arXiv:2601.10387*, 2026. URL https://arxiv.org/abs/2601.10387.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A Flexible and Efficient RLHF Framework. *arXiv preprint arXiv:2409.19256*, 2024. URL https://arxiv.org/abs/2409.19256.

Yuan Sui, Yulin Chen, Yibo Li, Xue Jiang, Yufei He, Yihong Dong, Xiaoxin He, Tianyu Gao, and Bryan Hooi. TACT: Mitigating Overthinking and Overacting in Coding Agents via Activation Steering. *arXiv preprint arXiv:2605.05980*, 2026. URL https://arxiv.org/abs/2605.05980.

Xijia Tao, Yihua Teng, Xinxing Su, Xinyu Fu, Jihao Wu, Chaofan Tao, Ziru Liu, Haoli Bai, Rui Liu, and Lingpeng Kong. MMSearch-Plus: Benchmarking Provenance-Aware Search for Multimodal Browsing Agents. *arXiv preprint arXiv:2508.21475*, 2025. URL https://arxiv.org/abs/2508.21475.

Jinming Wu, Zihao Deng, Wei Li, Yiding Liu, Bo You, Bo Li, Zejun Ma, and Ziwei Liu. MMSearch-R1: Incentivizing LMMs to Search. *arXiv preprint arXiv:2506.20670*, 2025. URL https://arxiv.org/abs/2506.20670.

Huanjin Yao, Qixiang Yin, Min Yang, Ziwang Zhao, Yibo Wang, Haotian Luo, Jingyi Zhang, and Jiaxing Huang. MM-DeepResearch: A Simple and Effective Multimodal Agentic Search Baseline. *arXiv preprint arXiv:2603.01050*, 2026. URL https://arxiv.org/abs/2603.01050.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. *arXiv preprint arXiv:2210.03629*, 2022. URL https://arxiv.org/abs/2210.03629.

Yu Zeng, Wenxuan Huang, Zhen Fang, Shuang Chen, Yufan Shen, Yishuo Cai, Xiaoman Wang, Zhenfei Yin, Lin Chen, Zehui Chen, Shiting Huang, Yiming Zhao, Xu Tang, Yao Hu, Philip Torr, Wanli Ouyang, and Shaosheng Cao. Vision-DeepResearch Benchmark: Rethinking Visual and Textual Search for Multimodal Large Language Models. *arXiv preprint arXiv:2602.02185*, 2026. URL https://arxiv.org/abs/2602.02185.

Zhengbo Zhang, Changtao Miao, Jinbo Su, Zhaowen Zhou, Chunxia Zhang, Xukai Wang, Ruiqi Liu, Kaiyuan Zheng, Jiansheng Cai, Bo Zhang, Zhe Li, Shiming Xiang, and Ying Yan. Visual-Seeker: Towards Visual-Native Multimodal Agentic Search via Active Visual Reasoning. *arXiv preprint arXiv:2606.15231*, 2026a. URL https://arxiv.org/abs/2606.15231.

Zhengbo Zhang, Jinbo Su, Zhaowen Zhou, Changtao Miao, Yuhan Hong, Qimeng Wu, Yumeng Liu, Feier Wu, Yihe Tian, Yuhao Liang, Zitong Shan, Wanke Xia, Yi-Fan Zhang, Bo Zhang, Zhe Li, Shiming Xiang, and Ying Yan. VisBrowse-Bench: Benchmarking Visual-Native Search for Multimodal Browsing Agents. *arXiv preprint arXiv:2603.16289*, 2026b. URL https://arxiv.org/abs/2603.16289.

Yuze Zhao, Jintao Huang, Jinghan Hu, Xingjun Wang, Yunlin Mao, Daoze Zhang, Hong Zhang, Zeyinzi Jiang, Zhikai Wu, Baole Ai, Ang Wang, Wenmeng Zhou, and Yingda Chen. SWIFT:A Scalable lightWeight Infrastructure for Fine-Tuning. *arXiv preprint arXiv:2408.05517*, 2024. URL https://arxiv.org/abs/2408.05517.

