ACCELERATING AGENTIC MULTIMODAL LLMS VIA ASYNCHRONOUS TOOL-TRAJECTORY SPECULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Agentic multimodal large language models use tools to acquire additional visual evidence, but repeated alternation between large-model inference and tool execution accumulates latency. Replacing the entire agent with a smaller model can reduce accuracy, motivating a division of work between exploratory action generation and final answering. We introduce **SpecAgent**, a training-free, asynchronous speculative execution framework in which a small draft model generates and executes tool actions while a large main model assesses the resulting evidence and remains responsible for answering. Separate quality and answerability assessments guide continued exploration, early answering, and main-model takeover. Assessment overlaps with draft exploration, while takeover reuses completed tool interactions to limit repeated work. Across five benchmarks, SpecAgent achieves $1.34\times$–$2.70\times$ end-to-end speedup over the Qwen3.5-27B agent while maintaining broadly comparable accuracy. These results demonstrate an accuracy–latency trade-off through small–large model collaboration in tool-assisted multimodal reasoning.

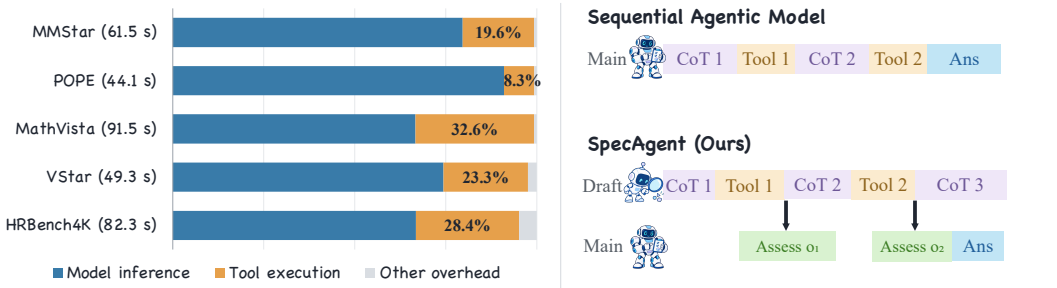
1 INTRODUCTION

Multimodal large language models (MLLMs) are moving beyond static visual perception toward tool-mediated active perception. In the static setting, a model reasons over the visual input supplied at the outset, without taking actions to obtain additional observations (Bai et al., 2025; Wang et al., 2024; Hurst et al., 2024). Rather than reasoning solely over the initial image, tool-use MLLMs can crop regions, extract text, or execute image-processing code to obtain additional observations during reasoning (OpenAI, 2025; Doshi, 2026; Zhang et al., 2025; Zheng et al., 2026; Hong et al., 2026; Lai et al., 2026). These observations guide further inspection, supporting fine-grained and multi-step visual understanding. However, this capability comes with additional latency: *when subsequent actions depend on earlier observations, the agent repeatedly alternates between large-model inference and tool execution. Their costs accumulate across interaction rounds, making efficient tool-assisted reasoning an important challenge.* As shown in Fig. 1(a), tool execution accounts for 8.3%–32.6% of total latency across five benchmarks for the Qwen3.5-27B agent.

Existing acceleration methods reduce these costs at different levels. Token-level speculative decoding accelerates individual generations but retains the surrounding tool-use process (Wang & Yang, 2025; Li et al., 2024a; Leviathan et al., 2023). SpecEyes (Huang et al., 2026) instead uses small-model, tool-free predictions to bypass exploration for selected queries, while routing the remaining queries to the large-model agent. Yet tasks requiring additional observations still incur the cost of tool-assisted exploration. Directly replacing the large model with a smaller one may reduce inference cost at the expense of answer accuracy, but preserving answer quality need not require the large model to generate every exploratory action. Then, *can small-large model collaboration accelerate tool-assisted exploration while preserving the large model's answer quality?*

To address this question, we propose **SpecAgent**, a training-free framework for asynchronous speculative tool execution. Our central idea is to separate the model that advances exploration from the model that assesses its progress and produces the final answer. A small draft model generates and executes exploratory tool actions, sharing the resulting evidence with the main model. As illustrated in Fig. 1(b), the main model assesses this evidence while the draft continues subsequent exploration, allowing the two processes to overlap without stepwise approval. This division of work aims to

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



(a) Latency cost of tool execution **(b) Asynchronous exploration and assessment**

Figure 1: **Tool-execution latency motivates asynchronous exploration and assessment.** (a) Latency breakdown of the Qwen3.5-27B agent across five multimodal benchmarks. Tool execution accounts for 8.3%–32.6% of total latency, highlighting a substantial cost beyond model inference. Parentheses report mean end-to-end latency. (b) A conventional agent alternates model generation and tool execution. SpecAgent delegates exploration to a draft model, allowing the main model to assess previously returned evidence while the draft generates and executes subsequent tool actions. This overlap provides an opportunity to reduce end-to-end latency.

reduce large-model exploration costs without relying on the small model to solve the entire task independently.

To orchestrate this asynchronous flow, SpecAgent introduces dual assessments of quality and answerability, disentangling whether an exploratory step is valid from whether the accumulated evidence is sufficient to conclude. This mechanism allows the large model to exit early once confident or seamlessly take over when the draft model encounters repeated failures, maximizing concurrency while curbing unproductive exploration. Across five benchmarks, *i.e.*, MMStar, POPE, MathVista, VStar, and HRBench4K, SpecAgent delivers a $1.34\times$–$2.70\times$ end-to-end speedup over the full Qwen3.5-27B agent with negligible impact on accuracy. On the challenging VStar benchmark, combining both assessments sustains a 95.81% accuracy while reducing mean latency from 68.22 to 49.27 seconds, highlighting the synergy between rapid speculative probing and authoritative oversight.

Our contributions are threefold: ♥ **Framework.** We introduce SpecAgent, an asynchronous execution framework that delegates tool exploration to a small model while a large model assesses returned observations and remains responsible for answering. ♠ **Coordination.** We design an online coordination mechanism that distinguishes useful progress from answer sufficiency, enabling answer decisions and takeover during exploration while retaining prior approved tool interactions as context. ♣ **Performance and Findings.** Across five multimodal benchmarks, SpecAgent delivers a $1.34\times$–$2.70\times$ end-to-end speedup over its synchronous, large-model counterpart Qwen3.5-27B with virtually no loss in accuracy. Ablation studies on VStar show that quality and answerability assessments are complementary, as combining them preserves the high accuracy of the quality-only variant while substantially reducing latency.

2 Related Work

Agentic Multimodal Reasoning. Agentic MLLMs extend visual reasoning by interleaving reasoning with visual inspection and executable operations to acquire and interpret additional observations during inference (OpenAI, 2025; Doshi, 2026; Zheng et al., 2026; Zhang et al., 2025; Yan et al., 2026; Wang et al., 2026; Mao et al., 2026). DeepEyes uses reinforcement learning to develop visually grounded active perception (Zheng et al., 2026). Thyme combines supervised fine-tuning with reinforcement learning to support code execution for image manipulation and mathematical computation (Zhang et al., 2025). These approaches primarily focus on learning when and how to interact with visual inputs. However, multi-step exploration often requires each tool result before the next action can be generated, causing large-model inference and tool execution to proceed sequentially and their latencies to accumulate across interaction rounds. SpecAgent reduces this end-to-end latency by *delegating tool exploration to a small model while the large model asynchronously*

assesses returned evidence. This allows assessment to overlap with continued exploration, while retaining the large model for final answering.

