Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

From Understanding to Action: A Survey on Closed-Loop Multimodal Agents

Yi Wei
University of Auckland
Yan Gao
University of Cambridge
Junyan Wang
Adelaide University
Hong Jia
hong.jia@auckland.ac.nz

University of Auckland

Research Article

Keywords: Multimodal agents, closed-loop interaction, evidence-centered agency, tool use, agent evaluation

Posted Date: October 16th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11179875/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

From Understanding to Action: A Survey on Closed-Loop Multimodal Agents

Yi Wei[1], Yan Gao[2], Junyan Wang[3], Hong Jia[1*]

[1]University of Auckland, Auckland, New Zealand.
[2]University of Cambridge, Cambridge, United Kingdom.
[3]Adelaide University, Adelaide, Australia.

*Corresponding author(s). E-mail(s): hong.jia@auckland.ac.nz;
Contributing authors: ywei598@aucklanduni.ac.nz; yg381@cam.ac.uk;
junyan.wang@adelaide.edu.au;

Abstract

Multimodal systems are increasingly moving from understanding-centered models to agents that inspect observations, call tools, interact with digital or physical environments, and update behavior from feedback. Existing surveys have covered multimodal large language models, tool use, GUI and web agents, embodied agents, and agent evaluation, but these lines of work are often organized by capability, modality, task domain, or system architecture. This survey offers an evidence-centered account of closed-loop multimodal agency. We define evidence as an inspectable intermediate state that connects multimodal observation to downstream decision, action, or revision. Based on this definition, we organize 105 reviewed papers around five evidence operations: active perception, cognitive workspace, knowledge-tool expansion, environment control, and learning from feedback. We further introduce the O-P-G-C evaluation lens, which separates outcome, process, grounding, and cost-risk signals. This lens clarifies why task success alone is insufficient for evaluating multimodal agents: a system may complete a task while using weak evidence, following an invalid trajectory, or incurring unnecessary cost and risk. Our analysis shows that current benchmark protocols cover outcome most consistently, while grounding, cost-risk, and failure diagnosis remain less uniformly supported. We conclude by identifying future research directions in evidence trace reporting, generalization across inputs and environments, feedback learning without error amplification, and cost-risk-aware deployment.

Keywords: Multimodal agents, closed-loop interaction, evidence-centered agency, tool use, agent evaluation

1 Introduction

Multimodal systems are expanding from understanding-centered models to agentic systems that act during task execution. Surveys of MLLMs often organize this progress around visual-language understanding, reasoning, and generation (Yin et al. 2024). Surveys of large multimodal agents further discuss systems that combine multimodal perception with planning, memory, tool use, environment interaction, action, and evaluation (Xie et al. 2025a). This shift is visible in web and desktop agents, mobile agents, and embodied or robotic systems. In these tasks, a system must decide what to observe, what evidence to seek, which operation to execute, and how to use feedback.

These systems require an evidence-centered analysis because a final answer or final action does not explain how the agent used task-relevant information. A correct answer may come from proper visual localization, but it may also come from language priors, accidental guessing, or unverified external results. A completed web or robotic task may also hide ineffective search, incorrect tool calls, wrong target binding, or high-risk actions. In this survey, evidence denotes an intermediate state that connects multimodal observation to downstream decision and can be inspected after it is produced. Under this definition, agent success and failure can be traced to different stages of evidence acquisition, evidence organization, external evidence expansion, environment control, or feedback-based update.

Existing surveys provide important foundations for this area, but closed-loop evidence relations still require more systematic organization. Surveys of LLM-based autonomous agents summarize capabilities such as planning, memory, tool use, and action (Wang et al. 2024b). Surveys of tool learning examine how foundation models select and use external tools (Qin et al. 2024). Surveys of embodied AI and VLA models focus on how agents perceive, act, and learn in physical environments (Duan et al. 2022). Recent surveys of agentic MLLMs further connect multimodal understanding with agent behavior (Yao et al. 2025). These works cover many relevant capabilities, but they leave an organizational question: how perception, reasoning, tool use, environment action, feedback learning, and evaluation can be placed within the same inspectable evidence chain.

This survey addresses this problem by treating the closed loop itself as the main unit of analysis. We organize the literature around five evidence operations: active perception acquires task-relevant observations, cognitive workspace structures evidence into reasoning states, knowledge-tool expansion retrieves or verifies external evidence, environment control tests decisions through action and state change, and learning uses feedback to update later behavior. These operations are analytical roles, not an assumption that every system implements five separate modules. They provide a common vocabulary for comparing work on multimodal perception, visual reasoning, retrieval and tool use, web and GUI agents, embodied control, and self-improving agents.

The evaluation discussion in this survey follows the same evidence-centered logic. We distinguish outcome, process, grounding, and cost-risk signals because task success alone cannot explain whether the evidence path is valid, grounded, or safe. This survey makes four contributions:

- We define multimodal agency as a closed-loop evidence process, in which evidence connects multimodal observation, reasoning, environment intervention, and feedback-based update.
- We organize recent work into five evidence operations that connect active perception, cognitive workspace, knowledge-tool expansion, environment control, and learning.
- We introduce the O-P-G-C evaluation lens to compare how benchmarks evaluate outcome, process, grounding, and cost-risk.
- We identify future research directions, including evidence trace reporting, generalization across inputs and environments, feedback learning without error amplification, and cost-risk-aware deployment.

2 Conceptualizing Closed-Loop Multimodal Agency

2.1 Closed-Loop Multimodal Agency

This survey defines multimodal agency as a closed-loop process that links perception, reasoning, intervention, and feedback through evidence. A multimodal agent does not only map visual or multimodal inputs to a response; it also decides what to inspect, what evidence to seek, what action or tool call to make, and how later behavior should change after feedback. Prior surveys describe LLM-based agents as systems that combine reasoning, planning, memory, tool use, and environment interaction (Wang et al. 2024b). Recent surveys on large multimodal agents extend this account to multimodal perception, action, collaboration, and evaluation (Xie et al. 2025a). We build on these accounts, but use the closed loop itself as the main unit of analysis. Under this view, evidence is the operational link that makes an agent's decisions, actions, and revisions inspectable.

In this survey, evidence denotes an intermediate, task-relevant state that connects a multimodal observation to a downstream decision and can be inspected after it is produced. Evidence may be perceptual, textual, procedural, or trajectory-level: a cropped image region, a selected video frame, an OCR span, a GUI element, a retrieved document, a tool output, a predicted state transition, or a failure trace. A system forms a closed-loop evidence process when it constructs such states from observations, uses them to guide reasoning or intervention, observes the resulting state change, and conditions later decisions on feedback. The analytical focus is therefore traceability: which information was selected, how it supported an answer or intervention, and how the outcome shaped subsequent behavior.

Figure 1 presents this definition as the organizing framework of the survey. The framework distinguishes five ways in which evidence is acquired, organized, verified, acted on, and revised: active perception selects task-relevant observations; the evidence workspace structures them for reasoning; knowledge-tool expansion brings in external evidence when the current observation is insufficient; environment control tests decisions through action and state change; and learning incorporates feedback into subsequent decisions. These loops structure Sections 3–7, although they are analytical roles rather than isolated system components. The O-P-G-C evaluation lens inside the loop indicates how Section 8 evaluates loop execution: outcome asks whether

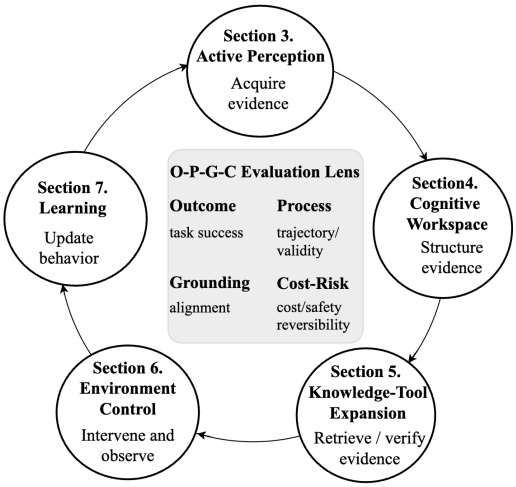


Fig. 1 Closed-loop evidence framework for multimodal agency. The five evidence loops organize Sections 3–7, while the O-P-G-C lens summarizes the evaluation dimensions discussed in Section 8

the task succeeds, process examines whether the trajectory is valid, grounding checks whether the evidence aligns with the target, and cost-risk captures cost, safety, and reversibility concerns. This distinction clarifies two questions that multimodal-agent surveys often treat together: how evidence supports closed-loop agency, and how the resulting loop should be evaluated.

The framework also guides our coding of evidence operations across the surveyed papers. It is not intended to classify systems by architecture. For each paper, the focal loop denotes the evidence operation addressed by the paper's main method, analysis, or evaluation. A supporting loop is recorded when another evidence operation materially affects the task but is not the paper's main object of study. We therefore do not require every included system to implement all five loops; detailed inclusion and coding rules are reported in Section 2.5.

2.2 Survey Landscape and Comparative Positioning

Recent surveys provide the immediate context for this review, but they organize multimodal agency through different analytical units. Model-centered MLLM surveys usually begin from architectures, training data, benchmarks, and general capabilities. Yin et al. (2024) organize MLLM research around model architectures, training strategies, and evaluation settings. Caffagni et al. (2024) review the development of MLLM architectures and benchmarked abilities. Agent-centered surveys move the focus from model capability to perception, planning, memory, tool use, action, and evaluation. Xie et al. (2025a) survey large multimodal agents through agent modules, applications, and evaluation settings. Yao et al. (2025) focus on the agentic capability of MLLMs, including tool use and interactive execution. Domain-specific surveys then examine narrower settings. Nguyen et al. (2025) review GUI agents as interface-grounded systems. Ma et al. (2026) survey vision-language-action models for embodied

AI. Kawaharazuka et al. (2025) discuss VLA models in robotics deployment. Shang et al. (2026) review embodied world models and predictive state modeling. Zhang et al. (2026) survey agentic reinforcement learning and feedback-driven optimization. These surveys are complementary, but their organizing categories can separate systems that share the same closed-loop reliability problem (Table 1).

The difference is therefore not coverage alone, but the unit of analysis. Existing surveys explain what components multimodal agents contain, which domains they operate in, or which capabilities they exhibit. This survey instead asks how those components form an auditable loop: observations are converted into evidence, evidence supports reasoning or intervention, intervention changes an information state or environment, and feedback constrains later behavior. Under this view, web agents, GUI agents, OS agents, tool-augmented agents, embodied agents, and self-improving agents can be compared through a common question: which part of the observation–evidence–intervention–feedback loop is closed, and where does the loop remain brittle, unaudited, costly, or unsafe?

2.3 Five Evidence Loops

The five loops in Figure 1 describe recurring evidence operations in multimodal agency. They specify how evidence is acquired, organized, checked, used to affect an environment, and revised after feedback. These loops are analytical roles: they may appear as separate modules, distributed procedures, prompt-level routines, tool calls, memory updates, or evaluation traces. The same loop categories can be applied across systems with different architectures, because they describe the functional role of evidence rather than implementation form.

Active perception is the loop in which an agent selects or produces observations for a task, following the view that perception can be an active, task-conditioned process (Bajcsy 1988). In multimodal systems, this may include choosing image regions, selecting video frames, querying a screen element, requesting another view, or deciding which sensory signal should be processed next. The key feature is that perception is treated as an action conditioned on the task state, not as a fixed input stage. Evidence from this loop supports later reasoning because it determines what information enters the agentic process.

The cognitive workspace loop organizes evidence into states that can support reasoning. We use the term to refer to an evidence workspace, not to a specific cognitive architecture. Examples include structured scene descriptions, intermediate rationales, plans, symbolic states, object lists, tables, or task memories. Chain-of-thought prompting provides one common example of exposing intermediate reasoning traces for analysis (Wei et al. 2022). This loop matters because multimodal agents often fail not only when evidence is missing, but also when available evidence is poorly represented or incorrectly connected to the decision.

Knowledge-tool expansion retrieves or verifies evidence through external sources such as databases, documents, search interfaces, APIs, calculators, or domain tools. ReAct is an early example of interleaving reasoning traces with actions that query external knowledge sources or environments (Yao et al. 2023). Environment control uses evidence to choose actions that change a digital or physical state, and it observes

Table 1 Positioning of this survey relative to related surveys. Section references indicate where each survey line connects most directly to this paper; they are not exclusive coverage claims

Survey	Primary scope	Main unit of analysis	Closest connection in this survey	Relation to this survey
Yin et al. (2024)	General MLLMs	Model components and capabilities	Sections 3–4: multimodal perception, reasoning, and evidence workspaces	Establishes MLLM foundations; this survey focuses on closed-loop interaction and traceable evidence.
Caffagni et al. (2024)	General MLLMs	Architectures, training, and tasks	Sections 3–4: multimodal inputs, representations, and reasoning abilities	Reviews capabilities that become inputs to agentic systems; this survey centers intervention and feedback.
Xie et al. (2025a)	Large multimodal agents	Agent modules and applications	Sections 3–6 and Section 8: perception, workspace, tool use, action, and evaluation	Covers major agent components; this survey re-indexes them by evidence flow and auditability.
Yao et al. (2025)	Agentic MLLMs	Agent capabilities and interaction modes	Sections 5–7: tool use, environment interaction, and feedback-driven improvement	Emphasizes agentic behaviors; this survey asks how those behaviors close or fail to close an evidence loop.
(Su et al. 2025)	Visual reasoning	Intermediate visual artifacts	Section 4 and Section 8: visual evidence states, process evaluation, and grounding	Clarifies visual intermediate states; this survey connects such states to retrieval, action, and feedback.
Nguyen et al. (2025)	GUI agents	GUI tasks, grounding, and actions	Sections 3, 6, and 8: GUI perception, executable control, and evaluation	Gives a domain-specific account of GUI control; this survey compares GUI control with web, OS, and embodied settings.
Ma et al. (2026)	Vision-language-action systems	Model-action coupling	Sections 6–8: embodied control, feedback, grounding, and cost-risk	Focuses on embodied action models; this survey treats VLA control as one instance of environment intervention.
Kawaharazuka et al. (2025)	Robotics	VLA policies and deployment	Sections 6–8: robotic state change, feedback, and deployment risk	Highlights deployment constraints; this survey links those constraints to cross-domain cost-risk and feedback analysis.
Shang et al. (2026)	Embodied world models	Predicted states and model-based control	Sections 6–7: state prediction, action verification, and feedback	Reviews predictive state modeling; this survey treats prediction as evidence for intervention and revision.
Zhang et al. (2026)	Learning agents	Feedback trajectories and optimization	Section 7 and Section 8: learning signals, process evaluation, and cost-risk	Reviews agentic reinforcement learning; this survey connects learning signals to cross-loop failure propagation.
This survey	Multimodal agents	Closed-loop evidence process	Sections 3–8: evidence acquisition, structuring, expansion, intervention, feedback, and evaluation	Organizes multimodal agents by how evidence is acquired, structured, expanded, acted on, revised, and audited.

the outcome of those actions. WebArena illustrates this setting through web tasks whose success depends on executable interactions with a realistic browser environment (Zhou et al. 2024). Learning closes the longer horizon of the loop by incorporating feedback, errors, or performance signals into later decisions. Reflexion provides a representative mechanism in which task feedback is converted into verbal memory for later trials (Shinn et al. 2023). These three loops mark the point at which multimodal agency moves from interpreting evidence to testing, correcting, and reusing it.