A Related Work

A.1 Text-Only Deep Search Agents

Text-only deep search agents obtain external information for complex questions through repeated retrieval, webpage visits, and reasoning on the open web. Search-R1 incorporates search calls into reinforcement learning, enabling models to acquire external knowledge during reasoning. DeepResearcher further trains long-horizon search behavior in real web environments. WebThinker integrates deep web exploration with reasoning to support complex problem solving and research report generation. For training data construction, WebShaper organizes synthesis through a formalization of information-seeking tasks, providing complex tasks for search capability training.

These studies establish foundations for multi-turn evidence acquisition, tool use, and search policy learning. Multimodal deep search additionally requires agents to locate external images, resolve visual entities and their relations, and use visual evidence in subsequent text retrieval and reasoning. We investigate the combination of cross-modal evidence acquisition and intermediate hypothesis verification in this setting.

A.2 Multimodal Deep Search Agents

Multimodal deep search incorporates visual understanding into information seeking on the open web. MMSearch-R1 uses reinforcement learning to train text and image search tool use, while WebWatcher trains multi-turn investigation using visual question answering data and search trajectories. Vision-DeepResearch develops data and training pipelines for fine-grained visual retrieval and long-horizon image–text interaction. Visual-Seeker emphasizes active visual evidence acquisition and uses data synthesis to develop multi-hop cross-modal search. In addition, POINTS-Seeker combines staged training with visual memory management for long-horizon interaction, and OpenSearch-VL provides an open training recipe spanning data construction, supervised fine-tuning, and reinforcement learning.

We share these works' focus on external evidence acquisition and multi-turn reasoning, and further connect visual evidence dependencies in training tasks to hypothesis verification during search. We construct multi-hop tasks with implicit visual constraints to train active retrieval and integration of external evidence. Through audits of complete trajectories, we also identify search-critical hypotheses that remain unverified, extract a corresponding internal direction, and selectively trigger verification and backtracking at runtime. Cross-modal evidence acquisition training and hypothesis verification thus jointly support final task solving.

B Constructing Tasks with Implicit Visual Constraints

B.1 Hypergraph Representation and Source Records

The multimodal hypergraph is $\mathcal{G} = (\mathcal{V}, \mathcal{E}_T, \mathcal{E}_V)$. A node represents a Wikipedia entity and stores its name, textual description, and source page. Single-entity images supplement its visual attributes. A textual edge $e_T = (u, r, v)$ records a relation extracted from the page text. Each original multi-entity image serves as a visual hyperedge $e_V = (V_e, I_e, b_e)$: I_e is the image, V_e contains its associated entities, and b_e contains contextual clues for locating it. We retain image URLs, source pages, and entity–image mappings as metadata. Distinguishing visual descriptions used in question rewriting are extracted from the original image at that stage.

B.2 Seed Selection and Hypergraph Expansion

Seed entities are selected using page text length, the number of related entities, and the number of available multi-entity images. These criteria balance the availability of useful relations and images against the scope of subsequent expansion.

We alternate textual and visual expansion from the seed entities. Textual expansion follows links on entity pages to discover new entities and extract relations. Visual expansion inspects multi-entity images associated with the current entities and adds qualifying images and the entities they connect.

Images must be locatable using contextual clues, and their associated entities must be linkable to Wikipedia pages. Single-entity images supplement node attributes; multi-entity images preserve the visual relations used in question construction.

B.3 Subgraph Sampling and Reference Dependencies

Starting from a root entity, we traverse textual edges and visual hyperedges to sample a connected candidate subgraph $\mathcal{G}_q = (\mathcal{V}_q, \mathcal{E}_{T,q}, \mathcal{E}_{V,q})$, subject to

$$|\mathcal{V}_q| \leq M_{\text{max}}, \qquad \text{depth}(\mathcal{G}_q) \leq D_{\text{max}}, \qquad |\mathcal{E}_{V,q}| \geq N_{\text{min}}. \tag{12}$$

Depth is measured from the root, with each traversal of a textual edge or visual hyperedge counting as one hop. We use $M_{\text{max}} = 50$, $D_{\text{max}} = 7$, and $N_{\text{min}} = 8$. These bounds apply to candidate subgraphs; an individual question need not use every relation.