Speculative Reasoning for Language Agents. Speculative decoding accelerates token generation through draft proposals and target verification (Leviathan et al., 2023; Chen et al., 2023; Cai et al., 2024; Li et al., 2024a). SpecReason and DREAM-R extend speculation to reasoning steps (Pan et al., 2026; Hu et al., 2026). Dynamic Speculative Planning explores asynchronous planning (Guan et al., 2026), while Speculative Actions investigates pre-execution of predicted actions (Ye et al., 2025). For multimodal agents, SpecEyes bypasses tool exploration using selected small-model answers, whereas queries requiring additional evidence are routed to the full agent (Huang et al., 2026). Evidence acquisition therefore still relies on the full agent for these queries. Our work extends speculation to this evidence-acquisition process: For queries that still require tool exploration, SpecAgent *delegates this exploration to a small model*, while the main model *evaluates the resulting evidence for quality and answer sufficiency*. This assessment does not require the small model's actions to match those of the main model. It guides continued exploration, final answering, or main-model takeover with the interaction history preserved.

3 SPECAGENT: ASYNCHRONOUS SPECULATIVE TOOL EXPLORATION

We first formulate agentic multimodal reasoning and its latency objective (Sec. 3.1). We then present the complete SpecAgent framework (Sec. 3.2), explain how it decides whether to continue exploration, answer, or transfer control to the main model (Sec. 3.3), and describe how asynchronous scheduling and context reuse support efficient execution (Sec. 3.4). Algorithm 1 summarizes the overall procedure.

3.1 TASK FORMULATION

Agentic multimodal reasoning. We model an agentic MLLM as a policy π_k coupled with a tool execution interface $\mathcal{T}$. Given an image–question pair $x = (I, q)$, the system maintains a state $s_t = (x, h_t, z_t)$, where h_t is the interaction history and z_t is the tool environment, including persistent variables created by code execution. Conditioned on the input and history, the policy produces either a tool action $a_t \in \mathcal{A}_{\text{tool}}$ or a final answer $y \in \mathcal{Y}$. A tool action specifies the operation and its arguments; its execution updates the environment and history as

$$a_t \sim \pi_k(\cdot \mid x, h_t), \qquad (o_t, z_{t+1}) = \mathcal{T}(a_t, z_t; I), \qquad h_{t+1} = h_t \parallel (a_t, o_t), \tag{1}$$

where o_t contains textual output and any returned images, and $\parallel$ appends the corresponding assistant–tool messages, including accompanying reasoning. The model observes the environment through these outputs, while tools retain access to the original image. Starting from an empty interaction history, execution yields a trajectory $\xi_k(x) = ((a_1, o_1), \ldots, (a_{K_k}, o_{K_k}), y)$, with $K_k = 0$ for an answer produced without tools.

Sequential latency. Eq. 1 establishes a sequential dependency between tool execution and subsequent reasoning: the history h_{t+1} incorporates the observation o_t, so generating the next action $a_{t+1} \sim \pi_k(\cdot \mid x, h_{t+1})$ requires the execution of a_t to finish first. A conventional agent therefore alternates model generation and tool execution along $\xi_k(x)$. Let M denote the main model used by the reference agent. For its trajectory $\xi_M(x)$ with $K_M(x)$ tool interactions and one tool call per generation turn, the end-to-end latency is

$$T_{\text{base}}(x) = \underbrace{\sum_{t=1}^{K_M(x)} c_t^M + c_{\text{answer}}^M}_{\text{model inference}} + \underbrace{\sum_{t=1}^{K_M(x)} c_t^{\text{tool}}}_{\text{tool execution}} + c_{\text{runtime}}, \tag{2}$$

where c_t^M is the model-request duration for generating the t_{th} action, c_t^{tool} is its tool-execution duration, and c_{answer}^M is the final-answer generation time. The residual c_{runtime} includes orchestration and additional attempts not represented in the displayed trajectory. For multiple tool calls in a turn, model generation is counted once and the tool term measures the elapsed execution time of that turn. Thus, latency depends on both the number of interactions and their individual costs. These

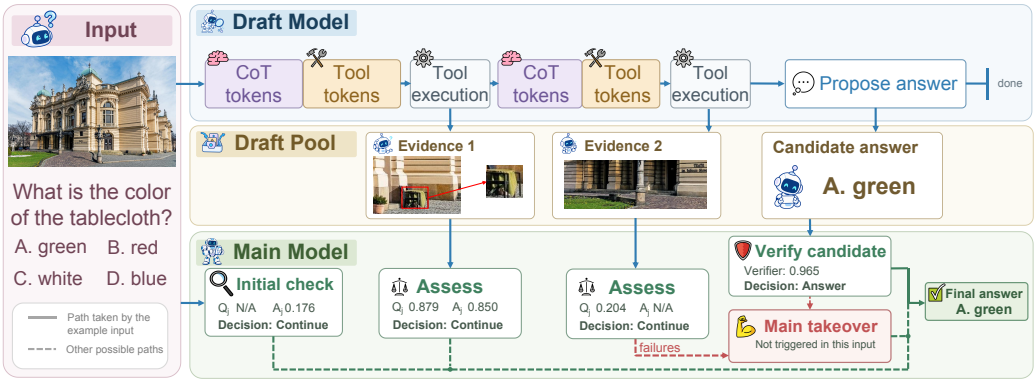


Figure 2: **Overview of SpecAgent.** The draft model generates and executes tool actions, sharing evidence with the main model for asynchronous quality and answerability assessment. An initial check enables answering from the original input, while online assessments guide continued exploration, answering, or takeover. A draft candidate undergoes terminal verification using accumulated evidence. When takeover is needed, the main model reuses stored interactions to continue exploration.

dependencies constrain subsequent observation-conditioned actions, but allow another model to assess already available observations while exploration continues.

Efficiency objective. We introduce a smaller draft model D and seek a coordination policy ω that reduces latency while preserving answer quality within a prespecified tolerance ϵ. For an evaluation distribution $\mathcal{D}$ and a fixed resource configuration, the objective is

$$\min_{\omega \in \Omega} \mathbb{E}_{x \sim \mathcal{D}}[T_\omega(x)] \quad \text{s.t.} \quad \mathrm{Acc}(\omega) \geq \mathrm{Acc}(M) - \epsilon, \tag{3}$$

where Ω denotes admissible coordination policies, Acc is expected answer correctness under the evaluation protocol, and $T_\omega(x)$ measures time from initialization to the actual returned result, including assessment, takeover, and cancellation. This is an empirical design objective rather than a quality guarantee. The following sections instantiate ω by assigning exploration to D and overlapping it with assessment by M, which remains responsible for the final answer.

3.2 Method Overview

As shown in Fig. 2, SpecAgent decouples tool exploration from main model reasoning: the draft model D generates and executes tool actions, while the main model M assesses the resulting evidence entries to answer or take over tool exploration when needed. Through a shared evidence pool, the draft continues without waiting for approval, allowing exploration and assessment to overlap. Algorithm 1 summarizes the framework.

Speculative exploration. Starting from $x = (I, q)$, the draft $\mathcal{D}$ generates and executes exploratory actions through $\mathcal{T}$. Each completed action-observation pair is stored as an evidence entry e_j in the shared pool $\mathcal{P}$. The draft then continues from its updated history without waiting for main-model $\mathcal{M}$ approval, allowing assessment to overlap with subsequent exploration.