2.4 Evaluation Dimensions: Outcome, Process, Grounding, and Cost-Risk

Closed-loop multimodal agency requires evaluation beyond final task success. A system may produce the correct answer while using weak evidence, taking unnecessary actions, or following an unstable trajectory. AgentBench similarly motivates evaluation in interactive environments, where reasoning and decision-making failures are not captured by static output accuracy alone (Liu et al. 2024b). We use O-P-G-C as a compact evaluation lens for this purpose: Outcome, Process, Grounding, and Cost-Risk. These dimensions are not separate benchmarks; they specify what an evaluation should observe when a system executes an evidence loop.

Outcome measures whether the system completes the target task under the specified protocol. Depending on the domain, this may be reported as accuracy, success rate, reward, completion time, or a human or automatic task judgment. WebArena uses execution-based checks to evaluate whether web agents complete tasks in realistic browser environments (Zhou et al. 2024). Process examines the trajectory that leads to the outcome. It records whether intermediate observations, reasoning states, tool calls, and actions are coherent and consistent with task constraints.

Grounding measures the alignment between the selected evidence and the target decision or action. It asks whether the cited region, frame, text span, retrieved source, or observed state change supports the system's claim or intervention. VisualWebArena makes this issue explicit by evaluating multimodal web agents on visually grounded tasks that require image-text interpretation and website actions (Koh et al. 2024). Cost-Risk records the resources and risks involved in closing the loop, including computation, tool use, time, intervention cost, safety risk, and reversibility. This dimension becomes more visible when agents interact with GUIs, web environments, robots, or physical systems where actions can change external state. OSWorld further illustrates this setting in real computer environments, where agents act across desktop and web applications under execution-based evaluation (Xie et al. 2024).

These dimensions are complementary and may disagree. A system can achieve a high outcome score while relying on weak grounding, or follow a valid process while incurring high cost or irreversible actions. For this reason, O-P-G-C is used as a diagnostic lens rather than a single aggregate metric. It makes evaluation gaps visible by separating what succeeds, how it succeeds, what evidence supports it, and what cost or risk is introduced.

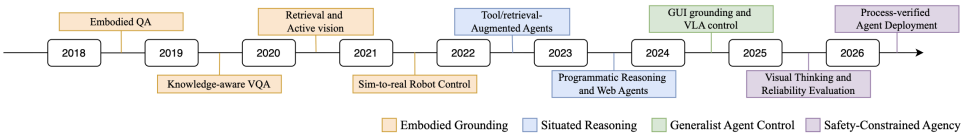


Fig. 2 Representative themes in the screened corpus from 2018 to 2026. The timeline marks the main themes used to organize the reviewed papers; it is not intended as an exhaustive summary of all multimodal-agent research in each year

2.5 Literature Collection and Coding Protocol

Our review process follows the reporting logic of systematic literature reviews in software engineering (Kitchenham et al. 2009). We built the corpus through literature search, relevance screening, and evidence-loop coding. The search targeted papers in which multimodal observation is connected to reasoning, tool use, action, feedback, or evaluation.

The search covered papers published or released between 2018 and 2026. This period captures the transition from embodied question answering and knowledge-aware VQA to retrieval-augmented agents, GUI and web control, visual thinking, and process-oriented evaluation. Figure 2 organizes the reviewed papers by year and marks the main themes used to group the corpus. These labels are representative of the corpus, not exhaustive descriptions of each year.

We searched ACM Digital Library, IEEE Xplore, arXiv, ACL Anthology, CVF Open Access, OpenReview, Google Scholar, and Semantic Scholar. Search terms covered multimodal agents, MLLM agents, vision-language agents, embodied agents, GUI agents, web agents, tool use, retrieval-augmented agents, active perception, visual grounding, reflection, self-improvement, and closed-loop evaluation. We also traced references forward and backward from representative papers and recent surveys, following Wohlin (2014).

Across these sources, we inspected 320 title and abstract records from keyword-query families and snowballing results. The search was bounded to papers published or released between 2018 and 2026 in computer science and artificial intelligence venues or repositories. Initial relevance screening retained 149 records for full-text assessment and excluded 171 records that were generic or static, text-only, non-agentic, or outside the scope of the survey. Full-text review then excluded 44 reports that were outside the closed-loop scope or did not provide enough analyzable detail. The final evidence-operation-coded corpus contains 105 studies: 18 on perception and observation selection, 21 on evidence workspace construction, 25 on knowledge and tool expansion, 31 on environment intervention, and 10 on feedback-driven learning. Figure 3 summarizes the path from search records to the final corpus.

For each included paper, we recorded its task domain, modality, environment type, model backbone, tool or retrieval interface, action space, evidence type, feedback source, evaluation setting, and reported metrics. We assigned a focal evidence operation according to the paper's main contribution and evaluation target, and marked other task-relevant operations as supporting operations. The coding was checked in

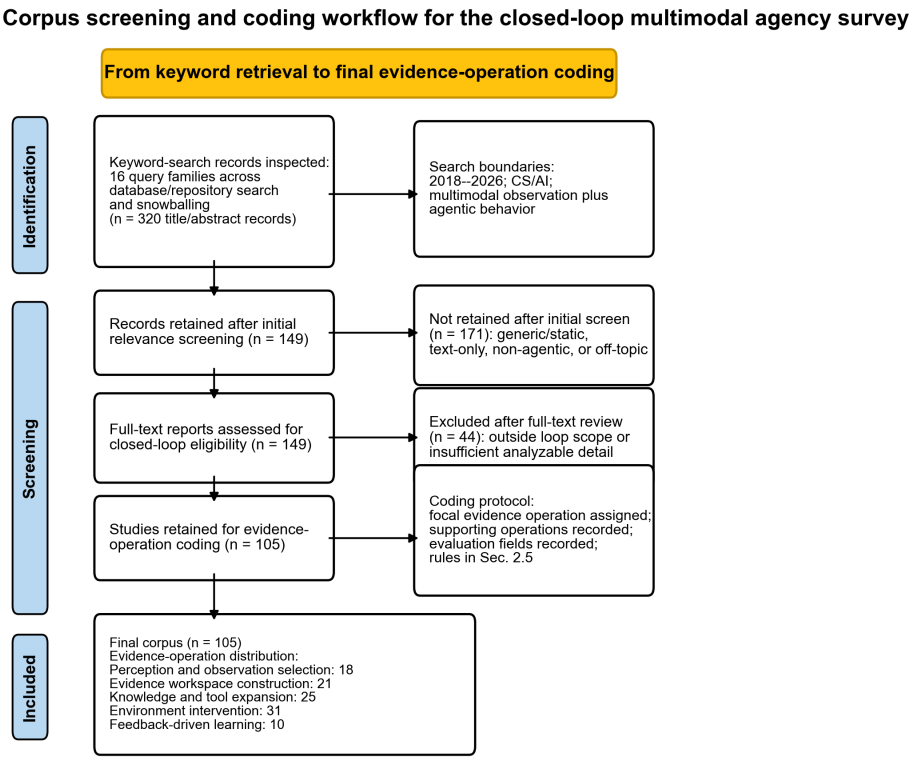


Fig. 3 Selection and coding flow for the reviewed corpus. The diagram follows Page et al.'s PRISMA 2020 reporting logic and summarizes how 320 title and abstract records were screened to a final evidence-operation-coded corpus of 105 studies (Page et al. 2021)

a second pass across papers with similar tasks or mechanisms. We also recoded 25 sampled papers, five from each focal evidence operation, without consulting the first-pass labels. The resulting coding table provides the basis for the evidence-operation analysis in Sections 3–7 and the evaluation discussion in Section 8.

3 Active Perception: Acquiring Grounded Evidence

3.1 Input Spaces and Partial Observability

Active perception becomes necessary when a multimodal agent cannot assume that the initial observation contains all task-relevant evidence (Bajcsy 1988). In the reviewed corpus, this problem appears across visual, temporal, interface-based, and embodied settings. The agent may need to inspect a small region, read a dense layout, select a key frame, query an interface state, or change its viewpoint before reasoning or action can proceed. We use input space to refer to the form in which observations become available to the agent, including pixels, layout structures, temporal streams, interface states, and sensorimotor signals.

Static visual inputs include scene images, high-resolution images, diagrams, charts, and maps. Their main bottleneck is spatial selectivity: task-relevant evidence may be small, visually subtle, or distributed across regions (Tong et al. 2024). Text-rich and layout inputs add another constraint, because documents, tables, forms, screenshots, and OCR-heavy pages require the agent to preserve both textual content and spatial organization (Baechler et al. 2024). In these cases, perception errors often arise from selecting the wrong region, missing small text, or losing relations among layout elements.

Temporal and multimodal streams introduce a different form of partial observability. In video and temporal inputs, evidence depends on when the agent observes as much as what it observes (Wang et al. 2025b). The relevant signal may be a frame, clip, event boundary, spoken cue, sound event, or synchronized audio-video pattern. This setting makes active perception a temporal selection problem, since the agent must decide which moments to inspect and whether later observations revise earlier evidence.

GUI, web, and mobile environments combine visual observation with structured interface state (Cheng et al. 2024). Screenshots, DOM or HTML nodes, accessibility trees, UI elements, and page states may describe the same interface at different levels of abstraction. Active perception in this setting therefore includes deciding which representation to trust, which element to inspect, and whether an interface state has changed after an action. Embodied and robotic environments add sensorimotor constraints, where RGB-D inputs, multi-view images, point clouds, camera poses, proprioception, and tactile or force cues are tied to viewpoint, motion, and physical contact (Park et al. 2023).

Figure 4 summarizes these input spaces as a corpus-grounded map for the rest of this chapter. The categories are not mutually exclusive: for example, a screenshot can be both a text-rich layout and an interactive interface state. Therefore, the figure is not intended as an exhaustive taxonomy of all sensing modalities. Its purpose is to show why active perception cannot be reduced to passive visual encoding: different input spaces require different evidence-selection decisions before the agent can reason, verify, act, or learn from feedback.

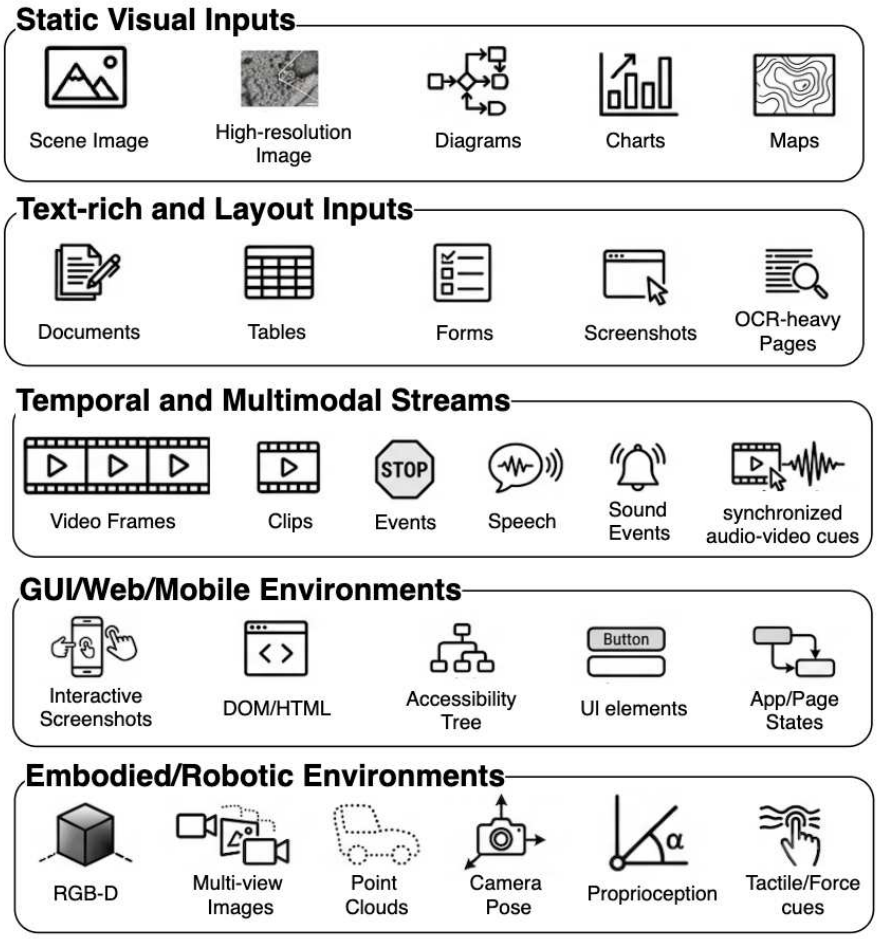


Fig. 4 Input spaces for active perception in multimodal agents. The categories summarize common input spaces in the reviewed corpus and are not mutually exclusive

3.2 Active Observation Policies: Search, Zoom, and Re-Inspection

Active observation policies specify how an agent chooses the next observation before committing to a decision or action. We use policy in a broad sense: it may be a learned controller, a prompted routine, a sequence of tool calls, or a parser-mediated procedure. In the reviewed corpus, these policies address a recurring bottleneck: task-relevant evidence is often present but not immediately accessible from the initial observation. Guided visual search is one representative formulation, where the model directs attention toward regions likely to contain the needed evidence (Wu and Xie 2024).