The generator selects relations that jointly determine an answer and organizes them into an acyclic reference dependency structure. Relations can form sequential dependencies or constraints that must be satisfied jointly. At least one visual hyperedge participates in identifying a key intermediate entity. The generator first determines the reference answer and creates an initial question with explicit entity names. It then replaces starting and intermediate entity names with relational or attribute descriptions in reverse topological order, preserving the answer and reference dependencies.

B.4 Rewriting and Reference Evidence

For each visual hop, the preceding clues establish a reference entity, while the image relation identifies a target entity. The target name is replaced by $c_j = (b_j, \rho_j, f_j)$, combining contextual image-retrieval clues, a reference-entity description, and distinguishing visual information about the target. We retain checkable attributes such as color, shape, texture, patterns, pose, relative position, and interactions, while removing names, aliases, and identifiers that reveal the target identity. The final question is text-only and does not supply the reference image.

Each visual hop produces one implicit visual constraint. For $\mathcal{C}(q) = \{c_j\}_{j=1}^{k(q)}$, we require $1 \leq k(q) \leq n$, with $n = 2$. For each QA pair, the reference record stores the answer, dependency structure, intermediate entities, textual relations, and each visual constraint's reference entity, target entity, contextual clues, visual description, image, and source. These records support complete-trajectory auditing.

C Training Objectives, Data Composition, and Tools

C.1 Data synthesis and trajectory filtering

Following Section 2.1, we construct multimodal hypergraphs and use Gemini 2.5 Pro (Comanici et al., 2025) to synthesize QA pairs with implicit visual constraints. For each candidate subgraph, we set the maximum number of nodes to $M_{\text{max}} = 50$, the maximum expansion depth to $D_{\text{max}} = 7$, and the minimum number of visual hyperedges to $N_{\text{min}} = 8$. We set $n = 2$, allowing at most two implicit visual constraints per question.

To obtain challenging training questions, we discard those that Qwen3-VL-32B-Instruct (Bai et al., 2025) answers correctly without tools. We then collect ReAct trajectories using Qwen3-VL-235B-A22B-Thinking. To train the model to actively retrieve external visual evidence, we retain only trajectories that invoke an image search tool and produce a correct final answer. Retained trajectories must contain 4–40 interaction turns and at most seven image search calls, excluding overly short interactions and excessive image search. The seven-call limit applies to `TextSearchImage` and `ImageSearchImage` combined.

C.2 Tool environment

The agent follows the ReAct interaction protocol, interleaving reasoning, tool calls, and observations, with at most one tool call per turn. Each call is a JSON object containing `name` and `arguments`, enclosed in `<tool_call>` tags. Table 3 lists the available tools. Retrieved pages and images receive `page_id` and `image_id` identifiers for subsequent operations.

Table 3: Tools available to the agent.

Tool	Input	Output
`web_search`	`query`	Up to 10 web search results with titles, snippets, and page IDs.
`deep_search`	`query`, `goal`	Up to eight web search results, with goal-relevant content extracted from each page.
`visit`	`page_id`, `goal`	A goal-focused summary of the specified page.
`TextSearchImage`	`query`	Up to five images with titles and source pages.
`ImageSearchImage`	`image_id`	Up to five visually similar images with titles and source pages.
`image_zoom_in`	`image_id`, `bbox_2d`	A cropped and enlarged region with a new image ID.
`ImageOCR`	`image_id`	Visible image text returned in Markdown format.

C.3 Supervised fine-tuning

We initialize from Qwen3-VL-32B-Thinking and combine trajectories from our QA pairs with implicit visual constraints and those from REDMM, LiveVQA, and FVQA. The filtered SFT corpus contains 14,203 trajectories: 13,919 for training and 284 for validation. Table 4 reports the source composition before this split, alongside the composition of the RL training set.