Online assessment. As draft exploration begins, the main model checks whether x alone supports an answer. If it abstains, it evaluates each evidence entry e_j as it becomes available, assessing whether the recorded interaction provides useful progress and whether the evidence supports answering directly. These judgments guide continuation, answering, or takeover while draft exploration proceeds, without waiting for the draft to complete the task.

Answering and takeover. If an evidence entry e_j supports an answer, the main model will answer directly and request draft cancellation. Otherwise, draft exploration continues unless repeated quality

failures or tool errors trigger main-model takeover. If the draft produces an answer or reaches its exploration budget before either outcome, the main model reviews the accumulated evidence entries in $\mathcal{P}$, treating any draft answer as a hypothesis. It returns an accepted answer or, if review remains unresolved, takes over tool exploration using the available context. Sec. 3.3 details these decision rules, while Sec. 3.4 explains asynchronous execution and context reuse.

3.3 Online Assessment and Adaptive Takeover

Assessment workflow. At initialization, the main model M checks whether the original input x supports an immediate answer while the draft model D begins exploring. If accepted, the answer is selected and draft cancellation is requested. Otherwise, M assesses each returned evidence entry e_j for useful progress and answer sufficiency, guiding continued exploration, answering, or takeover. Both assessments use x and only the latest entry. A candidate answer produced by D undergoes terminal review using the accumulated evidence pool $\mathcal{P}$.

Quality assessment. The main model labels an evidence entry e_j as GOOD or BAD according to its usefulness for answering the question. Given the label log-probabilities $\ell_{j,G}$ and $\ell_{j,B}$, we normalize them to obtain a quality score Q_j and a pass indicator v_j:

$$Q_j = \frac{\exp(\ell_{j,G})}{\exp(\ell_{j,G}) + \exp(\ell_{j,B})}, \qquad v_j = \mathbf{1}[Q_j \geq \tau_Q], \tag{4}$$

where τ_Q is the quality threshold. The score Q_j expresses relative support for GOOD over BAD. A useful entry can pass even if it provides only partial evidence.

Answerability assessment. Given the original input x and the latest entry e_j, the main model either abstains or proposes a candidate answer $\hat{y}_j$ with a brief evidence-based account and a verification decision $\rho_j \in \{0, 1\}$. Here, $\rho_j = 1$ indicates that the candidate is supported without another tool call. Let $\ell_{j,1}^{\text{ver}}$ and $\ell_{j,0}^{\text{ver}}$ be the log-probabilities of the two verification outcomes. Analogous to the quality assessment, we obtain an answerability score A_j and an acceptance indicator u_j as follows:

$$A_j = \frac{\exp(\ell_{j,1}^{\text{ver}})}{\exp(\ell_{j,1}^{\text{ver}}) + \exp(\ell_{j,0}^{\text{ver}})}, \qquad u_j = \mathbf{1}[\hat{y}_j \in \mathcal{Y} \ \wedge \ \rho_j = 1 \ \wedge \ A_j \geq \tau_A], \tag{5}$$

where $\mathcal{Y}$ contains valid task answers and τ_A is the answerability threshold. The indicator u_j equals one only when the judge provides a valid candidate, verifies that it is supported, and assigns it a score of at least τ_A. If both $u_j = 1$ and $v_j = 1$, the candidate $\hat{y}_j$ will be returned as the final answer and draft exploration will be terminated; otherwise, the draft model continues exploring. The initial assessment applies the same acceptance criterion to x alone, without a quality check.

Online decision rule. The main model accepts $\hat{y}_j$ only if both assessments pass. Otherwise, repeated quality failures trigger takeover, while exploration continues below the failure threshold:

$$d_j = \begin{cases} \textsc{Answer}(\hat{y}_j), & u_j = v_j = 1, \\ \textsc{Takeover}, & u_j v_j = 0 \text{ and } b_j \geq B_Q, \\ \textsc{Continue}, & \text{otherwise}, \end{cases} \qquad b_j = (1 - v_j)(b_{j-1} + 1), \tag{6}$$

where b_j counts ($b_0 = 0$), B_Q limits the consecutive quality failures. Accepting an answer requests draft cancellation. Explicit tool errors bypass assessment and trigger takeover after a separate consecutive-error threshold. Context reuse at takeover is described in Sec. 3.4.

Terminal review. When the draft produces a candidate answer, the main model verifies it against x and the accumulated evidence pool $\mathcal{P}$. This review evaluates the draft's candidate, allowing evidence from multiple steps to support the decision. The draft answer is accepted only if its support score exceeds τ_A; otherwise, the main model takes over exploration with the accumulated evidence as reference context.

3.4 Asynchronous Execution with Context Reuse

Nonblocking execution. After adding an evidence entry e_j to $\mathcal{P}$, the draft continues generating and executing tool actions while the main model assesses that entry. The quality and answerability assessments run concurrently. Once both finish, their results determine whether to answer, continue, or take over according to Eq. 6. This wait does not block the draft $\mathcal{D}$. Let r_j be the time e_j becomes available, and let s_j and f_j be its assessment start and completion times. For a sequence of entry assessments,

$$s_j \geq \max(r_j, f_{j-1}), \qquad f_j = s_j + \max(c_j^Q, c_j^A), \tag{7}$$

where c_j^Q and c_j^A are the quality and answerability assessment durations, including serving delays. The main model starts assessing e_j once it is available and the previous entry's assessment is complete, with possible polling delays (first relation in Eq. 7). If both assessments start together, their combined duration is the longer of the two (second relation in Eq. 7). Meanwhile, the draft continues generating and executing tool actions. Terminal review and takeover costs are accounted for separately.

To isolate the benefit of overlap, we compare the evidence-entry pipeline with a synchronous variant that waits for each assessment before continuing draft exploration. This comparison excludes the initial answerability check and terminal review.

Proposition 3.1 (Latency reduction through asynchronous execution)**.** *Consider a fixed sequence of $K \geq 1$ evidence entries, without early termination or takeover. Each draft generation–tool execution step takes time $d \geq 0$, and each entry assessment takes time $g = \max(c^Q, c^A) \geq 0$. Both schemes perform the same work with the same durations. Draft steps execute sequentially, and entries are assessed in order, one at a time, with the two assessments for each entry running concurrently. Under immediate scheduling and no resource contention, the pipeline completion times are*

$$\begin{aligned} T_{\text{sync}} &= K(d+g), \\ T_{\text{async}} &= d + g + (K-1)\max(d, g). \end{aligned} \tag{8}$$

Consequently, the latency reduction is

$$T_{\text{sync}} - T_{\text{async}} = (K-1)\min(d, g). \tag{9}$$

The proof and an extension to varying step durations appear in Appendix A. After the first entry, overlap hides the shorter of exploration and assessment behind the longer one.

Context reuse at takeover. At takeover, the main model reuses evidence entries in $\mathcal{P}$, with the context determined by the takeover trigger. ***i)*** If the draft model repeatedly fails quality assessment or encounters consecutive tool errors, the main model receives complete draft tool interactions from before the triggering turn. An interaction is retained only if all its evidence entries passed quality assessment. ***ii)*** If terminal review yields no accepted answer, the main model receives all accumulated evidence entries in $\mathcal{P}$ as reference context. In both cases, the main model continues exploration with its own tool budget. Supplying this context does not re-execute recorded actions or restore an earlier tool environment state. In case *i*, retained interactions may depend on omitted steps, which limits reuse for stateful tools.