For static visual inputs, active observation often takes the form of spatial search, cropping, and zooming. These operations are used when the relevant object, relation, text, or visual pattern occupies only a small part of the image. Adaptive chain-of-focus

methods make this process explicit by selecting regions and refining visual evidence through dynamic search and zooming (Zhang et al. 2025b). Similar high-resolution strategies decompose an image into local evidence units so that the model does not rely on a single global encoding (Wang et al. 2025a).

In GUI, web, and mobile environments, active observation must also choose among multiple interface representations. A system may inspect pixels, OCR text, DOM or HTML nodes, accessibility trees, or detected UI elements before selecting an action. SeeClick shows the importance of screenshot-based grounding when GUI agents cannot rely on platform-specific interface metadata (Cheng et al. 2024). Mobile-Agent further illustrates this setting by using visual perception tools to identify and localize mobile interface elements before operation (Wang et al. 2024a). Parser-based methods such as OmniParser make the observation step more explicit by extracting structured GUI elements from screenshots (Lu et al. 2024).

Temporal inputs require policies that decide which moments to inspect. Long videos contain redundant frames, sparse events, and query-dependent evidence, so uniform processing can waste context and miss key transitions. VideoAgent addresses this problem by selecting and accumulating evidence from sampled video observations (Wang et al. 2024c). VideoTree represents another strategy, organizing long videos into adaptive tree structures for coarse-to-fine reasoning (Wang et al. 2025b). For embodied settings, active observation may include changing viewpoint or moving through the environment to reduce uncertainty, as in zero-shot active visual search for robotic object finding (Park et al. 2023).

Across these settings, active observation policies also require stopping decisions. A system must decide when the current evidence is sufficient, when another crop or query is needed, and when further observation would add cost without improving the decision. Recent native active-perception models treat observation itself as part of iterative reasoning, making the number and order of observation steps an object of evaluation (Xing et al. 2026). This connects active perception to the later evaluation lens: outcome depends on whether the right evidence was found, process depends on the search trajectory, grounding depends on evidence-target alignment, and cost-risk depends on how much observation was required.

3.3 From Observations to Grounded Evidence

Active perception does not end when an agent obtains another observation. The observation must be converted into evidence that can support reasoning, tool use, or action. We therefore distinguish raw observations from grounded evidence. A raw observation is an available input, while grounded evidence is an observation segment, interface element, temporal cue, or sensorimotor signal that has been selected and linked to a task-relevant decision.

Grounded evidence can take several forms across the reviewed systems. In image-based tasks, it may be a region, object, relation, crop, mark, or high-resolution patch. Set-of-Mark prompting makes regions explicitly addressable by overlaying visual marks on an image (Yang et al. 2023a). In GUI and web tasks, evidence often appears as a UI element, OCR span, DOM node, accessibility-tree node, or observed page state. GUI

grounding methods such as SeeClick and CogAgent show how element-level evidence becomes a precondition for later interface action (Cheng et al. 2024; Hong et al. 2024).

For temporal inputs, grounded evidence is often a selected frame, clip, event boundary, or temporal summary. The difficulty is not only finding a relevant moment, but also preserving why that moment supports the answer or action. VideoTree addresses this problem by organizing long videos into adaptive structures that retain coarse-to-fine temporal evidence (Wang et al. 2025b). In embodied settings, grounded evidence may include a visible object, a spatial relation, a depth cue, a camera pose, or a state change observed after movement. Such evidence is grounded because it connects perception to the agent's current task state rather than remaining a detached sensory record.

The conversion from observation to evidence is also the interface between active perception and later loops. Once selected, evidence can be structured in the cognitive workspace, checked through retrieval or tools, used to choose an intervention, or stored for later feedback. Systems such as MM-REACT and ViperGPT illustrate this connection by routing visual evidence into tool-assisted reasoning procedures (Yang et al. 2023b; Surís et al. 2023). In this sense, active perception supplies the entry point for closed-loop agency, but the value of the selected observation depends on how it is represented and used downstream.

This distinction also clarifies why perception failures often propagate through agent trajectories. If the selected evidence is incomplete, weakly grounded, or linked to the wrong target, later reasoning may appear coherent while resting on an unstable basis. In GUI tasks, this may lead to acting on the wrong button or page state; in visual reasoning, it may lead to answering from an irrelevant crop; in embodied tasks, it may lead to moving toward the wrong object or viewpoint. Treating grounded evidence as an explicit intermediate state therefore makes active perception inspectable rather than hidden inside the final answer.

3.4 Evaluation and Failure Modes of Active Perception

Evaluating active perception requires more than measuring whether the final task is completed. A system may reach the correct answer after unnecessary observation steps, or fail because it looked at the wrong region, frame, interface element, or viewpoint. The evaluation target is therefore the evidence-acquisition process itself: what the agent inspected, whether the selected evidence matched the task, and how much observation was needed before the system acted or answered. This makes active perception a natural case for process-level and grounding-level evaluation.

Existing evaluations use several proxies for this process. Image and GUI grounding benchmarks measure whether a model can identify the correct region, element, or coordinate under visual and layout variation. ScreenSpot-Pro stresses this setting with professional high-resolution interfaces that contain small targets and dense layouts (Li et al. 2025). OSWorld-G extends this concern to computer-use grounding, where agents must locate actionable UI regions in realistic operating-system environments (Xie et al. 2025b). These benchmarks make grounding errors visible even when the downstream task is not fully executed.

For temporal inputs, evaluation also concerns whether the system selects the right moments and preserves their relation to the task. Long-video methods are often assessed by final QA accuracy, but this alone does not show whether the model inspected relevant frames or relied on spurious context. VideoTree reports performance under adaptive video representation, where the structure of selected clips affects downstream reasoning (Wang et al. 2025b). Native active-perception systems further expose the number and order of observation steps, making search trajectories part of the evaluation target (Xing et al. 2026).

The main failure modes of active perception can be grouped into missed evidence, wrong evidence, and inefficient evidence acquisition. Missed evidence occurs when the agent fails to inspect a relevant region, frame, text span, or state cue. Wrong evidence occurs when the agent selects an observation that is visually plausible but not aligned with the target. Inefficient evidence acquisition occurs when the agent repeatedly searches, zooms, or queries tools after sufficient evidence is already available. These failures may lead to different outcomes under the O-P-G-C lens: a missed cue may reduce outcome, wrong evidence may damage grounding, and inefficient search may increase process cost.

These evaluation issues suggest that active perception should be reported with both outcome metrics and evidence traces. Outcome metrics show whether the task was solved, while evidence traces show how the observation process contributed to the result. Useful traces include selected regions, frame indices, OCR spans, UI elements, tool calls, viewpoint changes, and stop decisions. Reporting such traces does not require every system to expose a full internal state, but it makes the evidence-acquisition loop easier to audit and compare across tasks.

4 Cognitive Workspace: Structuring Evidence for Reasoning

4.1 From Selected Evidence to Reasoning States

Active perception supplies selected evidence, but reasoning requires a state in which evidence can be compared, combined, and revised. Region-marking methods make this distinction explicit by turning parts of an image into addressable evidence units (Yang et al. 2023a). GUI grounding methods serve a similar role for interface elements, where the agent must identify which UI object can support the next decision (Cheng et al. 2024). Intermediate reasoning traces provide another reference point, since they expose how a model sequences information during problem solving (Wei et al. 2022). Here, cognitive workspace refers to the intermediate organization of selected evidence into reasoning states.

The term cognitive workspace does not denote a fixed neural architecture or memory module. It denotes an intermediate evidence state maintained during reasoning. Such a state may be textual, as in rationales and plans; spatial-visual, as in marked images or sketches; structured, as in object lists, tables, or scene graphs; or programmatic, as in code and visual programs. Visual-sketchpad methods illustrate the spatial-visual case by maintaining external visual marks during reasoning (Hu et al.

2024). Visual-programming methods illustrate the programmatic case by representing reasoning as executable operations over detected evidence (Gupta and Kembhavi 2023).

The need for a workspace becomes clear when evidence must be kept aligned across modality, layout, or time. A system may extract the correct text but lose the table or interface structure that gives the text its role, a difficulty addressed by UI and infographic understanding models such as ScreenAI (Baechler et al. 2024). A web agent may keep a plausible plan while the page state has changed, which makes local task history part of the reasoning state (He et al. 2024). These cases may appear as reasoning errors, but the source can be an unstable organization of evidence. Treating the workspace as inspectable lets us compare systems by how they bind, compose, and maintain evidence.

Table 2 summarizes the main reasoning patterns by the evidence conditions under which they are most useful. The table does not classify systems by architecture; it compares how different forms of reasoning organize evidence, what kind of intermediate state they produce, and what failure mode each pattern tends to expose. This comparison is useful because multimodal agents rarely rely on a single reasoning form: a system may use textual traces for step ordering, visual-spatial reasoning for localization, structured states for relational composition, programmatic traces for executable operations, and stateful workspaces for multi-step interaction.

Table 2 Reasoning patterns for organizing multimodal evidence

Pattern	Evidence condition	State	Example	Main risk
Textual trace	step ordering	rationale / plan	CoT (Wei et al. 2022)	unsupported rationale
Visual-spatial	localization	marks / crops	SoM (Yang et al. 2023a)	lost spatial relation
Structured	relation/schema use	tables / graphs	ScreenAI (Baechler et al. 2024)	schema mismatch
Programmatic	executable operations	code / programs	ViperGPT (Surís et al. 2023)	brittle execution
Stateful	changing task state	task history	WebVoyager (He et al. 2024)	state drift

4.2 Reasoning Representations and Use Conditions

Reasoning representations determine what an agent can do with selected evidence. A textual trace is useful when the task requires ordered explanation or planning, as in chain-of-thought reasoning (Wei et al. 2022). A visual-spatial workspace is more suitable when the answer depends on localization, alignment, or spatial comparison. Structured, programmatic, and stateful representations become useful when evidence

must be composed, executed, or maintained across interaction steps. The choice of representation is therefore a reasoning decision, not only an implementation detail.

Textual traces represent evidence as rationales, plans, or step sequences. This form is useful when a task requires the agent to make intermediate assumptions explicit, preserve an order of reasoning, or communicate why an answer was produced. Visual CoT extends this idea to multimodal reasoning by associating stepwise reasoning with visual evidence (Shao et al. 2024). The limitation is that a fluent textual trace can still be weakly grounded: the explanation may mention evidence that was not actually inspected or may omit the visual basis of a decision.

Visual-spatial representations keep part of the reasoning state in visual form. They are useful when the task depends on where evidence is located, how regions relate, or whether a visual relation is preserved across steps. Set-of-Mark prompting makes image regions addressable, which helps the model refer to visual evidence without relying only on language descriptions (Yang et al. 2023a). Visual Sketchpad follows a related motivation by using sketches as an external visual chain of thought (Hu et al. 2024). These methods are especially relevant for diagrams, charts, spatial layouts, GUI elements, and visual math, where converting everything into text can remove information needed for reasoning.

Structured representations organize evidence into explicit fields, objects, relations, or schemas. They are useful when reasoning requires comparison, counting, relation composition, or consistency checking across multiple evidence units. ScreenAI illustrates this need in documents, tables, charts, and user interfaces, where visual content and layout structure must be interpreted together (Baechler et al. 2024). GUI-oriented models such as CogAgent also rely on structured interface evidence when reasoning about actionable elements (Hong et al. 2024). The main risk is schema mismatch: a structure may look clean while omitting a relation that the task requires.

Programmatic and stateful representations are useful when reasoning must be executable or sustained over time. Programmatic reasoning represents intermediate decisions as code, visual programs, or tool traces, which can make compositional operations more explicit. VisProg represents visual reasoning as programs that call visual modules and compose their outputs (Gupta and Kembhavi 2023). Stateful reasoning maintains local task history, page state, belief state, or intermediate decisions across multiple steps. WebVoyager shows this need in web-agent tasks, where the agent must reason over changing pages rather than a single static observation (He et al. 2024). These two representation forms connect cognitive workspace to later loops, but their role in this section is limited to how they organize evidence for reasoning.

4.3 Reasoning Operations over Multimodal Evidence

The value of a reasoning workspace depends on the operations it supports over evidence. Across the reviewed systems, common operations include binding evidence to task targets, decomposing complex observations, composing intermediate states, verifying candidate claims, and revising the workspace as new evidence appears. These operations turn selected evidence into a basis for an answer, a query, an action proposal, or a state update. The operation perspective also separates the reasoning role of a workspace from the particular form in which it is implemented.

Binding is the operation that links an evidence unit to a task-relevant target. In visual reasoning, region-marking methods make this binding explicit by assigning references to image parts (Yang et al. 2023a). In GUI reasoning, grounding methods bind interface elements to intended operations before an agent can decide what to click, type, or inspect (Cheng et al. 2024). Binding errors occur when the selected evidence is visually plausible but attached to the wrong object, text span, relation, or interface element.