Table 4: Data composition. SFT counts are trajectories in the filtered corpus before the training–validation split; RL counts are training questions.

Source	SFT count	Share	RL count	Share
QA with implicit visual constraints (Ours)	6,669	46.95%	2,157	32.21%
REDMM (Chu et al., 2026)	2,798	19.70%	890	13.29%
LiveVQA (Fu et al., 2025)	3,051	21.48%	1,869	27.91%
FVQA	1,685	11.86%	445	6.65%
VDRBench (Zeng et al., 2026)	—	—	1,335	19.94%
Total	14,203	100%	6,696	100%

SFT uses Megatron-SWIFT (Zhao et al., 2024) on eight NVIDIA H20 96 GB GPUs. We fine-tune all language model parameters and freeze the vision encoder and visual alignment module. The key settings are: Adam optimizer; learning rate 2×10^{-5}; cosine learning-rate schedule; warmup fraction 0.05; global batch size 128; maximum sequence length 49,152 tokens; and an epoch budget of four. Overlength trajectories are removed (`truncation_strategy=delete`). The loss covers reasoning, tool calls, and final answers across all assistant turns. We use the checkpoint saved at update 320.

C.4 Reinforcement learning

RL starts from the SFT checkpoint and uses 6,696 training questions and 517 validation questions. Training runs on a customized verl (Sheng et al., 2024) implementation with eight NVIDIA RTX PRO 6000 Blackwell Server 96 GB GPUs. We update the language model parameters and freeze the vision encoder. We optimize a GRPO-style clipped policy objective (Shao et al., 2024) with a leave-one-out (RLOO) advantage estimator (Ahmadian et al., 2024). The baseline for each trajectory is the mean reward of the other trajectories for the same question. We use neither within-group standard-deviation normalization nor a separate value network.

Optimization settings are: AdamW optimizer; constant learning rate 10^{-6}; 64 questions per rollout batch; eight trajectories per question (`rollout.n=8`); and 32 question groups per optimization mini-batch (`ppo_mini_batch_size=32`). We make one optimization pass over each rollout batch (`ppo_epochs=1`). The lower and upper clipping parameters are 0.2 and 0.28, giving a policy-ratio

clipping interval of $[0.8, 1.28]$. We use neither a KL regularization loss nor a KL penalty in the reward. The epoch budget is two, and we use the checkpoint saved at training iteration 100.

Rollout settings are: temperature 0.85; top-p $= 1.0$; at most 40 interaction turns; at most 5,120 generated tokens per turn; and a context limit of 32,768 tokens. The initial prompt and subsequent interaction sequence have token budgets of 12,288 and 20,480, respectively. The latter includes both assistant outputs and tool observations.

C.5 Confidence-based routing and rewards

We assign trajectory confidence according to whether the outcome can be reliably attributed to the model's reasoning and action choices. High-confidence trajectories complete tool interactions normally, submit a final answer, and pass reliability checks; their answers may be correct or incorrect. Low-confidence examples include failures caused by environment or tool-service errors and correct answers obtained through degenerate repeated searches.

High-confidence trajectories receive a reward combining answer correctness, format validity, and per-turn response length:

$$R_i = 0.8A_i + 0.1F_i + 0.1L_i, \qquad L_i = 1 - \text{clip}\left(\frac{\ell_i^{\max} - 4096}{1024}, 0, 1\right). \tag{13}$$

Here, A_i is 1 for a correct answer and 0 otherwise, as judged by GPT-5.6-Luna. The format reward F_i is 1 when all unmasked turns have valid tool-call or answer formats and at least one valid turn is present; it is 0.5 for exactly one invalid turn and 0 otherwise. The quantity $\ell_i^{\max}$ is the maximum assistant output length, in tokens, over all turns in the trajectory. The length reward is 1 up to 4,096 tokens and decreases linearly to 0 at 5,120 tokens.