4 Experiments

We organize the evaluation around three research questions:

RQ1: What accuracy-latency trade-off does SpecAgent achieve across multimodal benchmarks?

RQ2: How does this trade-off vary with the reference agent's tool-use depth?

RQ3: How do quality and answerability assessments contribute to this trade-off, and how sensitive is it to the quality threshold?

4.1 Experimental Setup

Benchmarks and evaluation metrics. We evaluate SpecAgent on five complementary benchmarks: MMStar (Chen et al., 2024) for general multimodal understanding, POPE (Li et al., 2023) for object

Table 1: **Main results on five multimodal benchmarks.** We report task accuracy (Acc.) and end-to-end wall-clock speedup (Spd.) relative to the Qwen3.5-27B agent (1.00×). The best and second-best results are marked in **bold** and underlined fonts.

Method	VStar		MMStar		HRBench4K		POPE		MathVista	
	Acc. ↑	Spd. ↑	Acc. ↑	Spd. ↑	Acc. ↑	Spd. ↑	Acc. ↑	Spd. ↑	Acc. ↑	Spd. ↑
W/o Tools										
Qwen3.5-27B (Team, 2026)	90.05%	5.18×	77.20%	3.91×	85.25%	4.61×	90.64%	13.13×	81.40%	4.34×
W/ Tools										
Qwen3.5-9B (Team, 2026)	88.48%	1.31×	76.00%	**2.60**×	86.38%	1.13×	90.21%	2.65×	80.50%	**2.26**×
Qwen3.5-27B (Team, 2026)	94.76%	1.00×	**81.27%**	1.00×	87.50%	1.00×	**91.50%**	1.00×	82.90%	1.00×
SpecEyes (Huang et al., 2026)	95.29%	1.29×	77.27%	0.80×	86.75%	0.76×	91.06%	0.90×	83.00%	0.81×
SpecAgent (ours)	**95.81%**	**1.66**×	80.80%	1.34×	**91.12%**	**1.50**×	91.13%	**2.70**×	**83.80%**	1.80×

hallucination, MathVista (Lu et al., 2024) for visual mathematical reasoning, and VStar (Wu & Xie, 2024) and HRBench4K (Wang et al., 2025) for fine-grained visual perception. The accuracy and end-to-end speedup are reported to assess the effectiveness and efficiency of SpecAgent: accuracy is the percentage of correctly answered samples under the corresponding benchmark protocol, while speedup measures the efficiency gain of SpecAgent relative to the full baseline agent. Specifically, we record the end-to-end latency of every sample, including reasoning and tool executions. The speedup of a method m is defined relative to the full baseline agent as the ratio between their average end-to-end latencies:

$$\mathrm{Speedup}(m) = \frac{\frac{1}{N}\sum_{i=1}^{N} T_i^{\mathrm{base}}}{\frac{1}{N}\sum_{i=1}^{N} T_i^{m}}, \tag{10}$$

where T_i^{base} and T_i^m denote the end-to-end latency of the full baseline agent and method m on sample i, respectively.

Comparison Baselines and Implementation Details. We compare SpecAgent against four baselines: (1) *Main Model w/ Tools*, which uses the main model to complete the task with full tool access; (2) *Main Model w/o Tools*, which uses the same main model but disables all tools; (3) *Draft Agent*, which directly uses the draft model and its tools without main-model intervention; (4) *SpecEyes*, a tool-free routing baseline implemented following (Huang et al., 2026). Both SpecEyes and SpecAgent use Qwen3.5-27B (Team, 2026) as the main model and Qwen3.5-9B (Team, 2026) as the draft model. SpecEyes adopts min-token aggregation for cognitive gating, with $K = 64$, $\epsilon = 10^{-6}$, and a gating threshold of $\tau = 0.95$. For SpecAgent, the quality-assessment τ_Q threshold is set to 0.7, while the answerability threshold τ_A is set to 0.9. All experiments are conducted on four NVIDIA RTX 4090 48GB GPUs.

4.2 Main Results

[To Q1] Consistent acceleration with broadly comparable accuracy. SpecAgent reduces end-to-end latency across all five benchmarks while achieving comparable or higher accuracy. Specifically, Tab. 1 reports 1.34×–2.70× speedup over the full Qwen3.5-27B agent, corresponding to approximately 25.4%–63.0% lower mean latency. Accuracy increases by 1.05, 3.62, and 0.90 percentage points on VStar, HRBench4K, and MathVista, respectively. On MMStar and POPE, accuracy remains comparable to the full agent, with POPE achieving the largest speedup of 2.70×. These results support our motivation that *maintaining answer quality does not require the large model to generate every exploratory action*.

[To Q1] Collaboration balances answer quality and inference cost. By combining draft-model tool exploration with main-model assessment, SpecAgent achieves higher accuracy than the standalone draft agent and the main model without tools, while remaining faster than the full main-model agent. As shown in Tab. 1, SpecAgent provides 1.34×–2.70× speedup over the full agent and exceeds the draft agent's accuracy by 0.92–7.33 percentage points. Compared with the main model without tools, its accuracy gains range from 0.49 to 5.87 points. These comparisons show that model collaboration offers a practical accuracy–latency trade-off: higher answer quality than the simpler acceleration alternatives, at lower latency than full main-model tool exploration.

Table 2: **Results by reference tool depth on VStar.** Groups use the tool-call count of Qwen3.5-27B. All methods use the same samples within each group. Cells report accuracy (%) / E2E latency (s). Best and second-best results are marked in **bold** and underlined fonts

Method	1 call $N = 23$	2 calls $N = 99$	> 2 calls $N = 69$
Qwen3.5-27B	**100.00** / 38.93	**97.98** / 70.24	88.41 / 111.26
Qwen3.5-9B	91.30 / 32.17	92.93 / 44.75	78.26 / 61.26
SpecEyes	**100.00** / 32.81	95.96 / 42.16	92.75 / 87.61
SpecAgent	**100.00** / 31.77	95.96 / 30.51	**94.20** / 82.95

Table 3: **Assessment ablation on VStar.** QJ: quality assessment ($\tau_Q = 0.7$); AG: answerability ($\tau_A = 0.9$). Time is mean E2E latency; speedup is relative to Full Agent ($N = 191$).

Method	QJ	AG	Acc. (%)	Time (s)	Spd.
Qwen3.5-27B	✗	✗	94.76	81.65	1.00×
SpecAgent	✗	✗	92.15	63.42	1.29×
	✓	✗	**95.81**	68.22	1.20×
	✗	✓	92.67	33.88	**2.41**×
	✓	✓	**95.81**	49.27	1.66×

[To Q1] More consistent acceleration than tool-free routing. SpecAgent achieves lower latency than SpecEyes across all five benchmarks, with higher accuracy on four and comparable accuracy on POPE. As shown in Tab. 1, SpecAgent accelerates the full agent by $1.34\times$–$2.70\times$, whereas SpecEyes provides $1.29\times$ speedup on VStar but only $0.76\times$–$0.90\times$ on the other benchmarks. Accuracy gains over SpecEyes are 0.52, 3.53, 4.37, and 0.80 percentage points on VStar, MMStar, HRBench4K, and MathVista, respectively. On MathVista, SpecAgent is also $2.22\times$ faster than SpecEyes; on POPE, it achieves $3.00\times$ speedup relative to SpecEyes with comparable accuracy (91.13% versus 91.06%). These results show that the complete SpecAgent framework provides more consistent acceleration than the evaluated tool-free routing baseline.