Decomposition and composition are needed when a task cannot be solved from a single evidence unit. Decomposition breaks a complex image, document, chart, interface, or video into smaller units that can be inspected separately. Compositional chain-of-thought methods make this process explicit by separating a multimodal problem into intermediate reasoning steps (Mitra et al. 2024). Composition then combines local evidence into a larger state, such as a relation among objects, a table-level interpretation, or a multi-step plan. Structured models for UI and infographic understanding illustrate this need because layout, text, and visual elements must be interpreted as a connected state (Baechler et al. 2024).

Verification checks whether a candidate reasoning state is supported by the available evidence. Programmatic reasoning makes this operation more explicit by converting reasoning steps into executable programs or code-like traces. VisProg represents compositional visual reasoning as programs that call visual modules and combine their outputs (Gupta and Kembhavi 2023). ViperGPT similarly uses Python execution to perform visual inference through interpretable program steps (Surís et al. 2023). In this section, verification refers to checking a workspace state against available evidence; external retrieval and tool expansion are discussed in Section 5.

Revision updates the reasoning workspace when new evidence changes the current interpretation. This operation is central in multi-step settings, where the agent must maintain what has already been observed, inferred, or ruled out. Web agents illustrate this requirement because page states can change after each action, making the previous reasoning state incomplete or stale (He et al. 2024). Visual workspaces can also support revision when intermediate marks, sketches, or visual notes are updated during reasoning (Hu et al. 2024). Revision connects cognitive workspace to the closed-loop structure of the survey: reasoning states must remain open to later evidence, action outcomes, and feedback.

4.4 Evaluation and Failure Modes of Reasoning Workspaces

Evaluating a reasoning workspace requires looking beyond the final answer. A correct answer can be produced from an incomplete or weakly grounded trace, while an incorrect answer may result from a small binding or state-update error. Chain-of-thought evaluations have shown that intermediate rationales can improve interpretability, but they do not by themselves guarantee faithful reasoning (Wei et al. 2022). For multimodal agents, the evaluation target is therefore not only whether the answer is correct, but whether the intermediate state remains aligned with the evidence it claims to use. We discuss recurring failure patterns rather than an exhaustive taxonomy, since the same system may show several failures in one trajectory.

A recurring problem is unsupported rationale, where a textual trace gives a plausible explanation without sufficient evidence support. This problem is difficult to detect when the rationale is fluent or the final answer is correct. Visual CoT methods reduce part of this risk by associating reasoning steps with visual evidence (Shao et al. 2024). However, a multimodal rationale can still be unfaithful if the cited region, object, or relation was not actually used by the model. Evaluating textual traces therefore requires checking whether each reasoning step is grounded in an identifiable evidence unit.

Grounding can also be lost when the workspace weakens the link between a reasoning state and the perceptual evidence that supports it. This problem appears in spatial reasoning when a region, mark, or crop is associated with the wrong target. Set-of-Mark prompting addresses part of this issue by making visual references explicit, but the reference itself can still be misread or misused (Yang et al. 2023a). In GUI reasoning, lost grounding can cause an agent to reason about the correct intent while acting on the wrong interface element (Cheng et al. 2024). Grounding evaluation should therefore inspect both the selected evidence and the binding between that evidence and the task target.

Mismatch appears when evidence is organized into a schema or executable trace that does not match the task requirement. Structured reasoning can compress evidence into tables, objects, relations, or interface states, but the chosen schema may omit a relation required by the task. ScreenAI illustrates why layout-aware structure matters for documents, charts, and user interfaces (Baechler et al. 2024). Programmatic reasoning has a related risk: an executable trace may fail because the program calls the wrong operation, uses an incorrect intermediate variable, or assumes a visual predicate that was never verified. ViperGPT makes program execution visible, which helps expose some of these failures during visual reasoning (Surís et al. 2023).

State drift becomes visible when the workspace no longer matches the current task or environment state. This problem is common in multi-step web, GUI, and embodied settings because observations, plans, and action outcomes change over time. WebVoyager illustrates this setting because each browser action can alter the page state that subsequent reasoning depends on (He et al. 2024). A reasoning workspace should therefore be evaluated for trace consistency: whether selected evidence, intermediate states, actions, and updates remain mutually compatible across steps. Under the O-P-G-C lens, these failures mainly affect process and grounding, but they can also reduce outcome and increase cost-risk when incorrect reasoning leads to unnecessary or irreversible actions.

5 Knowledge-Tool Expansion: Retrieving and Verifying External Evidence

5.1 Evidence Gaps in Multimodal Reasoning

Knowledge-tool expansion addresses evidence gaps that cannot be resolved from the agent's current observations or intermediate reasoning state. Such gaps may be factual,

computational, procedural, or source-based. A visual entity may need external identification; a chart may require calculation; a GUI state may require parsing; a claim may require support from a document or web source. Retrieval-augmented generation provides a general precedent for treating retrieved passages as external evidence in knowledge-intensive tasks (Lewis et al. 2020). In multimodal agents, external evidence can also take the form of structured data, tool output, parsed interface state, or executable trace.

Evidence gaps are especially visible in tasks where perception does not fully determine the answer. OK-VQA shows that visual question answering may require knowledge beyond the image content (Marino et al. 2019). InfoSeek extends this setting to visual information seeking, where entities in images must be linked to external knowledge sources (Chen et al. 2023b). These tasks show that evidence acquisition does not end with selecting a region, frame, span, or interface element. The agent must judge whether the selected evidence is sufficient for the decision and identify the external source that can supply missing support.

The relevant problem is how the agent controls external evidence channels. ReAct is an early example of interleaving reasoning with actions that query external sources during problem solving (Yao et al. 2023). AVIS adapts this idea to visual information seeking by coordinating tools for image-based question answering (Hu et al. 2023). Chameleon treats tool selection as a compositional process in which modules are invoked according to task demands (Lu et al. 2023). These examples shift the analysis from tool access to the evidential role of what the tool or source returns.

The evidential status of external results is central to knowledge-tool expansion. A retrieved passage, parser output, calculator result, or tool trace is useful only if it can be linked to the task target and checked against the agent's intermediate reasoning state. Knowledge-tool expansion can therefore be treated as a control problem over external evidence: the agent must decide when to query, what source or tool to use, how to interpret the result, and whether the added evidence should revise the current decision. These routes are not mutually exclusive: the same task may combine retrieval, parsing, computation, and cross-checking, but the choice is usually driven by the kind of evidence that is missing. This framing also separates evidence-producing or evidence-verifying calls from actions that change the environment, which are discussed as environment control in Section 6.

5.2 Retrieval as External Evidence Acquisition

Retrieval is the most direct way to fill an evidence gap without changing the external environment. A retrieval step converts an unresolved part of the task into a query over documents, web pages, databases, or other indexed sources. In retrieval-augmented generation, retrieved passages condition generation for knowledge-intensive tasks (Lewis et al. 2020). In multimodal agents, retrieval must also be tied to perceptual evidence: the query may originate from an image region, an OCR span, a video event, a GUI element, or an intermediate reasoning state. The retrieved result becomes useful only when it can be connected back to the evidence need that triggered the search.

The main difficulty is query formation from partial multimodal evidence. A visual question may contain an object whose name, identity, or context is uncertain, so the agent must decide what part of the observation should be verbalized or searched. AVIS illustrates this problem in visual information seeking, where an agent coordinates external tools to answer image-based questions (Hu et al. 2023). MMSearch frames a related setting in which large models act as multimodal search engines that connect visual inputs with search procedures (Jiang et al. 2025). These works show that retrieval quality depends on how the agent represents the missing evidence before the search begins.

Retrieval also requires source and granularity control. Some tasks need a short factual passage, while others need a full document, a web page, an entity record, or a sequence of sources. Self-RAG makes this control explicit by learning when to retrieve and when to critique retrieved information (Asai et al. 2024). Provenance-aware multimodal search further highlights that a returned answer should be traceable to the source that supports it (Tao et al. 2026). For knowledge-tool expansion, the important question is therefore whether retrieval returns evidence at the right level of detail for the current decision.

Iterative retrieval becomes necessary when the first result does not resolve the evidence gap. The agent may reformulate the query, inspect another source, narrow the visual target, or compare retrieved statements against the current reasoning state. ReAct provides a general pattern for alternating reasoning steps with search or action steps during problem solving (Yao et al. 2023). In multimodal settings, the same alternation may involve returning to the image, changing the queried region, or using retrieved text to guide another perception step. Retrieval is therefore part of the closed evidence loop: it adds external support, but it can also redirect what the agent inspects next.

Retrieval failures occur when the returned source is available but does not support the intended decision. A query can drift away from the original evidence gap, a source can be topically related but target-misaligned, or a retrieved passage can be used beyond what it actually states. Work on verifiability in generative search shows that generated answers must be checked against the sources they cite (Liu et al. 2023). Citation-oriented generation similarly treats source attribution as part of answer quality (Gao et al. 2023b). These concerns are directly relevant to multimodal agents because a retrieved source must support both the textual claim and the perceptual target that motivated retrieval.

5.3 Tools as Evidence Transformation and Verification

Tools support knowledge-tool expansion in several ways, but their survey-relevant role is to produce evidence that can be inspected and reused. A tool call may parse visual content, execute a program, compute a value, query an API, identify an object, or check a candidate claim. Toolformer shows that language models can learn to call external tools during generation when the tool output improves prediction (Schick et al. 2023). For multimodal agents, the relevant output is not only a final answer but an intermediate result that can support later reasoning. The tool call therefore matters when its input, output, and connection to the task target can be examined.

One recurring role is evidence transformation. Many multimodal tasks contain information that is visible but not yet organized in a form that supports reasoning. OCR systems, UI parsers, document parsers, object detectors, and layout models convert raw observations into spans, elements, regions, fields, or relations. OmniParser illustrates this role in GUI settings by turning screen observations into structured interface elements for agent use (Lu et al. 2024). ScreenAI shows a related need across documents, tables, charts, and user interfaces, where visual content and layout structure must be represented jointly (Baechler et al. 2024). In these cases, the tool does not necessarily add new world knowledge; it changes the form of existing evidence so that the agent can reason over it.

Tools are also used when evidence must be checked through computation or execution. Some reasoning steps depend on operations that are difficult to perform reliably through free-form text generation, such as arithmetic, symbolic manipulation, spatial counting, or compositional visual queries. Program-aided language models represent part of this process as executable code, making intermediate computation more explicit (Gao et al. 2023a). Program-of-thought prompting follows a related motivation for numerical reasoning by separating computation from natural-language explanation (Chen et al. 2023a). In multimodal reasoning, ViperGPT uses Python execution to perform visual inference through interpretable program steps (Surís et al. 2023). These methods make tool output useful as evidence because the result can be checked against the operation that produced it.

Tool routing becomes important when several external operations could address the same evidence gap. Chameleon treats reasoning as a process of selecting and composing different modules according to task demands (Lu et al. 2023). LLaVA-Plus connects visual-language models with external tools for task-specific multimodal operations (Liu et al. 2024a). MM-REACT combines multimodal reasoning with tool use in cases where the base model alone is insufficient (Yang et al. 2023b). These systems point to a broader control problem: the agent must decide whether the missing support requires search, parsing, calculation, visual recognition, program execution, or another external operation.

Across these uses, tool-mediated evidence introduces failure modes that are different from retrieval errors. A tool may be appropriate but called with the wrong argument, or it may return a correct result that the agent attaches to the wrong target. A parser can miss an interface element, a calculator can be applied to the wrong quantity, and a program can encode an incorrect visual predicate. Visual programming systems make these risks visible because the reasoning path contains explicit module calls or executable steps (Gupta and Kembhavi 2023). For knowledge-tool expansion, the evaluation target is therefore the whole evidence path: why the tool was called, what argument was passed to it, what result it returned, and how that result was used in later reasoning.

5.4 Evaluating Expanded Evidence

Evaluating knowledge-tool expansion requires checking whether external results become valid evidence for the agent's decision. A retrieved passage, search result, parser output, API response, or program trace is not reliable simply because it comes

from an external source. Work on verifiability in generative search makes this distinction explicit by asking whether generated claims are supported by cited sources (Liu et al. 2023). For multimodal agents, the same question has an additional grounding condition: the external result must support the textual claim and the perceptual target that triggered the query or tool call. Evaluation should therefore inspect both the returned result and the link between that result and the agent's current evidence need.

Source quality is one part of this evaluation, but source relevance is often more important for agent trajectories. A source may be authoritative yet answer a different question, describe a different entity, or omit the visual detail that motivated retrieval. Citation-oriented generation treats attribution as part of answer quality because a claim should be traceable to the evidence that supports it (Gao et al. 2023b). Provenance-aware multimodal search makes this issue more visible by requiring search results to preserve the relation between an answer and its supporting source (Tao et al. 2026). In knowledge-tool expansion, a useful source is therefore one that is relevant to the task target, specific enough for the decision, and recoverable after the agent uses it.

The process of expansion must also be evaluated, since a correct external result can still be used incorrectly. A model may issue an underspecified query, call a suitable tool with the wrong argument, accept the first retrieved source without checking, or attach a returned result to the wrong visual target. ARES evaluates retrieval-augmented generation by separating context relevance, answer faithfulness, and answer relevance (Saad-Falcon et al. 2024). Ragas follows a related direction by providing automatic signals for retrieval-augmented generation quality (Es et al. 2024). These evaluation ideas are useful for multimodal agents, but they need to be extended to include perceptual targets, tool arguments, and intermediate reasoning states.

Recurring failures include missing evidence, irrelevant evidence, source-target mismatch, tool misuse, and unverifiable evidence. Missing evidence occurs when the agent answers without querying a source or tool that the task requires. Irrelevant evidence occurs when the returned result is related to the topic but does not support the specific decision. Source-target mismatch appears when the external result supports a claim about a different object, region, time point, interface element, or entity. Tool misuse occurs when the tool is selected correctly but the argument, operation, or interpretation is wrong. Unverifiable evidence appears when the agent cannot expose the source, trace, or operation that produced the support.