Low-confidence trajectories receive zero total reward and have all tokens masked out of the training loss. They remain in their groups, so their zero rewards still enter the leave-one-out baselines for other trajectories. This prevents reward hacking through format or length bonuses for abnormal early termination or repeated short attempts.

For a group of $G = 8$ trajectories sampled for the same question, the advantage is

$$\widehat{A}_i = R_i - \frac{1}{G-1}\sum_{j \neq i} R_j, \tag{14}$$

where R_j is the reward after confidence-based routing.

If a turn exceeds 5,120 tokens or cannot be parsed as a valid tool call or final answer, we discard that output and resample from the same interaction prefix, with up to three retries. Discarded outputs enter neither the interaction history nor the training trajectory.

C.6 Evaluation and Inference Settings

Evaluation uses temperature 0.4, top-p $= 1.0$, a per-turn generation limit of 5,120 tokens, and a context limit of 49,152 tokens. SFT and RL allow up to 40 interaction turns, while FULL allows 100. HLI-Guard uses the *Hypothesis Lock-in Axis* estimated for the final RL checkpoint and reads the mean activation over thinking tokens at layer 42 for each interaction step. We set $\alpha = 0.2$, defining the trigger region as the highest-scoring 20% of the calibration distribution. GPT-5.6-Luna performs evidence organization, hypothesis extraction, and conflict assessment.

We evaluate on MMBrowseComp (Li et al., 2025), BrowseComp-VL (Geng et al., 2025), MMSearch (Jiang et al., 2024), MMSearch-Plus (Tao et al., 2025), and VisBrowse (Zhang et al., 2026b). Following prior work, we use the single-image subset of MMSearch-Plus and the full public test sets of the other benchmarks. Answer correctness is assessed using each benchmark's official evaluation protocol. For our models in the main comparison, we run each question independently four times and report mean accuracy. Timeouts and runs without a submitted answer count as incorrect. The overall score is the arithmetic mean across the five benchmarks.

D Complete-Trajectory Auditing and Axis Estimation

D.1 Trajectory Sampling and Validity Filtering

We select 1,000 synthetic questions and independently sample five trajectories per question from each of Qwen3-VL-32B-Instruct and Qwen3-VL-32B-Thinking, yielding 5,000 raw trajectories per model. Sampling uses the same system prompt, temperature 0.6, top-p 0.95, and a maximum of 40 interaction turns.

We retain only complete trajectories that normally submit an answer through the final-answer action within the turn limit. Trajectories that reach the limit without an answer, violate the tool protocol, or contain incomplete interaction records are removed before annotation.

D.2 Annotation Inputs and Audit Procedure

The annotation model is GPT-5.4-mini. Its inputs include the question, the complete interaction trajectory, and reference evidence

$$\mathcal{R}_x = \{(v_k, c_k, I_k)\}_{k=1}^{K_x},$$

where v_k, c_k, and I_k are the intermediate entity, constraint relation, and reference image associated with an implicit visual constraint. Reference evidence helps establish which entity bindings and relations require verification. Whether verification is complete is determined from the external evidence actually obtained in the trajectory.

The audit identifies search-critical hypotheses, locates their first occurrence, examines supporting or refuting evidence across the complete trajectory, and maps labels back to steps. It follows Section 2.2: both support and refutation count as completed verification, while insufficient evidence or unresolved necessary disambiguation counts as incomplete verification. Unverified hypotheses are traced to their first occurrence; a step is positive if it first introduces at least one such hypothesis.

The following table illustrates only the labeling logic:

Verification over the complete trajectory	Contribution to the label of the first-occurrence step
Sufficient supporting evidence is later obtained	This hypothesis does not make the step positive.
Sufficient refuting evidence is later obtained, even if the initial conjecture was wrong	This hypothesis does not make the step positive.
Sufficient evidence or necessary disambiguation is still missing at the end	This hypothesis makes the step positive.
The hypothesis is deferred, omitted, or abandoned without sufficient verification evidence	This hypothesis makes the step positive.

The final label of a step depends jointly on all search-critical hypotheses first introduced in that step. A trajectory may contain multiple positives. Automatic annotation quality is evaluated against an independent set of approximately 300 human-annotated trajectories, using the same criteria for human and automatic annotation.