4.3 Framework Analysis

[To Q2] Acceleration extends beyond shallow reference trajectories. SpecAgent reduces mean latency across all reference tool-use depths, although the speedup does not increase monotonically with depth. Tab. 2 groups 191 VStar samples by the full agent's tool-call count, using identical subsets for all methods. On the one-call subset, SpecAgent retains 100% accuracy with $1.23\times$ speedup. The largest gain occurs at two calls: latency decreases from 70.24 to 30.51 seconds ($2.30\times$), with a 2.02-point accuracy decrease; compared with SpecEyes, accuracy is identical, and latency is 27.6% lower. Above two calls, accuracy rises from 88.41% to 94.20% while latency falls from 111.26 to 82.95 seconds ($1.34\times$). These results reveal different benefits across reference tool-use depths: the two-call subset shows the largest latency reduction, whereas the deeper subset gains both accuracy and speed over the full agent. On the latter subset, the substantial accuracy advantage over the draft agent further highlights the value of small–large model collaboration.

[To Q3] Quality and answerability assessments have complementary effects. Quality assessment improves answer accuracy, while answerability reduces latency; combining them retains the quality-only accuracy at substantially lower latency. On VStar with $\tau_Q = 0.7$ and $\tau_A = 0.9$ (Tab. 3), adding quality assessment to the variant with neither component raises accuracy from 92.15% to 95.81%, with latency increasing from 63.42 to 68.22 seconds. Answerability alone reduces latency to 33.88 seconds but achieves 92.67% accuracy. Combining both reaches 95.81% at 49.27 seconds, reducing quality-only latency by 27.8% and improving answerability-only accuracy by 3.14 points. These results show that the accuracy gains from quality assessment can be retained at lower latency by adding answerability assessment, supporting separate decisions about whether exploration is useful and whether it can stop.

[To Q3] Stricter quality filtering does not consistently improve accuracy. Increasing τ_Q does not yield a monotonic improvement in accuracy and can reduce speedup, as shown in Fig. 3. For example, $\tau_Q = 0.60$ and 0.70 both achieve approximately 95.8% accuracy, above the full-agent reference, but yield $1.78\times$ and $1.66\times$ speedup, respectively. The highest displayed accuracy occurs at $\tau_Q = 0.85$; further increasing the threshold to 0.95 reduces both accuracy and speedup. These results show that stricter acceptance of intermediate tool results does not necessarily improve final-answer accuracy, supporting the selection of τ_Q based on the accuracy–latency trade-off rather than strictness alone.

[To Q3] More conservative answer acceptance is not uniformly better. The answerability threshold offers a trade-off between accuracy and speed. Increasing τ_A does not necessarily improve final-answer accuracy and can reduce speedup. As shown in Fig. 4, $\tau_A = 0.90$ and 0.95 both achieve



Figure 3: **Sensitivity to the quality assessment threshold τ_Q on VStar.** The curve shows accuracy, and bars show average per-sample speedup relative to the full agent; dashed lines indicate the reference accuracy and $1.00\times$ speedup. At $\tau_Q = 0.60$, SpecAgent exceeds the reference accuracy with $1.78\times$ speedup. Increasing the threshold does not consistently improve accuracy.

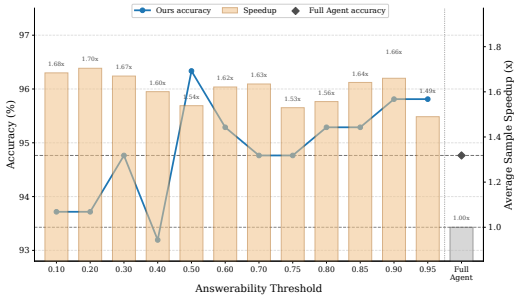


Figure 4: **Sensitivity to the answerability threshold τ_A on VStar.** The curve shows accuracy, and bars show average per-sample speedup relative to the full agent; dashed lines mark the reference accuracy and $1.00\times$ speedup. Increasing τ_A from 0.90 to 0.95 leaves accuracy unchanged while reducing speedup from $1.66\times$ to $1.49\times$.

approximately 95.8% accuracy, above the full-agent reference, but increasing the threshold reduces average per-sample speedup from $1.66\times$ to $1.49\times$. The highest displayed accuracy, approximately 96.3%, occurs at $\tau_A = 0.50$ with $1.54\times$ speedup, whereas $\tau_A = 0.20$ achieves the largest speedup of $1.70\times$ with accuracy below the reference. These results highlight a distinction between conservative candidate acceptance and final-answer reliability: a higher acceptance threshold does not necessarily produce a more accurate completed trajectory. The additional computation incurred by deferring an answer must therefore be justified by its accuracy benefit, rather than assumed to improve reliability.

5 Discussion

Serving capacity shapes asynchronous gains. Asynchronous execution reduces waiting within a sample, but assessment requests still compete for shared serving resources. Our concurrency study in Tab. 4 shows that increasing the number of simultaneously processed samples narrows the latency advantage; at high concurrency, request backlogs are accompanied by response timeouts that reduce the valid-output rate. Thus, nonblocking exploration does not eliminate the resource cost of assessment, and deployment efficiency should be evaluated jointly with completion reliability. Detailed results are provided in Appendix G.1.

Standalone accuracy does not fully determine collaborative utility. A smaller model's performance as a complete agent does not fully characterize its value within a collaborative system. On VStar and HRBench4K, SpecAgent achieves higher accuracy than the draft-only agent while retaining acceleration over the full main-model agent. These results support allocating exploration and answer assessment separately: a model need not match the main agent's standalone accuracy to participate in an effective division of work. The benefit is established for the complete framework, whose initial answering, online assessment, and takeover jointly determine the outcome.

6 Conclusion

We presented **SpecAgent**, a training-free framework which delegates tool exploration to a small model while a large model assesses the resulting evidence and produces the final answer. Online quality and answerability assessments guide early answering and takeover, supported by asynchronous execution and context reuse. Across five multimodal benchmarks, SpecAgent achieves $1.34\times$–$2.70\times$ end-to-end speedup with broadly comparable accuracy. These results support separating exploratory action generation from final answering as a practical approach to efficient agentic multimodal reasoning.

AI USE STATEMENT

In this work, we used generative AI tools for these tasks in the required-disclosure category: assisting the interpretation and presentation of author-provided experimental results during manuscript revision. We did not use generative AI tools to generate synthetic data sets, develop theoretical models or conceptual frameworks, formulate or prove mathematical claims, propose or refine hypotheses, design methodology or experiments, implement methods, or perform qualitative or thematic data analysis. Additionally, we used generative AI tools for tasks in the recommended-disclosure category: polishing the manuscript to improve readability, drafting and revising prose, suggesting document structure, and offering suggestions for modifying figures and illustrative diagrams. The authors checked all AI-assisted interpretations against the underlying experimental outputs and fixed evaluation tables, and reviewed all AI-assisted text and figures for accuracy and originality. We take responsibility for the final manuscript, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies inference efficiency for tool-using multimodal models using existing benchmarks. Faster execution does not remove the underlying models' potential for hallucination, biased predictions, or inappropriate tool use, and the assessment mechanisms in SpecAgent do not constitute a safety guarantee. We use the models and datasets mentioned in the manuscript, respecting their licenses and applicable privacy requirements.