Under the O-P-G-C lens, these failures affect different aspects of loop execution. Missing or irrelevant evidence mainly weakens outcome because the final answer or decision lacks support. Source-target mismatch mainly affects grounding because the external evidence is attached to the wrong perceptual or task target. Tool misuse and unverifiable evidence affect process because the trajectory cannot be checked or reproduced. They may also increase cost-risk when repeated searches, unnecessary tool calls, or incorrect external operations consume resources or lead to unsafe actions. This is the boundary where knowledge-tool expansion connects to environment control: once an external operation changes the world state, evaluation must consider the action outcome as well as the evidence it produced.

6 Environment Control: Intervening and Observing State Change

6.1 Evidence-Grounded Intervention

Environment control begins when selected or expanded evidence is used to choose an intervention that changes a digital or physical state. The intervention may be a browser click, a GUI operation, a mobile action, a navigation command, or a robot manipulation primitive. WebArena formulates this setting through browser tasks in which agents must execute actions to complete user goals (Zhou et al. 2024). OSWorld extends the same concern to real computer environments, where agents act across desktop and web applications (Xie et al. 2024). The analytical unit in this section is therefore the relation between evidence, action, and observed state change.

An intervention is evidence-grounded when the chosen action can be traced to a target identified in the current evidence state. In GUI and web tasks, this target may be a button, menu item, text field, link, or page state. SeeClick studies this problem by grounding GUI actions in visible interface elements (Cheng et al. 2024). In embodied tasks, the target may be an object, pose, spatial relation, or contact condition. RT-2 connects visual-language inputs to robotic control actions, making the action target part of the multimodal decision process (Zitkovich et al. 2023). Across these settings, the selected evidence must specify what the intervention is meant to change.

Intervention makes feedback observable through state change. After an action, the agent may observe a page transition, a changed DOM state, a filled form, a new error message, a robot pose change, or a changed object configuration. VisualWebArena evaluates multimodal web agents on tasks where visually grounded actions affect website state (Koh et al. 2024). AndroidWorld similarly evaluates agents through executable interactions in Android applications (Rawles et al. 2025). These settings make state change a form of evidence: the outcome of an action can support, revise, or reject the reasoning state that produced it.

Evidence-grounded intervention also clarifies the boundary between knowledge-tool expansion and environment control. An external call belongs to knowledge-tool expansion when it returns evidence for reasoning or verification. It belongs to environment control when it changes an external state that later has to be observed. This distinction matters because state-changing actions introduce cost, reversibility, and safety concerns that are weaker or absent in evidence-only calls. The central question for environment control is whether an evidence-supported intervention produces the expected state change without creating unnecessary risk.

6.2 Action Spaces and Control Interfaces

Action spaces determine how evidence can be converted into an intervention. They specify the operations available to the agent, the arguments required by each operation, and the abstraction level at which control is expressed. A browser agent may issue clicks, text entry, and navigation commands; a robot may issue motion commands, grasping primitives, or policy outputs. These interfaces shape evidence grounding because selected evidence must identify an actionable target in the representation

expected by the controller. The same evidence state can therefore pose different control problems depending on whether the interface expects a coordinate, an element identifier, a natural-language command, or a continuous control signal.

Digital environments often expose action as discrete operations over interface states. WebArena represents web tasks through executable browser actions that must satisfy user goals in realistic websites (Zhou et al. 2024). VisualWebArena adds visually grounded web tasks in which the agent must connect visual evidence to website operations (Koh et al. 2024). OSWorld extends this setting to desktop and web applications, where actions can affect files, windows, and application state (Xie et al. 2024). AndroidWorld evaluates agents through interactions with Android applications, making taps, text entry, and app navigation part of the control interface (Rawles et al. 2025). These settings make action targets inspectable, but they also make a misgrounded target immediately executable.

Within digital settings, control interfaces expose action targets at different levels of structure. Some agents act on screen coordinates, while others act on UI elements, DOM nodes, accessibility trees, or parsed interface structures. SeeClick studies GUI grounding by linking visual interface elements to executable actions (Cheng et al. 2024). CogAgent treats GUI interaction as a visual-language control problem over graphical interfaces (Hong et al. 2024). Structured interfaces can make the target easier to audit, but they may omit visual details that matter for the task. Coordinate-level control preserves the visual surface, but it makes target binding more sensitive to layout changes, resolution, and small localization errors.

Embodied and robotic environments add continuous dynamics to the control problem. An action may change the camera viewpoint, the robot pose, the object configuration, or the contact state. RT-2 connects vision-language inputs to robotic control actions, showing how web-scale knowledge can be transferred into physical action selection (Zitkovich et al. 2023). OpenVLA represents another line of work in which vision-language-action models map observations and language goals to robot actions (Kim et al. 2025). CALVIN evaluates language-conditioned manipulation through multi-step interaction with objects and environments (Mees et al. 2022). RLBench similarly frames robotic control through manipulation tasks where success depends on state changes in the scene (James et al. 2020). In these environments, evidence grounding must account for geometry, timing, and physical feasibility in addition to semantic target selection.

The difference between digital and physical control lies partly in the cost of an action mismatch. A wrong click may be reversible if the page can be restored, while a wrong file operation, purchase, collision, or grasp can be costly or unsafe. This difference makes action-space design part of evaluation, not only an engineering detail. A useful control interface should make the intended target, action argument, and resulting state change inspectable. The comparison in Table 3 shows why the same evidence-grounding problem has different implications across control settings.

6.3 State Change as Feedback

State change is the feedback signal that makes environment control a closed-loop process. After an intervention, the agent observes whether the environment changed in

Table 3 Representative environment-control settings and state-change evidence

Setting	Action interface	State-change evidence	Main risk
Web or browser	Click, type, scroll, navigate	Page state, URL, DOM change, task completion	Wrong page state or irreversible submission
GUI or desktop	Click, hotkey, menu, file operation	Window state, selected element, file or app state	Wrong target or unsafe operation
Mobile app	Tap, swipe, type, app navigation	Screen transition, app state, notification, form state	State drift across screens
Embodied navigation	Move, turn, inspect, stop	Pose, viewpoint, object proximity, route progress	Collision or wrong viewpoint
Robotic manipulation	Grasp, push, place, trajectory	Object pose, contact, force, success signal	Irreversible or unsafe physical change

the expected direction. The change may appear as a page transition, a DOM update, a completed form field, a moved object, a new pose, a contact signal, or an error message. WebArena makes this action-conditioned feedback explicit by evaluating agents through executable browser tasks (Zhou et al. 2024). The post-action state becomes evidence for whether the preceding target selection, reasoning state, and action argument were valid.

In digital environments, state change is often discrete but still difficult to interpret. A browser or application may update the visible page, change an internal state, open a new window, trigger a validation error, or complete a task without making the success condition visually obvious. VisualWebArena highlights this issue in visually grounded web tasks, where the agent must connect visual observations to website actions and their outcomes (Koh et al. 2024). OSWorld extends the problem to desktop applications, where file state, window focus, and application context can change after an action (Xie et al. 2024). AndroidWorld shows a similar need in mobile applications, where screen transitions and app states define whether an action succeeded (Rawles et al. 2025). These settings require the agent to treat state observation as part of control, not as a separate final check.

In embodied and robotic environments, feedback is tied to geometry, dynamics, and physical contact. An action may change the robot pose, camera viewpoint, object configuration, distance to a target, or force signal. CALVIN evaluates language-conditioned manipulation through long-horizon interaction with objects and environments (Mees et al. 2022). RLBench similarly frames robotic control through manipulation tasks where success depends on state changes in the scene (James et al. 2020). In these settings, feedback may be delayed, noisy, or partially observable, which makes state change harder to use as evidence. The agent must therefore connect action outcomes to the physical condition that the intervention was intended to affect.

State-change feedback also determines whether the agent should continue, revise, or stop. If the observed state matches the expected effect, the agent can use it as support for the current plan. If the state changes in an unexpected way, the agent may need to re-inspect the environment, revise the reasoning state, or choose a recovery action. WebVoyager illustrates this issue in web-agent tasks, where each browser action can alter the page state used by later decisions (He et al. 2024). In robotic settings, vision-language-action models face a related problem because later actions depend on

the state produced by earlier control decisions (Kim et al. 2025). Feedback is therefore not just an outcome signal; it is evidence that changes the next control decision.

The value of state-change feedback depends on whether the system records enough context to make the change interpretable. A final success flag may show that the task was completed, but it does not explain which action produced the relevant change. A useful trace should record the pre-action state, the selected target, the action argument, the post-action state, and any observed error or recovery signal. This trace makes it possible to evaluate whether the agent acted on the right evidence and whether the environment responded as expected. Under the O-P-G-C lens, state-change feedback mainly informs process and grounding, while also affecting outcome and cost-risk when an action has external consequences.

6.4 Evaluation and Failure Modes of Environment Control

Evaluating environment control requires inspecting both task outcome and action trajectory. A successful final state may hide unnecessary, unstable, or weakly grounded actions, while a failed task may result from a single target-selection or state-observation error. WebArena evaluates agents through executable web tasks, making task completion dependent on action sequences rather than static answers (Zhou et al. 2024). OSWorld extends this evaluation problem to computer environments where actions affect applications, files, and windows (Xie et al. 2024). The evaluation target is therefore the full evidence-action-state path: what evidence supported the action, what state changed, and whether the change matched the intended goal.

Environment-control evaluation need to check whether the intervention is applied to the intended target. In web and GUI tasks, target mismatch can appear as clicking the wrong button, typing into the wrong field, opening the wrong page, or operating on a stale interface state. VisualWebArena makes this issue visible in visually grounded web tasks where agents must connect visual evidence to website actions (Koh et al. 2024). In mobile settings, AndroidWorld similarly requires agents to execute actions whose success depends on screen transitions and app states (Rawles et al. 2025). These failures mainly affect grounding because the action may be syntactically valid while being attached to the wrong target.

Process failures arise when the action sequence does not maintain a valid relation between the current state and the next intervention. An agent may repeat an ineffective action, skip a required step, fail to notice that a state has changed, or continue from an outdated plan. ST-WebAgentBench reflects this concern by evaluating safety and trustworthiness in web-agent trajectories (Levy et al. 2026). These failures are not always visible from the final answer because the task may still terminate with a plausible state. Process-level evaluation should therefore record intermediate observations, action arguments, state transitions, and recovery attempts.

Cost-risk failures become more important when actions have external consequences. In digital environments, the risk may involve irreversible submission, file modification, account changes, or unintended transactions. In embodied environments, the risk may involve collision, unsafe contact, object damage, or motion that cannot be easily reversed. SafeVLA reflects this broader concern by studying safety constraints

for vision-language-action systems (Zhang et al. 2025a). Environment-control evaluation should therefore consider not only whether the agent reached the goal, but also how much cost or risk was introduced by the path it took.

These failures often interact within a single trajectory. A weakly grounded target can produce an incorrect action, the incorrect action can change the state, and the changed state can make later reasoning less reliable. Under the O-P-G-C lens, outcome asks whether the goal was reached, process asks whether the action trajectory was valid, grounding asks whether each intervention was supported by the right evidence, and cost-risk asks whether the action path introduced unnecessary or unsafe consequences. This framing keeps environment control tied to evidence rather than treating action as a separate capability. It also prepares the transition to learning, since repeated action outcomes can become feedback for later behavior.

7 Learning and Self-Improvement: Updating Behavior from Feedback

7.1 Feedback Signals and Update Targets

Learning and self-improvement in multimodal agency refer to how a system uses feedback to update later trajectories, memory, policies, or model parameters. A feedback signal has closed-loop value only when it can be linked to the evidence state and action that produced the observed result. Reflexion shows how language agents can convert task outcomes into verbal memory for later attempts (Shinn et al. 2023). Self-Refine treats feedback as an intermediate signal for iterative revision, including cases where the model produces its own feedback (Madaan et al. 2023). For multimodal agents, such feedback must also account for visual grounding, interface state, tool traces, and physical state change.

Feedback signals supervise different parts of an agent trajectory. Outcome feedback indicates whether the final answer or action succeeded, but it may not identify the step that caused the error. Process-level feedback evaluates intermediate reasoning or action trajectories and can make failure assignment more local (Zheng et al. 2026). Grounding feedback checks whether the selected evidence matches the intended region, object, text span, interface element, or state. Cost-risk feedback records whether an attempted update or intervention introduced unnecessary computation, unsafe action, or irreversible change. These signals correspond to the O-P-G-C lens; in the learning setting, they are reused as evidence for later decisions.

The main difficulty is credit assignment across perception, reasoning, tool use, and control. A wrong answer may result from a missing crop, an incorrect OCR span, a misleading retrieved passage, a poorly formed tool query, or an action applied to the wrong target. If feedback only marks success or failure, the agent may update the wrong part of its behavior. In GUI and web agents, task failure may come from perceiving the wrong element, reasoning over a stale page state, or executing an invalid action sequence (Xie et al. 2024). In embodied agents, similar failures may involve viewpoint, contact, timing, or object state. Learning therefore requires feedback to remain connected to the evidence state that produced the decision.

Update targets also differ in persistence. Some systems update the current trajectory by revising an answer, retrying a tool call, or choosing another observation. Others store feedback in memory, adjust prompting or retrieval policies, refine action policies, or train model parameters. Reinforcement-learning studies of vision-language-action models treat environment feedback as a learning signal for action selection (Liu et al. 2025). GUI-oriented reinforcement learning similarly uses interaction feedback to improve action prediction over interface states (Lu et al. 2026). The common requirement is that feedback be represented in a form that can guide a later decision; in this sense, learning and self-improvement are evidence operations rather than general claims of adaptation.