D.3 Activation Readout, Data Splits, and Probes

Both analysis models contain 64 Transformer layers with hidden dimension 5,120. Step representations average residual-stream activations over the thinking tokens of each agent output step, as defined in the main text. The Thinking five-fold evaluation is specified in Appendix F. The additional Instruct probe analysis splits complete trajectories into 80% fitting and 20% held-out data using seed 42.

The layerwise logistic regression probe is

$$\widehat{p}_\ell \left(y_{\tau,t} = 1 \mid \mathbf{h}_{\tau,t}^{(\ell)} \right) = \sigma\left(\mathbf{w}_\ell^\top \mathbf{h}_{\tau,t}^{(\ell)} + b_\ell \right).$$

L1 and L2 probes use the same labels and split, the `liblinear` solver, and regularization parameter $C = 1.0$, differing only in the penalty. Probes are trained on the fitting set and evaluated by ROC-AUC on held-out trajectories.

D.4 Class Means and Checkpoint-Specific Estimation

Let $\mathcal{D}^1_{\text{fit}}$ and $\mathcal{D}^0_{\text{fit}}$ be the two step classes in the fitting set. Their layerwise mean activations are

$$\boldsymbol{\mu}_c^{(\ell)} = \frac{1}{|\mathcal{D}^c_{\text{fit}}|} \sum_{(\tau,t)\in\mathcal{D}^c_{\text{fit}}} \mathbf{h}^{(\ell)}_{\tau,t}, \qquad c \in \{0,1\}.$$

Following the main text, we normalize the difference between positive and negative class means and compute ROC-AUC using projections of held-out steps. This direction is estimated directly from class means; logistic regression parameters do not enter the online projection.

We independently analyze trajectories and estimate directions for the original Instruct, original Thinking, SFT, and GRPO checkpoints, calibrating the readout layer and trigger threshold for each. Offline auditing may access complete trajectories, whereas runtime readout uses only the current step's activations and information already available.

E HLI-Guard States and Control Procedure

E.1 Readout Layer and Threshold Calibration

For the final RL checkpoint, HLI-Guard uses the separately estimated axis at layer 42. We project the mean thinking-token activations of calibration steps onto this axis and set $\gamma_\alpha = Q_{1-\alpha}$ with $\alpha = 0.2$, selecting the highest-scoring 20% of that distribution. GPT-5.6-Luna performs evidence organization, hypothesis extraction, and conflict assessment. The resulting calibration region does not prescribe a fixed number of triggers during an individual search.

E.2 Pending Hypotheses and Lazy Updates

The controller maintains

$$\mathcal{U}_t = \{(h, t_0(h))\}, \tag{15}$$

recording each pending search-critical hypothesis and its first-occurrence step. It updates these records only when the lock-in score triggers HLI-Guard. Newly acquired evidence is used to reassess existing hypotheses; sufficiently supported or refuted hypotheses leave the pending set, and newly identified unverified hypotheses are added. Verification outcomes, excluded candidates, and their supporting sources remain available after a hypothesis leaves the set. A refuted candidate may still leave an entity slot unresolved and require alternatives.

E.3 Candidate Slots and Stored States

For each intermediate entity, relation, or attribute slot u_j, the controller retains a candidate set $\mathcal{H}_j$. A triggering step can create a node $v_t = (t, \mathcal{E}_t, \mathcal{S}_t)$, where $\mathcal{E}_t$ preserves the available evidence and $\mathcal{S}_t = \{(u_j, \mathcal{H}_j)\}$ records the slots and candidates. Nodes are stored in trigger order and updated only at subsequent triggers.

New relevant observations update candidates as supported, refuted, or pending. Supported candidates can resolve their slots; refuted candidates are excluded, but their slots may still require alternatives. Nodes whose verification is complete and that require no further processing are removed. Other nodes retain their evidence and candidate states for possible backtracking.