REPRODUCIBILITY STATEMENT

We are committed to fully transparent and reproducible research. Section 3 and Algorithm 1 describe the framework, assessment rules, and takeover procedure. Section 4.1 identifies the evaluation benchmarks, baselines, and accuracy and latency metrics, while Appendix D describes implementation details, including assessment inputs, context reuse, and timing boundaries. The assumptions and complete proof of Proposition 3.1, together with the extension to varying step durations, are provided in Appendix A. We will release our code.

REFERENCES

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*, 2024.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Jiaqi Wang, Yu Qiao, Dahua Lin, et al. Are we on the right way for evaluating large vision-language models? *Advances in Neural Information Processing Systems*, 37:27056–27087, 2024.

Rohan Doshi. Introducing Agentic Vision in Gemini 3 Flash. `https://blog.google/innovation-and-ai/technology/developers-tools/agentic-vision-gemini-3-flash/`, January 2026. Accessed: 2026-02-24.

Haodong Duan, Junming Yang, Yuxuan Qiao, Xinyu Fang, Lin Chen, Yuan Liu, Xiaoyi Dong, Yuhang Zang, Pan Zhang, Jiaqi Wang, et al. Vlmevalkit: An open-source toolkit for evaluating large multi-modality models. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pp. 11198–11201, 2024.

Yilin Guan, Qingfeng Lan, Fei Sun, Dujian Ding, Devang Acharya, Chi Wang, William Wang, and Wenyue Hua. Dynamic speculative agent planning. In *International Conference on Learning Representations*, volume 2026, pp. 7695–7717, 2026.

Jack Hong, Chenxiao Zhao, ChengLin Zhu, Weiheng Lu, and Guohai Xu. Deepeyesv2: Toward agentic multimodal model. In *International Conference on Learning Representations*, volume 2026, pp. 114851–114872, 2026.

Yunhai Hu, Zining Liu, Xiangyang Yin, Tianhua Xia, Bo Bao, Eric Sather, Vithursan Thangarasa, and Sai Qian Zhang. Dream-r: Multimodal speculative reasoning with rl-based refined drafting, precise verification, and fully parallel execution. *arXiv preprint arXiv:2605.28678*, 2026.

Haoyu Huang, Jinfa Huang, Zhongwei Wan, Xiawu Zheng, Rongrong Ji, and Jiebo Luo. Speceyes: Accelerating agentic multimodal llms via speculative perception and planning. *arXiv preprint arXiv:2603.23483*, 2026.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Xin Lai, Junyi Li, Wei Li, Tao Liu, Tianjian Li, and Hengshuang Zhao. Mini-o3: Scaling up reasoning patterns and interaction turns for visual search. In *International Conference on Learning Representations*, volume 2026, pp. 76722–76746, 2026.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International conference on machine learning*, pp. 19274–19286. PMLR, 2023.

Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 292–305, 2023.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle: Speculative sampling requires rethinking feature uncertainty. *arXiv preprint arXiv:2401.15077*, 2024a.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-2: Faster inference of language models with dynamic draft trees. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 7421–7432, 2024b.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations (ICLR)*, 2024.

Delin Mao, Chenghao Sun, Jingwei Song, Chishui Chen, and Linfeng Zhang. Toolvision: Learning when and how to use visual tools with capability-aligned supervision. *arXiv preprint arXiv:2608.08907*, 2026.

OpenAI. Introducing OpenAI o3 and o4-mini. `https://openai.com/index/introducing-o3-and-o4-mini/`, April 2025. Accessed: 2026-02-24.

Rui Pan, Yinwei Dai, Zhihao Zhang, Gabriele Oliaro, Zhihao Jia, and Ravi Netravali. Specreason: Fast and accurate inference-time compute via speculative reasoning. *Advances in Neural Information Processing Systems*, 38:12730–12749, 2026.

Qwen Team. Qwen3. 5: Towards native multimodal agents. *URL: https://qwen. ai/blog*, 2026.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.

Qixun Wang, Yang Shi, Letian Cheng, Zhuoran Zhang, Yan He, Yuqi Tang, Qi Zhang, Xinlei Yu, Ruizhe Chen, Tianrun Xu, et al. Beacon: Knowing when and how to perform agentic visual reasoning. *arXiv preprint arXiv:2607.28595*, 2026.

Wenbin Wang, Liang Ding, Minyan Zeng, Xiabin Zhou, Li Shen, Yong Luo, Wei Yu, and Dacheng Tao. 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*, volume 39, pp. 7907–7915, 2025.

Yancheng Wang and Yingzhen Yang. Efficient visual transformer by learnable token merging. *IEEE transactions on pattern analysis and machine intelligence*, 47(11):9597–9608, 2025.

Penghao Wu and Saining Xie. V*: Guided visual search as a core mechanism in multimodal llms. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13084–13094. IEEE, 2024.

Shilin Yan, Jintao Tong, Hongwei Xue, Xiaojun Tang, Yangyang Wang, Kunyu Shi, Guannan Zhang, Ruixuan Li, and Yixiong Zou. Act wisely: Cultivating meta-cognitive tool use in agentic multimodal models. *arXiv preprint arXiv:2604.08545*, 2026.

Naimeng Ye, Arnav Ahuja, Georgios Liargkovas, Yunan Lu, Kostis Kaffes, and Tianyi Peng. Speculative actions: A lossless framework for faster agentic systems. *arXiv preprint arXiv:2510.04371*, 2025.

Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. Draft& verify: Lossless large language model acceleration via self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11263–11282, 2024.

Yi-Fan Zhang, Xingyu Lu, Shukang Yin, Chaoyou Fu, Wei Chen, Xiao Hu, Bin Wen, Kaiyu Jiang, Changyi Liu, Tianke Zhang, et al. Thyme: Think beyond images. *arXiv preprint arXiv:2508.11630*, 2025.

Ziwei Zheng, Minghao Yang, Jack Hong, Chenxiao Zhao, Guohai Xu, Le Yang, and Chao Shen. Deepeyes: Incentivizing" thinking with images" via reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 126775–126798, 2026.

APPENDIX

A LATENCY ANALYSIS OF ASYNCHRONOUS EXECUTION

Setting and assumptions. We analyze the time from the start of the first draft step to completion of the final entry assessment. Both execution schemes process the same fixed sequence of $K \geq 1$ evidence entries, without early termination or takeover. The initial answerability check, terminal review, and final-answer generation are outside this pipeline. Draft steps execute sequentially, and entries are assessed in order, one at a time. Within each entry, quality and answerability assessments start together and run concurrently. Scheduling is immediate, and concurrency does not change operation durations through resource contention.

Let $\delta_j \geq 0$ be the time to generate and execute the j-th draft action, and let $g_j = \max(c_j^Q, c_j^A) \geq 0$ be the corresponding assessment duration. We use δ_j to distinguish exploration time from the decision d_j in Eq. 6. In asynchronous execution, the availability time of entry j is

$$r_j = \sum_{i=1}^{j} \delta_i. \tag{11}$$

With $f_0 = 0$, Eq. 7 specializes under immediate scheduling to

$$f_j = \max(r_j, f_{j-1}) + g_j. \tag{12}$$

General completion time. For varying step durations, the asynchronous pipeline completes at

$$T_{\text{async}} = f_K = \max_{1 \leq i \leq K} \left(\sum_{j=1}^{i} \delta_j + \sum_{j=i}^{K} g_j \right). \tag{13}$$

Proof. We prove the expression by induction on the number of entries. For $K = 1$, nonnegative durations give $f_1 = r_1 + g_1 = \delta_1 + g_1$, as required. Suppose the expression holds for $K - 1$. The recurrence then gives

$$f_K = \max\{r_K + g_K,\ f_{K-1} + g_K\} \tag{14}$$

$$= \max\left\{ r_K + g_K,\ \max_{1 \le i \le K-1} \left(r_i + \sum_{j=i}^{K} g_j \right) \right\} \tag{15}$$

$$= \max_{1 \le i \le K} \left(r_i + \sum_{j=i}^{K} g_j \right). \tag{16}$$

Substituting Eq. 11 establishes Eq. 13. □

Constant step durations.