7.2 Experience Memory for Evidence Reuse

After feedback is obtained, the next question is which part of the trajectory should be retained as reusable evidence. In this survey, experience memory denotes a record that preserves an observation, an evidence state, a decision, an action, an outcome, and, when available, a diagnosis of why the outcome occurred. Prior surveys of LLM agents treat memory as a central mechanism for maintaining state across tasks (Wang et al. 2024b). In multimodal agency, the more specific question is whether memory preserves a traceable evidence relation. A useful memory item should indicate what the system observed, what it inferred, what it did, and whether the result supported or revised the prior decision.

Experience memory can store feedback at different granularities. At the local level, it records evidence-action pairs, such as an image region linked to a label, an OCR span linked to an answer, a UI element linked to a click, or a grasp target linked to a state change. At a broader level, it stores trajectory patterns, such as failed search routes, effective decomposition strategies, tool-call recipes, or conditions under which an action should be avoided. Reflexion represents this broader level through verbal lessons derived from previous task outcomes (Shinn et al. 2023). In multimodal settings, a similar mechanism must preserve enough context to recover the visual, textual, interface, or physical evidence to which the lesson applies.

Reflection is the operation that converts raw feedback into a reusable memory item. An effective reflection should specify the condition under which a lesson applies, the evidence that supports it, and the update it recommends. If the reflection is too general, the agent may reuse the lesson in a situation where it does not apply. Self-Refine illustrates an iterative form of feedback and revision in which critiques guide subsequent outputs (Madaan et al. 2023). Process supervision further places evaluation on intermediate steps, which can make reflection more specific to the failing part of a trajectory (Zheng et al. 2026).

Memory reuse introduces another selection problem. At inference time, the agent must decide which prior traces are relevant to the current observation and which experiences should be ignored. In web and GUI environments, this decision is difficult because layouts, page states, and task goals may change across episodes. WebVoyager illustrates how screenshots and action history can condition later decisions in web-agent execution (He et al. 2024). OSWorld also emphasizes that desktop tasks

require agents to operate over changing application states and long action trajectories (Xie et al. 2024). Therefore, Experience memory supports self-improvement only when retrieval, grounding, and state matching are controlled with the same care as the original evidence acquisition process.

7.3 Feedback-Driven Behavior Update

Feedback becomes part of learning only when it changes later behavior. This update does not necessarily require retraining the whole model. It may occur within the current trajectory, as when an agent revises an answer, retrieves again, observes again, or retries an action; it may also occur at a more persistent level, as when memory, policy, or model parameters are updated. Surveys of agentic reinforcement learning also treat interaction feedback as an important signal for improving agent behavior (Zhang et al. 2026). In the framework of this survey, the central question is not whether a system "learns" in a general sense, but how feedback is converted into an inspectable update.

The lightest form of update occurs within a single task trajectory. A system may use feedback to revise an intermediate answer, replace an evidence source, change its observation policy, or select a different tool call. Self-Refine shows how feedback can guide later revisions within the same generation process (Madaan et al. 2023). Reflexion stores failure outcomes as verbal experience so that later attempts can avoid similar errors (Shinn et al. 2023). This form of update has low cost and does not require parameter change, but its improvement depends on whether the feedback is specific enough.

More persistent updates change the action policy or model behavior. For GUI agents, feedback can improve action prediction over interface states, such as click locations, target elements, or next operations. UI-R1 uses reinforcement learning to improve action prediction for GUI agents (Lu et al. 2026). GUI-R1 extends an R1-style training recipe to generalist GUI vision-language-action models (Luo et al. 2025). These methods indicate that feedback can do more than repair a single error: it can change how the system selects actions under similar interface states.

In embodied and robotic agents, behavior update must also account for the state change that follows an action. Feedback in this setting includes not only success or failure labels, but also object pose, contact state, execution outcome, and environmental change. Reinforcement-learning studies of VLA generalization show that environment feedback can be used to improve action selection in vision-language-action models (Liu et al. 2025). VLA-RL and RLinf-VLA represent recent directions that use reinforcement learning to optimize VLA models (Lu et al. 2025). In these systems, the effectiveness of an update depends on whether feedback is correctly linked to the original observation, action arguments, and subsequent state change.

7.4 Evaluation and Failure Modes of Feedback-Driven Learning

Evaluating learning and self-improvement requires evidence that feedback changed later behavior in a warranted way. A higher score after an update is not sufficient, because the improvement may come from retrying, a different task distribution, or an unrelated change in the trajectory. Evaluation therefore needs to connect three

objects: the feedback signal, the update target, and the later behavior that changed. Process reward models are relevant here because they provide supervision over intermediate steps in addition to final outcomes (Zheng et al. 2026). For closed-loop multimodal agents, this supervision should also cover textual traces, selected visual regions, retrieved sources, tool outputs, interface states, and action outcomes.

Learning can fail when feedback is stored or reused without sufficient grounding. A system may learn from the wrong evidence, such as a misleading crop, an incorrect parser output, or a stale interface state. It may also overgeneralize a reflection by applying a lesson from one task to a visual or interaction context where it does not fit. Reflexion makes such verbal lessons explicit, which helps inspection but also makes their scope part of evaluation (Shinn et al. 2023). In multimodal settings, evaluation should check whether the reused experience still matches the current visual, textual, interface, or physical state.

Another risk comes from misalignment between the update objective and the task goal or safety constraint. An agent may improve the measured outcome while increasing tool cost, relying on brittle shortcuts, or taking actions that are difficult to reverse. AI safety research identifies reward hacking and unsafe exploration as important problems that arise from imperfect optimization signals (Amodei et al. 2016). These problems are especially visible in GUI and embodied agents, where exploration may modify files, submit forms, move objects, or change physical state. Agentic reinforcement learning therefore needs to report both outcome gains and the process through which those gains are obtained (Zhang et al. 2026).

Learning evaluation should also make the update path auditable. A survey should specify what feedback was observed, how it was converted into a memory or policy update, which later decisions used that update, and whether the update transferred beyond the original example. GUI reinforcement learning provides a clear setting for this requirement because action prediction over interface states can be compared before and after feedback-driven updates (Lu et al. 2026). Studies of VLA generalization also evaluate whether reinforcement learning improves action selection beyond the training situations (Liu et al. 2025). Under the O-P-G-C lens, effective learning should improve outcome while preserving inspectable process traces, grounded evidence use, and acceptable cost-risk.

8 Auditable Evaluation and Cross-Loop Failures

Closed-loop multimodal agency requires evaluation of the evidence process, not only the terminal prediction or task state. A final answer, completion flag, or reward summarizes the endpoint of an interaction. It does not indicate whether the agent selected relevant observations, organized evidence faithfully, queried an appropriate tool, acted on the intended target, or used feedback in a reliable way. This distinction is especially important for multimodal agents, whose trajectories may include visual regions, screen elements, retrieved sources, tool outputs, environment states, action effects, and failure traces. Evaluation protocols are therefore more informative when they make these intermediate objects available for inspection.

We use the O-P-G-C lens as a compact audit schema for this purpose. Outcome records whether the task goal is reached. Process records whether the trajectory is coherent and necessary. Grounding records whether evidence is aligned with the target decision or action. Cost-risk records the resources, safety concerns, and reversibility issues introduced by closing the loop. These dimensions are not used as four independent subsections of the survey. They provide a shared vocabulary for comparing benchmarks and diagnosing failures across the five evidence operations discussed in Sections 3–7 (Table 4).

Table 4 Operationalization of the O-P-G-C lens for closed-loop multimodal agents. SR denotes success rate. The axes describe what an evaluation protocol makes visible, rather than the quality of a benchmark as a whole

Axis	Audit object	Logged signal	Typical metrics or evidence	Diagnostic cue
Outcome (O)	Answer or state	Goal completion	Accuracy, SR, pass@k, reward, completion time	Task unresolved
Process (P)	Trace or trajectory	Step validity and necessity	Step accuracy, retries, invalid actions, path length, recovery steps	Unreliable trajectory
Grounding (G)	Evidence-target link	Observation support	IoU, target accuracy, citation support, evidence recall, state-change match	Unsupported decision
Cost-Risk (C/R)	Budget or exposure	Resource and safety cost	Tokens, latency, tool calls, action cost, violations, recovery cost	Unsafe or impractical loop

8.1 From Outcome Scores to Auditable Evaluation

Outcome metrics remain necessary because closed-loop behavior is organized around task success. Depending on the domain, the endpoint may be a correct answer, a completed web task, a valid sequence of interface operations, or a robotic state transition. WebArena evaluates whether web agents complete tasks in realistic browser environments (Zhou et al. 2024); OSWorld extends this endpoint to open-ended desktop and web applications (Xie et al. 2024); and CALVIN evaluates long-horizon language-conditioned manipulation through target-state completion (Mees et al. 2022). These endpoint metrics are clear and aggregable, which makes them useful for leaderboards and model comparison.

Outcome metrics, however, compress the trajectory that produced the result. A successful endpoint may reflect correct evidence use, but it may also result from repeated invalid actions, lucky guessing, or a brittle shortcut in the environment. Conversely, a failed endpoint may still contain partially correct perception, retrieval, planning, or recovery behavior that a binary success score discards. AgentBench motivates evaluation across interactive environments where reasoning and decision-making failures are not captured by static answer accuracy (Liu et al. 2024b). Agentic-MME makes this limitation explicit by evaluating whether visual expansion and knowledge expansion are invoked correctly during the process, rather than grading only final

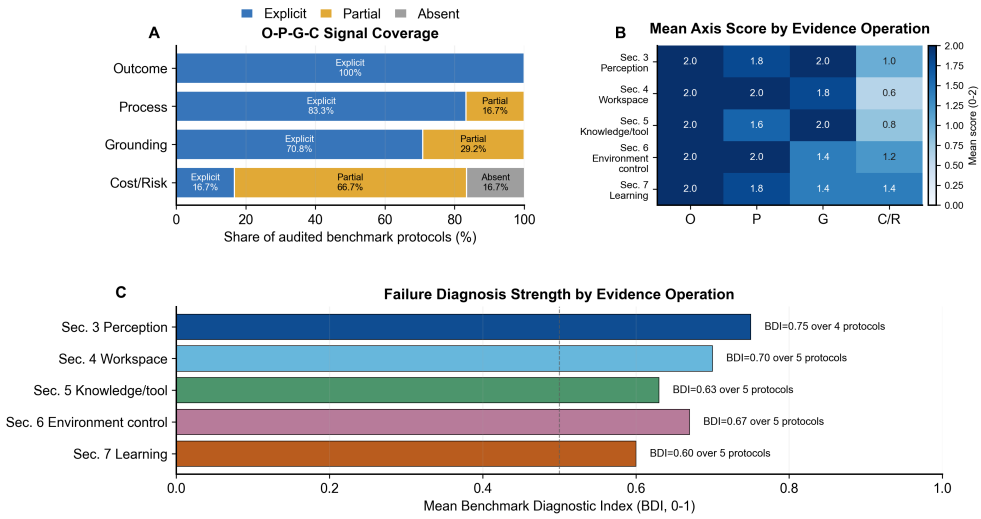


Fig. 5 Evaluation coverage and failure-diagnosis strength across audited benchmark protocols. (a) O-P-G-C signal coverage records whether each protocol makes outcome, process, grounding, and cost-risk signals explicit, partial, or absent. (b) Mean axis score by evidence operation, using the Section 3–7 organization of this survey. (c) Mean Benchmark Diagnostic Index (BDI), which summarizes how strongly protocols support failure diagnosis for each evidence operation; n denotes the number of audited protocols

responses (Wei et al. 2026). For closed-loop agents, the object of evaluation is therefore the relation between the endpoint and the evidence trajectory.

Table 4 expresses this relation as an audit schema. It separates benchmark difficulty from benchmark transparency. A difficult benchmark may expose only final accuracy, whereas an easier benchmark may reveal which evidence was selected, how it was transformed, whether it supported the action, and what cost or risk the trajectory introduced. This distinction allows the survey to compare evaluation protocols without reducing them to a single scalar.

8.2 Benchmark Coverage Across Evidence Operations

Existing benchmarks expose different parts of the closed loop. Perception and workspace benchmarks examine how agents acquire, localize, and transform evidence. Retrieval and deep-research benchmarks examine how agents expand evidence through external sources. Control benchmarks test whether reasoning leads to valid environment changes. Feedback-learning benchmarks ask whether feedback improves later behavior. Figure 5 summarizes the resulting imbalance across audited protocols: outcome is usually explicit, process and grounding are less uniform, and cost-risk remains the least consistently reported dimension (Table 5).

The benchmark comparison indicates that evaluation maturity differs across evidence operations. Active-perception and GUI-grounding benchmarks often provide

Table 5 Representative benchmarks for auditable closed-loop evaluation. Direct means the benchmark explicitly reports the dimension; partial means the dimension is inferable or indirectly measured; weak means the dimension is absent or only lightly discussed. The table uses full evidence-operation labels rather than lettered loop codes

Evidence operation	Benchmark	Setting	Evaluation signals	Main diagnostic gap
Perception and observation selection (Section 3)	V*Bench (Wu and Xie 2024)	Visual search	Outcome direct; process partial; grounding direct; cost-risk weak	Limited interaction and recovery
Evidence workspace construction (Section 4)	TIR-Bench (Li et al. 2026)	Tool-mediated visual reasoning	Outcome direct; process direct; grounding partial; cost-risk weak	Faithfulness of intermediate traces
Knowledge and tool expansion (Section 5)	MMSearch (Jiang et al. 2025)	Multimodal web retrieval	Outcome direct; process partial; grounding direct; cost-risk partial	Action consequences mostly absent
Knowledge and tool expansion (Section 5)	MMDeepResearch-Bench (Huang et al. 2026)	Multimodal report synthesis	Outcome direct; process direct; grounding partial; cost-risk weak	Judge dependence and synthesis drift
Environment intervention (Section 6)	VisualWebArena (Koh et al. 2024)	Visually grounded web tasks	Outcome direct; process partial; grounding direct; cost-risk partial	Web-only scope and attribution
Environment intervention (Section 6)	OSWorld (Xie et al. 2024)	Real desktop applications	Outcome direct; process partial; grounding partial; cost-risk partial	Hard failure attribution
Environment intervention (Section 6)	AndroidWorld (Rawles et al. 2025)	Mobile app interaction	Outcome direct; process partial; grounding partial; cost-risk weak	Platform and task-distribution bias
Environment intervention (Section 6)	ST-WebAgentBench (Levy et al. 2026)	Web safety policies	Outcome direct; process partial; grounding partial; cost-risk direct	Safety policies still domain-bound
Feedback-driven learning (Section 7)	UI-R1 (Lu et al. 2026)	GUI reinforcement learning	Outcome direct; process partial; grounding weak; cost-risk partial	Limited recovery and feedback attribution

explicit target evidence, which makes mislocalization easier to diagnose. Tool and deep-research benchmarks increasingly expose retrieved sources, tool calls, or provenance traces, but they still struggle to show whether the external evidence was necessary and correctly used. Control benchmarks record task completion in realistic environments, yet the cause of failure may be distributed across perception, planning, grounding, and action execution. Learning benchmarks report improvement after feedback or optimization, but they rarely show that the feedback targeted the relevant error. These gaps motivate failure analysis across the closed loop rather than only at the endpoint.