E.4 Tool-Observation Intervention

After the agent generates a step, HLI-Guard reads its mean thinking-token activation and computes the axis score. When the score reaches the calibrated threshold, an auxiliary model organizes available evidence and identifies key hypotheses requiring verification. The tool call executes normally. HLI-Guard prepends a verification prompt to the returned tool content, guiding subsequent search toward external evidence that supports or refutes these hypotheses. The controller uses only evidence available during the interaction.

E.5 Compression Boundary and Backtracking

For nonempty $\mathcal{U}_t$, let $b_t = \min_{(h,j)\in\mathcal{U}_t} j$. The controller summarizes the history strictly before b_t, preserving relevant facts, verification outcomes, and source identifiers. The interaction from b_t onward remains in full. Subsequent compression advances only when earlier pending hypotheses are resolved and the boundary moves forward; a new trigger alone does not require further compression.

When new evidence refutes a hypothesis underlying the current search, its candidate is excluded. The controller reconstructs context from the evidence and hypothesis state at the corresponding step, incorporates later relevant evidence, and directs search toward alternative candidates. If several hypotheses are affected, it selects the earliest affected step. Excluded candidates remain recorded to avoid reusing a refuted premise. If all candidates for a slot are excluded, the controller generates alternatives for the same slot. It backtracks further to an earlier unresolved node only when no viable candidate can be formed.

E.6 Control Cycle

1. Generate the next search step and compute its mean thinking-token activation and axis projection s_t.
2. If $s_t < \gamma_\alpha$, continue the original search without updating the pending-hypothesis records.
3. Otherwise, assess new evidence against existing hypotheses, record support or refutation, and extract newly proposed unverified hypotheses with their first-occurrence steps.
4. Execute the step's tool call normally and prepend the verification prompt to its observation.
5. When pending hypotheses remain, retain the full interaction from the earliest still-pending hypothesis onward. Summarize only the preceding history, and extend the summary only if the boundary advances.
6. If evidence refutes a premise of the current search, exclude that candidate and reconstruct context at the earliest affected step to explore alternatives.
7. Continue search with the updated evidence summary, unresolved history, hypothesis records, and excluded candidates.

F Supplementary Analysis of the Hypothesis Lock-In Axis

F.1 Analysis Samples and Evaluation Splits

For the base Qwen3-VL-32B-Thinking analysis in Figure 4, we select 1,000 questions from quality-filtered synthetic tasks and sample five trajectories per question, yielding 5,000 raw trajectories. Sampling uses temperature 0.6, top-p 0.95, and at most 40 interaction turns. We retain trajectories that submit a final answer, follow the tool protocol, and have complete records.

For the layer-wise evaluation, we divide the analysis data into five folds. Each fold serves once as the held-out set, while the other four estimate the class means and the normalized mean-difference axis. We compute ROC-AUC from the held-out projection scores and report the mean and sample standard deviation across folds. Layer 42 achieves 0.9219 ± 0.0134. The layer-42 distribution panel uses held-out projections, with a separately normalized Gaussian kernel density estimate for each class and Scott's bandwidth rule.

F.2 Distribution of Lock-In Onset Steps

Figure 4c groups the base Thinking trajectories by their number of lock-in onset steps. The accuracies for 0, 1, and 2+ steps are $593/2068 = 28.68\%$, $220/1206 = 18.24\%$, and $117/958 = 12.21\%$, respectively. The last group combines all trajectories with at least two onset steps.

F.3 Additional Instruct Analysis

For the additional Qwen3-VL-32B-Instruct analysis, complete trajectories are split into 80% fitting and 20% test data using seed 42. L1 and L2 logistic regression probes use `liblinear` with $C = 1.0$.

The highest probe ROC-AUC is 0.9138, while the mean-difference axes obtain approximately 0.86–0.88. These results are separate from the Thinking five-fold evaluation above.

F.4 Random Grouping and Role Prompt Variation

For the additional Instruct robustness analysis, we retain the data split and class counts but randomly assign steps to two pseudo-classes, then estimate a difference-of-means direction and evaluate held-out AUC with the same procedure. For role prompts, we use 12 prompts across three categories—searcher-like, default, and answer-like—and independently collect trajectories, audit labels, and estimate directions for each prompt.