Proof of Proposition 3.1. Set $\delta_j = d$ and $g_j = g$ for every entry. Eq. 13 reduces to

$$T_{\text{async}} = \max_{1 \le i \le K} \big(id + (K - i + 1)g \big). \tag{17}$$

The maximized expression is affine in i, with slope $d - g$. If $d \ge g$, a maximum occurs at $i = K$, giving $Kd + g$. If $d \le g$, a maximum occurs at $i = 1$, giving $d + Kg$. These expressions coincide when $d = g$, and together yield

$$T_{\text{async}} = d + g + (K - 1)\max(d, g). \tag{18}$$

Synchronous execution waits for each assessment before starting the next draft step, so $T_{\text{sync}} = K(d + g)$. Using $d + g - \max(d, g) = \min(d, g)$ gives

$$T_{\text{sync}} - T_{\text{async}} = (K - 1)\min(d, g). \tag{19}$$

For $K = 1$, both schemes take $d + g$ and the saving is zero. The expressions also hold if either duration is zero. □

Scope of the result. The comparison isolates asynchronous scheduling relative to a synchronous draft–main pipeline, rather than relative to the full main-model agent. In SpecAgent, assessment outcomes may alter the executed trajectory through early answering or takeover. Polling delays, resource contention, and cooperative cancellation also affect observed latency. The proposition therefore quantifies an overlap opportunity under fixed work; the net speedup of the complete system is evaluated using end-to-end measurements.

B Algorithm of SpecAgent

We present the algorithm of SpecAgentin Algorithm 1.

C Related Works

Speculative Decoding for LLMs. Speculative decoding accelerates autoregressive generation by using inexpensive draft proposals to reduce the number of sequential target-model decoding steps. Early approaches generate candidate tokens with a smaller model and verify them in parallel with the target model; a rejection-sampling correction preserves the target distribution (Leviathan et al., 2023; Chen et al., 2023). Subsequent work improves how candidates are generated and organized for verification. To avoid maintaining a separate draft model, Draft & Verify skips selected layers of the target model during drafting and uses the full model for verification (Zhang et al., 2024). Medusa adds multiple decoding heads to predict future tokens and uses tree attention to verify candidate continuations together (Cai et al., 2024). EAGLE instead performs autoregression in feature space, conditioning on sampled tokens to address feature uncertainty (Li et al., 2024a). Building on this approach, EAGLE-2 uses context-dependent draft confidence to construct dynamic candidate trees, allocating proposals according to their estimated acceptance likelihood (Li et al.,

2024b). These methods accelerate token generation within model calls. In a tool-using agent, faster decoding can shorten action generation, but it does not by itself remove the need to execute tools and acquire observations before subsequent observation-conditioned actions. SpecAgent addresses this interaction-level cost by delegating tool exploration to a small model and overlapping it with main-model assessment. Its acceptance decisions concern the usefulness of completed interactions and the evidence supporting an answer, without requiring the draft trajectory to reproduce the main model's token sequence. This operates at a different level from exact speculative sampling and does not inherit its distribution-preservation guarantee.

D Implementation Details

Evaluation framework and model serving. We use VLMEvalKit (Duan et al., 2024) to evaluate SpecAgent on MMStar, POPE, MathVista, VStar, and HRBench4K. The agent is integrated into the evaluation framework through an API-based inference interface, which receives each benchmark sample and returns the final answer for evaluation. In all experiments, `api-nproc`, a parameter in VLMEvalKit, was set to 4, which means that up to four samples were evaluated concurrently. We use Qwen3.5-27B as the main model M and Qwen3.5-9B as the draft model D. Both models are served using vLLM and accessed through OpenAI-compatible APIs. Experiments are conducted on four NVIDIA RTX 4090 GPUs with 48 GB memory each. The full-agent and tool-free main-model baselines use Qwen3.5-27B, while the draft-only baseline uses Qwen3.5-9B.

E More Discussions

More tool use does not imply more avoidable latency. The largest speedup occurs in the two-call reference group, rather than the group with the most tool calls. Nevertheless, SpecAgent improves both accuracy and latency in the group where the full agent makes more than two calls. This pattern suggests that reference tool-use depth alone is insufficient to predict the benefit of speculation: additional interactions can offer opportunities for overlap, but may also involve exploration that remains necessary under the collaborative policy. Reference call counts characterize the baseline's behavior, rather than the intrinsic tool requirements of a sample.

Progress assessment and answer acceptance serve different objectives. The ablation results indicate that deciding whether exploration is useful and deciding whether an answer is sufficiently supported have different effects on the final outcome. Adding answerability assessment to quality assessment retains the same aggregate accuracy while reducing mean latency, whereas answerability assessment alone is faster but less accurate. The threshold sweeps further show that making either judgment stricter does not consistently improve final-answer accuracy. Together, these findings support treating exploration control and answer acceptance as separate decisions, rather than assuming that uniformly conservative judgments yield the best accuracy–latency trade-off.

F Case Study

To better illustrate the workflow of SpecAgent, we select three representative VStarBench cases covering distinct answer paths. In Fig. 5 main model answers during its initial check, bringing draft exploration to an early close. Fig. 6 traces how main model rejects an unsupported Draft answer and takes over the solving process. The case where the draft tool results are evaluated and used by main model to answer is shown in Fig. 7.

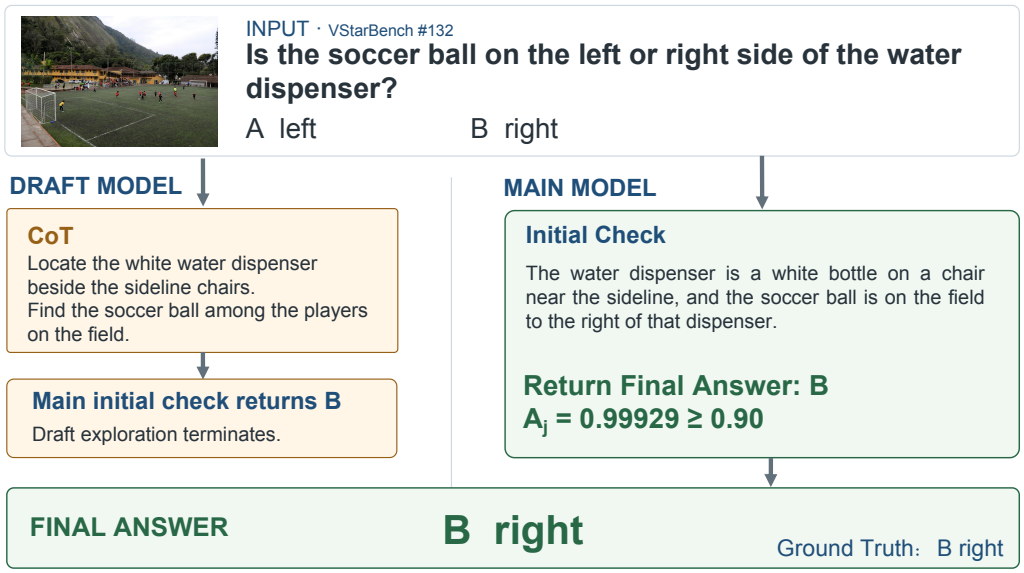


Figure 5: **Fast answer through the initial check.** In sample #132 of VstarBench, draft model begins locating the water dispenser and soccer ball while Main checks the original image independently. Main model identifies the dispenser beside the sideline chairs and the ball to its right. It returns **B (right)** with an answer-support score of 0.99929. The accepted initial answer ends draft exploration without a tool call.