8.3 Diagnosing Cross-Loop Failures

Closed-loop failures are better diagnosed by tracing where an error first appears and how it changes later states. A wrong observation may enter the evidence workspace as a false premise, shape a retrieval query, and select an invalid action target. An unsupported tool result may be stored as evidence and then cited, summarized, or acted on. An incorrect intervention may alter the external state and make the next observation difficult to interpret. A misassigned reward, memory, or feedback signal may preserve a shortcut in later episodes. The same final failure can therefore have different causes, and the same local error can produce different downstream symptoms (Table 6).

Outcome indicates whether the failure affects the endpoint. Process locates where the trajectory becomes invalid. Grounding checks whether the evidence remains aligned with the decision or action. Cost-risk records the practical consequence of the error, including recovery cost and irreversible state change. A web agent that clicks the wrong element illustrates the overlap. The error affects outcome if the task fails, grounding if the clicked element does not match the intended target, process if the trajectory ignores prior evidence, and cost-risk if the click submits a form or changes a user account state.

8.4 Implications for Benchmark Design

The next generation of multimodal-agent benchmarks should treat closed-loop diagnosis as a design goal. First, benchmarks should log the evidence objects that support each step, including visual regions, OCR spans, retrieved sources, tool outputs, interface elements, and state changes. Second, they should evaluate whether each step is necessary and grounded, not only whether a final answer or task state is correct. Third, interaction benchmarks should record cost, reversibility, and safety-relevant state changes, especially for web, GUI, OS, and embodied environments. Fourth, learning benchmarks should connect feedback signals to later behavioral changes, so that improvement can be attributed to the relevant part of the trajectory.

Higher task success is valuable, but it is more informative when paired with evidence about how the system obtained, transformed, verified, acted on, and updated information. The O-P-G-C lens, benchmark coverage table, and cross-loop failure taxonomy together turn the five-loop framework into an audit protocol. They help explain which systems succeed and which parts of the closed-loop evidence process remain brittle, unaudited, costly, or unsafe.

Table 6 Cross-loop failure patterns in multimodal agents. Section references indicate where each failure source is discussed in this survey; the table summarizes how errors become observable, how they propagate, and what evidence can support diagnosis or mitigation

Failure source	Typical failure	Observable cue	Propagation path	Audit or mitigation signal
Perception and observation selection (Section 3)	Observation error	Wrong, missed, or over-selected evidence	Workspace state, retrieval query, or action target is conditioned on the wrong observation	Re-inspection, region evidence, uncertainty-triggered search
Evidence workspace construction (Section 4)	Representation or trace error	Unsupported artifact, plan, schema, or intermediate trace	Tool choice, action plan, or final answer inherits an incorrect intermediate state	Process audit, trace-evidence alignment, ablation of intermediate artifacts
Knowledge and tool expansion (Section 5)	External-evidence error	Stale, irrelevant, or weakly supported source or tool output	Synthesis, citation, decision, or environment action uses unsupported external evidence	Source tracing, query audit, citation-support checking
Environment intervention (Section 6)	Action or state-change error	Wrong target, invalid operation, or unsafe state transition	Next observation, recovery path, or task state is based on a changed environment	Action grounding, rollback, safety shield, state-change verification
Feedback-driven learning (Section 7)	Feedback-update error	Reward, memory, or feedback shortcut	Future trajectories, policy updates, or benchmark-specific behavior reproduce the error	Out-of-distribution feedback, attribution checks, recovery evaluation

9 Conclusion and Future Directions

9.1 Synthesis of the Survey

This survey defines multimodal agency as a closed-loop process organized around evidence. The definition centers the intermediate states that connect multimodal observations to downstream decisions. Evidence may appear as an image region, text span, video frame, interface element, retrieved result, tool output, state change, or feedback signal. Based on this definition, Figure 1 organizes closed-loop multimodal agency into five connected evidence loops: active perception, cognitive workspace, knowledge-tool expansion, environment control, and learning. The framework emphasizes that agent capability depends not only on the final output, but also on how evidence is acquired, organized, expanded, acted on, and updated through feedback.

Sections 3–7 examine the evidence operations that make up the closed loop. Section 3 discusses how systems actively select usable observations from static images, documents, videos, GUI, web, mobile, and embodied environments. Section 4 analyzes how selected evidence is organized into textual, visual-spatial, structured, programmatic, or stateful reasoning states. Section 5 examines how systems supplement insufficient observations through retrieval, tool use, and external verification. Section 6 discusses how evidence is converted into environment intervention and how state change provides feedback. Section 7 examines how feedback is stored, attributed, and converted into later behavior updates.

Section 8 further shows that evaluation of closed-loop multimodal agents needs to go beyond final task success. The O-P-G-C lens decomposes evaluation into outcome, process, grounding, and cost-risk. Figure 5 shows that current benchmark protocols cover outcome most consistently, while grounding, cost-risk, and failure diagnosis remain unevenly covered. This result suggests that evaluation should ask not only whether a system completes a task, but also what evidence it used, what trajectory it followed, and whether the cost and risk of completion are acceptable. The main contribution of this survey is therefore an evidence-centered organization that makes multimodal-agent capabilities, failures, and evaluation criteria easier to compare.

9.2 Future Research Directions

Future benchmarks need clearer reporting of evidence traces. A useful trace should record the task goal, input or environment state, selected evidence, operations applied to the evidence, external sources or tool outputs, environment actions, feedback signals, and the final outcome. Evaluation frameworks such as AgentBench emphasize agent performance across task environments, but aggregate scores alone cannot explain which part of the closed loop produced a failure (Liu et al. 2024b). Work on ALCE and generative-search verifiability shows that checkable links between outputs and sources are important for evaluating generation systems (Gao et al. 2023b). For multimodal agents, this requirement should extend to visual regions, OCR spans, UI elements, tool traces, state changes, and feedback signals.

Future systems also need stable evidence operations across inputs, tasks, and environments. The generalization problem is not only whether a model can process more

modalities, but also whether it can select the right evidence unit under a new input structure and connect it to the appropriate reasoning or action target. VisualWebArena shows that visual web tasks require systems to connect visual cues with website operations (Koh et al. 2024). ScreenSpot-Pro further shows that small targets and dense layouts in high-resolution professional interfaces increase grounding difficulty (Li et al. 2025). VideoTree shows that temporal evidence in long-video tasks also requires selecting the right granularity among frames, events, and cross-frame relations (Wang et al. 2025b). Future evaluation should test whether systems preserve closed-loop evidence consistency when input shift, task shift, and state shift occur together

Feedback learning needs to avoid converting flawed feedback into persistent behavioral bias. Closed-loop agents may learn from outcomes, trajectories, grounding signals, tool outputs, and environment states, but these feedback signals may be incomplete, noisy, or incorrectly attributed. Reflexion shows that task outcomes can be converted into verbal experience for later attempts, but it also makes experience quality a property that needs to be checked (Shinn et al. 2023). Process reward models extend supervision from final outcomes to intermediate steps, providing one direction for finer-grained credit assignment (Zheng et al. 2026). Future systems need memory reuse, policy update, and parameter update to preserve conditions, evidence sources, and state-matching checks so that error amplification is reduced.

Real deployment requires evidence, action, feedback, and risk to be managed within the same execution trajectory. Web, GUI, and mobile agents may submit forms, modify files, access sensitive pages, or rely on unverified external results. ST-WebAgentBench incorporates safety and trustworthiness into web-agent evaluation, showing that such risks cannot be summarized by task success alone (Levy et al. 2026). Embodied agents also need to handle collision, failed grasping, object damage, and unrecoverable states. SafeVLA incorporates safety constraints into the learning problem for vision-language-action models, showing that risk in physical action needs to be considered together with behavior optimization (Zhang et al. 2025a). Future deployment protocols should include action budgets, tool budgets, risk levels, confirmation rules, rollback mechanisms, failure recovery records, and audit logs so that closed-loop agency remains inspectable, constrained, and recoverable in real environments.

Declarations

Funding

The authors received no financial support for the research,authorship, or publication of this article.

Competing interests

The authors declare that they have no competing interests.

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Data availability

The data underlying this survey comprise literature records, evidence-operation coding annotations, and benchmark evaluation annotations. These data will be provided as supplementary material accompanying the article.

Materials availability

Not applicable.

Code availability

The Python scripts used for corpus summaries, benchmark-score aggregation, and figure generation will be provided as supplementary material accompanying the article.

Author contribution

*Y*i Wei: Conceptualization, methodology, investigation, data curation, formal analysis, visualization, and writing—original draft. Yan Gao: Methodology, validation, and writing—review and editing. Junyan Wang: Validation and writing—review and editing. Hong Jia: Conceptualization, supervision, and writing—review and editing.

References

Amodei D, Olah C, Steinhardt J, et al (2016) Concrete problems in AI safety. Preprint at https://arxiv.org/abs/1606.06565

Asai A, Wu Z, Wang Y, et al (2024) SELF-RAG: Learning to retrieve, generate, and critique through self-reflection. In: International Conference on Learning Representations, URL https://openreview.net/forum?id=hSyW5go0v8

Baechler G, Sunkara S, Wang M, et al (2024) ScreenAI: A vision-language model for UI and infographics understanding. In: Larson K (ed) Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24. International Joint Conferences on Artificial Intelligence Organization, Jeju, Korea, pp 3058–3068, https://doi.org/10.24963/ijcai.2024/339

Bajcsy R (1988) Active perception. Proceedings of the IEEE 76(8):966–1005. https://doi.org/10.1109/5.5968

Caffagni D, Cocchi F, Barsellotti L, et al (2024) The revolution of multimodal large language models: A survey. In: Findings of the Association for Computational Linguistics: ACL 2024, pp 13590–13618, https://doi.org/10.18653/v1/2024.findings-acl.807

Chen W, Ma X, Wang X, et al (2023a) Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. Transactions on Machine Learning Research URL https://openreview.net/forum?id=YfZ4ZPt8zd

Chen Y, Hu H, Luan Y, et al (2023b) Can pre-trained vision and language models answer visual information-seeking questions? In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pp 14948–14968, https://doi.org/10.18653/v1/2023.emnlp-main.925

Cheng K, Sun Q, Chu Y, et al (2024) SeeClick: Harnessing GUI grounding for advanced visual GUI agents. In: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, Bangkok, Thailand, pp 9313–9332, https://doi.org/10.18653/v1/2024.acl-long.505

Duan J, Yu S, Tan HL, et al (2022) A survey of embodied AI: From simulators to research tasks. IEEE Transactions on Emerging Topics in Computational Intelligence 6(2):230–244. https://doi.org/10.1109/TETCI.2022.3141105

Es S, James J, Espinosa-Anke L, et al (2024) RAGAs: Automated evaluation of retrieval augmented generation. In: Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations. Association for Computational Linguistics, St. Julians, Malta, pp 150–158, https://doi.org/10.18653/v1/2024.eacl-demo.16

Gao L, Madaan A, Zhou S, et al (2023a) PAL: Program-aided language models. In: Proceedings of the 40th International Conference on Machine Learning, pp 10764–10799

Gao T, Yen H, Yu J, et al (2023b) Enabling large language models to generate text with citations. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pp 6465–6488, https://doi.org/10.18653/v1/2023.emnlp-main.398

Gupta T, Kembhavi A (2023) Visual programming: Compositional visual reasoning without training. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 14953–14962, https://doi.org/10.1109/CVPR52729.2023.01436

He H, Yao W, Ma K, et al (2024) WebVoyager: Building an end-to-end web agent with large multimodal models. In: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics, pp 6864–6890, https://doi.org/10.18653/v1/2024.acl-long.371

Hong W, Wang W, Lv Q, et al (2024) CogAgent: A visual language model for GUI agents. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 14281–14290

Hu Y, Shi W, Fu X, et al (2024) Visual sketchpad: Sketching as a visual chain of thought for multimodal language models. In: Advances in Neural Information Processing Systems, pp 139348–139379, https://doi.org/10.52202/079017-4423

Hu Z, Iscen A, Sun C, et al (2023) AVIS: Autonomous visual information seeking with large language model agent. In: Advances in Neural Information Processing Systems, pp 867–878, https://doi.org/10.52202/075280-0040

Huang P, Zhong Z, Wan Z, et al (2026) MMDeepResearch-Bench: A benchmark for multimodal deep research agents. Preprint at https://arxiv.org/abs/2601.12346

James S, Ma Z, Arrojo DR, et al (2020) RLBench: The robot learning benchmark and learning environment. IEEE Robotics and Automation Letters 5(2):3019–3026. https://doi.org/10.1109/LRA.2020.2974707