Table F1: Direction discrimination and cross-prompt similarity reported in existing analyses.

Analysis	Metric	Result
Difference-of-means direction from behavioral labels	Held-out ROC-AUC	Approx. 0.86–0.88
Difference-of-means direction from random pseudo-classes	Held-out ROC-AUC	0.51–0.56
Directions independently estimated within the same role category	Mean cosine similarity at the same layer	Approx. 0.88–0.93
Directions independently estimated across role categories	Mean cosine similarity at the same layer	Approx. 0.81–0.87

The direction associated with behavioral labels discriminates held-out steps better than random grouping. Independently estimated directions retain high positive cosine similarity under changes in role prompts, suggesting that this activation structure is reasonably stable under the prompts examined. The ranges summarize existing analyses and are not confidence intervals.

F.5 Checkpoints and Direction Estimation

Directions for the original Instruct, original Thinking, SFT, and GRPO checkpoints are estimated independently from their own trajectories, with separate readout layers and thresholds. The analysis examines whether behavioral discrimination can be reproduced at different checkpoints; directions are not directly shared across checkpoints. The layer-42 peak in Figure 4 characterizes the base Thinking checkpoint; the final RL system uses its own axis.

G Sources and Aggregation of Main Results

G.1 Results for External Methods

Table G1 lists reference papers and configurations for the external methods in the main comparison. Results marked with an asterisk in Table 1 are taken from the corresponding papers.

Table G1: Sources and settings for external results.

Method	Source	Setting and notes
Visual-Seeker-8B	Visual-Seeker (Zhang et al., 2026a)	8B model; all five benchmark results are available.
Vision-DeepResearch-8B / 30B-A3B	Vision-DeepResearch (Huang et al., 2026)	Results for the two model sizes are retained separately.
POINTS-Seeker-8B	POINTS-Seeker (Liu et al., 2026)	The MMSearch-Plus score of 25.2 also comes from the original paper.
MM-DeepResearch-8B / 32B	MM-DeepResearch (Yao et al., 2026)	Results for the two model sizes are retained separately.
OpenSearch-VL-8B / 30B-A3B / 32B	OpenSearch-VL (Chen et al., 2026)	Results for the three model sizes are retained separately.

Method	Source	Setting and notes
REDSearcher-MM-SFT / RL-30B	REDSearcher (Chu et al., 2026)	Multimodal SFT and RL checkpoints.
SearchEyes-9B / 27B	SearchEyes (Jiao et al., 2026)	Results for the two model sizes are retained separately.
LMM-Searcher-30B†	LMM-Searcher (Du et al., 2026)	100-turn version with context management, retaining the results of the setting marked with an asterisk in the original table.

G.2 Average Scores and Gains

Let $s_1, \ldots, s_5$ denote the five benchmark scores. We use

$$\bar{s} = \frac{1}{5} \sum_{i=1}^{5} s_i.$$

The score sums for SFT, SFT + GRPO, and FULL are 257.9, 265.5, and 284.4, yielding averages of 51.58, 53.10, and 56.88. Average gains between consecutive stages are 1.52 and 3.78 percentage points, and FULL improves over SFT by 5.30 points. This calculation neither treats missing results as zero nor mixes averages computed over different numbers of benchmarks. The strongest reported external scores on the five benchmarks are 30.1, 57.2, 82.4, 34.8, and 48.3, respectively, averaging 50.56. Hence, the average improvement over these benchmark-specific best results is 56.88 − 50.56 = 6.32 points. This aggregate is not the score of a single external model.

The grouped accuracy reported in Section 2.2 is calculated by dividing the number of correct trajectories by the number of trajectories in each group, then rounding to two decimal places. There are 2,164 trajectories with at least one lock-in onset step, of which 337 are correct, giving $337/2164 = 15.57\%$ accuracy. These proportions describe observations in the current analysis set.