Jiang D, Zhang R, Guo Z, et al (2025) MMSearch: Unveiling the potential of large models as multi-modal search engines. In: International Conference on Learning Representations, URL https://openreview.net/forum?id=J2Jyp1SZ0n

Kawaharazuka K, Oh J, Yamada J, et al (2025) Vision-language-action models for robotics: A review towards real-world applications. IEEE Access 13:162467–162504. https://doi.org/10.1109/ACCESS.2025.3609980

Kim MJ, Pertsch K, Karamcheti S, et al (2025) OpenVLA: An open-source vision-language-action model. In: Proceedings of The 8th Conference on Robot Learning, pp 2679–2713, URL https://proceedings.mlr.press/v270/kim25c.html

Kitchenham B, Brereton OP, Budgen D, et al (2009) Systematic literature reviews in software engineering: A systematic literature review. Information and Software Technology 51(1):7–15. https://doi.org/10.1016/j.infsof.2008.09.009

Koh JY, Lo R, Jang L, et al (2024) VisualWebArena: Evaluating multimodal agents on realistic visual web tasks. In: Ku LW, Martins A, Srikumar V (eds) Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, Bangkok, Thailand, pp 881–905, https://doi.org/10.18653/v1/2024.acl-long.50

Levy I, Wiesel B, Marreed S, et al (2026) ST-WebAgentBench: A benchmark for evaluating safety and trustworthiness in web agents. In: International Conference on Learning Representations, URL https://openreview.net/pdf/0846b87f9fc1335b49ed0f9aec51ffb55f7395f1.pdf

Lewis P, Perez E, Piktus A, et al (2020) Retrieval-augmented generation for knowledge-intensive NLP tasks. In: Advances in Neural Information Processing Systems, pp 9459–9474, URL https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html

Li K, Meng Z, Lin H, et al (2025) ScreenSpot-Pro: GUI grounding for professional high-resolution computer use. In: Proceedings of the 33rd ACM International Conference on Multimedia. Association for Computing Machinery, New York, pp 8778–8786, https://doi.org/10.1145/3746027.3755688

Li M, Zhong J, Zhao S, et al (2026) TIR-Bench: A comprehensive benchmark for agentic thinking-with-images reasoning. In: Computer Vision – ECCV 2026, Lecture Notes in Computer Science, vol 17016. Springer, Cham, pp 464–481, https://doi.org/10.1007/978-3-032-37383-0_27

Liu J, Gao F, Wei B, et al (2025) What can RL bring to VLA generalization? an empirical study. In: Advances in Neural Information Processing Systems, pp 107724–107754, https://doi.org/10.52202/085713-3248

Liu NF, Zhang T, Liang P (2023) Evaluating verifiability in generative search engines. In: Findings of the Association for Computational Linguistics: EMNLP 2023, pp 7001–7025, https://doi.org/10.18653/v1/2023.findings-emnlp.467

Liu S, Cheng H, Liu H, et al (2024a) LLaVA-Plus: Learning to use tools for creating multimodal agents. In: Computer Vision – ECCV 2024, Lecture Notes in Computer Science, vol 15105. Springer, Cham, pp 126–142, https://doi.org/10.1007/978-3-031-72970-6_8

Liu X, Yu H, Zhang H, et al (2024b) AgentBench: Evaluating LLMs as agents. In: The Twelfth International Conference on Learning Representations, URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/e9df36b21ff4ee211a8b71ee8b7e9f57-Abstract-Conference.html

Lu G, Guo W, Zhang C, et al (2025) VLA-RL: Towards masterful and general robotic manipulation with scalable reinforcement learning. Preprint at https://arxiv.org/abs/2505.18719

Lu P, Peng B, Cheng H, et al (2023) Chameleon: Plug-and-play compositional reasoning with large language models. In: Advances in Neural Information Processing Systems, pp 43447–43478, https://doi.org/10.52202/075280-1882

Lu Y, Yang J, Shen Y, et al (2024) OmniParser for pure vision based GUI agent. Preprint at https://arxiv.org/abs/2408.00203

Lu Z, Chai Y, Guo Y, et al (2026) UI-R1: Enhancing efficient action prediction of GUI agents by reinforcement learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp 17608–17616, https://doi.org/10.1609/aaai.v40i21.38816

Luo R, Wang L, He W, et al (2025) GUI-R1: A generalist R1-style vision-language action model for GUI agents. Preprint at https://arxiv.org/abs/2504.10458

Ma Y, Song Z, Zhuang Y, et al (2026) A survey on vision-language-action models for embodied AI. IEEE Transactions on Neural Networks and Learning Systems 37(7):3031–3051. https://doi.org/10.1109/TNNLS.2025.3650584

Madaan A, Tandon N, Gupta P, et al (2023) Self-refine: Iterative refinement with self-feedback. In: Advances in Neural Information Processing Systems, pp 46534–46594, https://doi.org/10.52202/075280-2019

Marino K, Rastegari M, Farhadi A, et al (2019) OK-VQA: A visual question answering benchmark requiring external knowledge. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 3195–3204, https://doi.org/10.1109/CVPR.2019.00331

Mees O, Hermann L, Rosete-Beas E, et al (2022) CALVIN: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. IEEE Robotics and Automation Letters 7(3):7327–7334. https://doi.org/10.1109/LRA.2022.3180108

Mitra C, Huang B, Darrell T, et al (2024) Compositional chain-of-thought prompting for large multimodal models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 14420–14431, https://doi.org/10.1109/CVPR52733.2024.01367

Nguyen D, Chen J, Wang Y, et al (2025) GUI Agents: A Survey. In: Findings of the Association for Computational Linguistics: ACL 2025, pp 22522–22538, https://doi.org/10.18653/v1/2025.findings-acl.1158

Page MJ, McKenzie JE, Bossuyt PM, et al (2021) The PRISMA 2020 statement: An updated guideline for reporting systematic reviews. BMJ 372:n71. https://doi.org/10.1136/bmj.n71

Park J, Yoon T, Hong J, et al (2023) Zero-shot active visual search (ZAVIS): Intelligent object search for robotic assistants. In: 2023 IEEE International Conference on Robotics and Automation (ICRA), pp 2004–2010, https://doi.org/10.1109/ICRA48891.2023.10161345

Qin Y, Hu S, Lin Y, et al (2024) Tool learning with foundation models. ACM Computing Surveys 57(4):101. https://doi.org/10.1145/3704435, 40 pages

Rawles C, Clinckemaillie S, Chang Y, et al (2025) AndroidWorld: A dynamic benchmarking environment for autonomous agents. In: International Conference on Learning Representations, URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/01a83bc2f2732a58e6aa731e659e7101-Abstract-Conference.html

Saad-Falcon J, Khattab O, Potts C, et al (2024) ARES: An automated evaluation framework for retrieval-augmented generation systems. In: Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational

Linguistics: Human Language Technologies, pp 338–354, https://doi.org/10.18653/v1/2024.naacl-long.20

Schick T, Dwivedi-Yu J, Dessì R, et al (2023) Toolformer: Language models can teach themselves to use tools. In: Advances in Neural Information Processing Systems, pp 68539–68551, https://doi.org/10.52202/075280-2997

Shang Y, Tang Y, Zhang X, et al (2026) A survey of embodied world models. Preprints.org, preprint at https://doi.org/10.20944/preprints202604.0928.v2; not peer reviewed

Shao H, Qian S, Xiao H, et al (2024) Visual CoT: Advancing multi-modal language models with a comprehensive dataset and benchmark for chain-of-thought reasoning. In: Advances in Neural Information Processing Systems, pp 8612–8642, https://doi.org/10.52202/079017-0275

Shinn N, Cassano F, Gopinath A, et al (2023) Reflexion: Language agents with verbal reinforcement learning. In: Advances in Neural Information Processing Systems, pp 8634–8652, https://doi.org/10.52202/075280-0377

Su Z, Xia P, Guo H, et al (2025) Thinking with images for multimodal reasoning: Foundations, methods, and future frontiers. Preprint at https://arxiv.org/abs/2506.23918

Surís D, Menon S, Vondrick C (2023) ViperGPT: Visual inference via python execution for reasoning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp 11854–11864, https://doi.org/10.1109/ICCV51070.2023.01092

Tao X, Teng Y, Su X, et al (2026) MMSearch-Plus: Benchmarking provenance-aware search for multimodal browsing agents. In: The Fourteenth International Conference on Learning Representations, URL https://openreview.net/forum?id=VGYgG2GH0d

Tong S, Liu Z, Zhai Y, et al (2024) Eyes wide shut? exploring the visual shortcomings of multimodal LLMs. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 9568–9578

Wang J, Xu H, Ye J, et al (2024a) Mobile-Agent: Autonomous multi-modal mobile device agent with visual perception. In: ICLR 2024 Workshop on Large Language Model (LLM) Agents, URL https://arxiv.org/abs/2401.16158

Wang L, Ma C, Feng X, et al (2024b) A survey on large language model based autonomous agents. Frontiers of Computer Science 18(6):186345. https://doi.org/10.1007/s11704-024-40231-1

Wang W, Ding L, Zeng M, et al (2025a) Divide, conquer and combine: A training-free framework for high-resolution image perception in multimodal large language

models. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp 7907–7915, https://doi.org/10.1609/aaai.v39i8.32852

Wang X, Zhang Y, Zohar O, et al (2024c) VideoAgent: Long-form video understanding with large language model as agent. In: Computer Vision – ECCV 2024, Lecture Notes in Computer Science, vol 15138. Springer, Cham, pp 58–76, https://doi.org/10.1007/978-3-031-72989-8_4

Wang Z, Yu S, Stengel-Eskin E, et al (2025b) VideoTree: Adaptive tree-based video representation for LLM reasoning on long videos. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp 3272–3283, URL https://openaccess.thecvf.com/content/CVPR2025/papers/Wang_VideoTree_Adaptive_Tree-based_Video_Representation_for_LLM_Reasoning_on_Long_CVPR_2025_paper.pdf

Wei J, Wang X, Schuurmans D, et al (2022) Chain-of-thought prompting elicits reasoning in large language models. In: Advances in Neural Information Processing Systems, vol 35. Curran Associates, Inc., Red Hook, NY, USA, pp 24824–24837, https://doi.org/10.52202/068431-1800

Wei Q, Yang Y, Wang S, et al (2026) Agentic-MME: What agentic capability really brings to multimodal intelligence? Preprint at https://arxiv.org/abs/2604.03016

Wohlin C (2014) Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering. Association for Computing Machinery, New York, NY, USA, EASE '14, pp 1–10, https://doi.org/10.1145/2601248.2601268

Wu P, Xie S (2024) V^*: Guided visual search as a core mechanism in multimodal LLMs. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 13084–13094

Xie J, Chen Z, Zhang R, et al (2025a) Large multimodal agents: A survey. Visual Intelligence 3(1):24. https://doi.org/10.1007/s44267-025-00093-y

Xie T, Zhang D, Chen J, et al (2024) OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In: Advances in Neural Information Processing Systems, pp 52040–52094, https://doi.org/10.52202/079017-1650

Xie T, Deng J, Li X, et al (2025b) Scaling computer-use grounding via user interface decomposition and synthesis. In: Advances in Neural Information Processing Systems, https://doi.org/10.52202/085713-0817

Xing Z, Xu R, Wang Y, et al (2026) Native active perception as reasoning for omnimodal understanding. In: Proceedings of the 43rd International Conference on

Machine Learning, URL https://openreview.net/forum?id=3dLPVNhFZh

Yang J, Zhang H, Li F, et al (2023a) Set-of-mark prompting unleashes extraordinary visual grounding in GPT-4V. Preprint at https://arxiv.org/abs/2310.11441

Yang Z, Li L, Wang J, et al (2023b) MM-REACT: Prompting ChatGPT for Multimodal Reasoning and Action. Preprint at https://arxiv.org/abs/2303.11381

Yao H, Zhang R, Huang J, et al (2025) A survey on agentic multimodal large language models. Preprint at https://arxiv.org/abs/2510.10991

Yao S, Zhao J, Yu D, et al (2023) ReAct: Synergizing Reasoning and Acting in Language Models. In: International Conference on Learning Representations, URL https://openreview.net/forum?id=WE_vluYUL-X

Yin S, Fu C, Zhao S, et al (2024) A survey on multimodal large language models. National Science Review 11(12):nwae403. https://doi.org/10.1093/nsr/nwae403

Zhang B, Zhang Y, Ji J, et al (2025a) SafeVLA: Towards safety alignment of vision-language-action model via constrained learning. In: Advances in Neural Information Processing Systems, pp 153335–153373, https://doi.org/10.52202/085713-5128

Zhang G, Geng H, Yu X, et al (2026) The landscape of agentic reinforcement learning for LLMs: A survey. Transactions on Machine Learning Research URL https://openreview.net/forum?id=RY19y2RI1O

Zhang X, Gao Z, Zhang B, et al (2025b) Adaptive chain-of-focus reasoning via dynamic visual search and zooming for efficient VLMs. Preprint at https://arxiv.org/abs/2505.15436

Zheng C, Zhu J, Ou Z, et al (2026) A comprehensive survey of process reward models: Data generation, model construction, and usage. In: Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, San Diego, California, United States, pp 3591–3607, https://doi.org/10.18653/v1/2026.acl-long.163

Zhou S, Xu FF, Zhu H, et al (2024) WebArena: A realistic web environment for building autonomous agents. In: The Twelfth International Conference on Learning Representations, URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/4410c0711e9154a7a2d26f9b3816d1ef-Abstract-Conference.html

Zitkovich B, Yu T, Xu S, et al (2023) RT-2: Vision-language-action models transfer web knowledge to robotic control. In: Proceedings of The 7th Conference on Robot Learning, pp 2165–2183, URL https://proceedings.mlr.press/v229/zitkovich23a.html