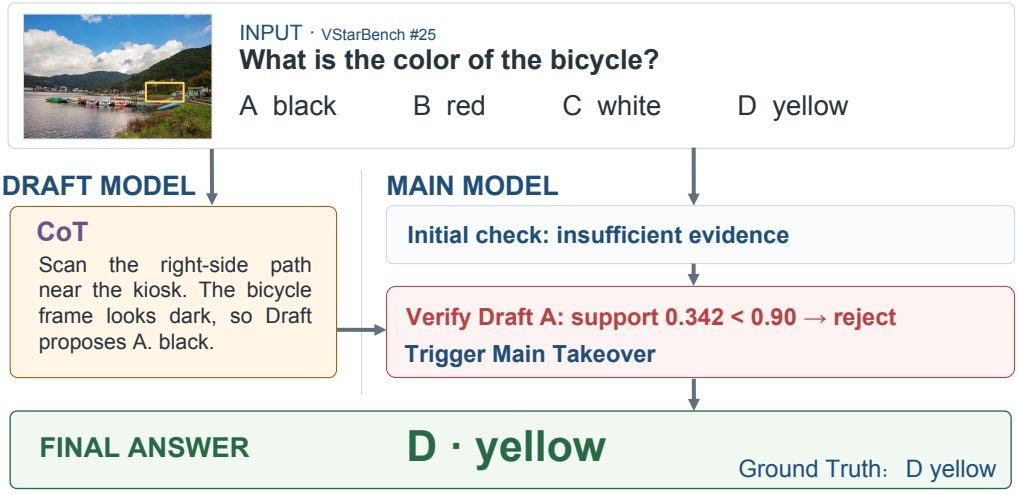


Figure 6: **Main takeover after Draft-answer rejection.** In sample #25 of VstarBench, the initial check is insufficient, and draft model proposes A (black) from the small bicycle near the kiosk without calling a tool. Main model assigns this candidate an evidence-support score of 0.342, below the 0.90 threshold, and rejects it. Main model then restarts from the original image in a fresh context and sandbox. In the recorded continuation, it makes three crop calls; its final reasoning relies on the second crop and returns **D (yellow)**. The overview figure shows the takeover trigger and final answer, rather than these subsequent crop calls.

API nproc	Accuracy (%)	Valid (%)	E2E (s)	Speedup
1	95.29	99.48	24.92	**2.12**×
2	95.29	98.95	32.29	1.84×
4	**95.81**	98.95	55.01	1.66×
8	94.24	95.81	116.04	1.08×
16	83.25	84.29	152.47	1.33×

Table 4: **Sensitivity to API concurrency on VStar (191 samples).** We vary EvalKit's API `nproc` (Duan et al., 2024), rather than vLLM's `max-num-seqs`. Accuracy and valid-output rate are computed over all 191 samples; requests that time out while awaiting vLLM responses are marked invalid. E2E is mean per-sample wall-clock time, including the elapsed time of timed-out samples. Speedup is relative to the full agent at the same API concurrency.

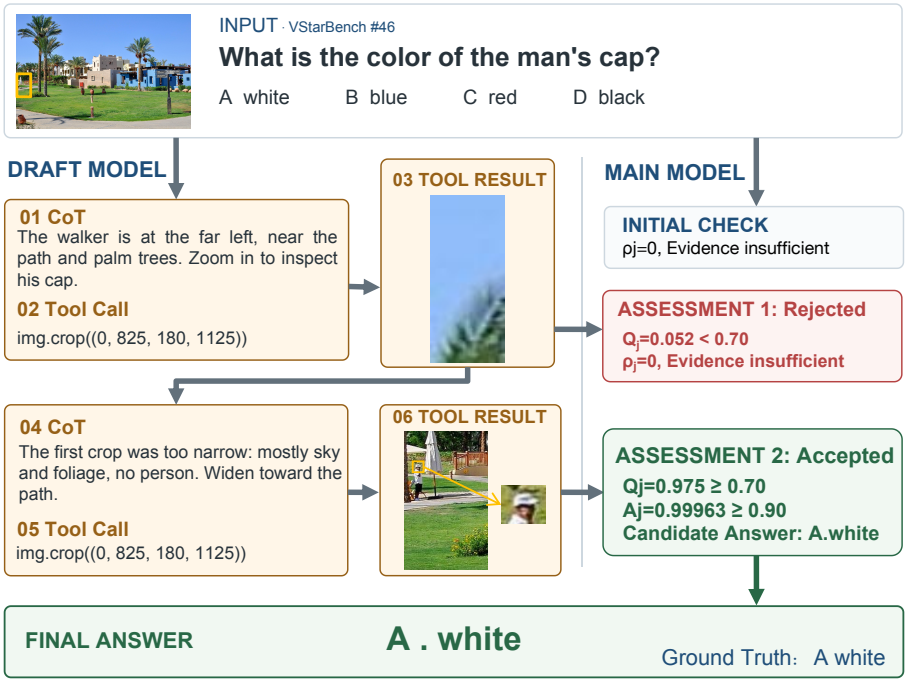


Figure 7: **Answer from Draft tool evidence.** In sample #46 of VstarBench, Main model's initial check finds insufficient evidence to determine the man's cap color. Draft model's first crop misses the person and contains mostly sky and foliage; Main model rejects this tool evidence with a quality score of 0.052, below the 0.70 threshold. Draft model then widens the crop toward the path. The second crop shows the walker's white cap, and main model accepts the evidence with a quality score of 0.975 and an answer-support score of 0.99963. Main model returns **A (white)** before draft model produces a final answer.

G Additional Experiment Results

G.1 Effect of API Concurrency

Experimental setup. We evaluate how SpecAgent behaves as more samples are processed simultaneously under the same hardware configuration. On VStar, we vary EvalKit's `api-nproc` over $\{1, 2, 4, 8, 16\}$, with $\tau_Q = 0.7$ and $\tau_A = 0.9$. This parameter controls sample-level concurrency, rather than the number of sequences scheduled internally by vLLM. At each concurrency level, we compare against the full Qwen3.5-27B agent running at the same API concurrency. Table 4 reports

accuracy, valid-output rate, mean end-to-end latency, and speedup over the corresponding baseline. Requests that time out while awaiting vLLM responses are marked invalid. Timed-out samples remain included in the latency calculation, with their elapsed time measured until failure. Thus, the reported latency characterizes all evaluated samples rather than only successful completions.

Latency gains narrow as concurrency increases. SpecAgent provides substantial acceleration at concurrency levels 1–4, while maintaining accuracy between 95.29% and 95.81%. However, its relative advantage decreases as concurrency rises: speedup falls from $2.12\times$ at concurrency 1 to $1.84\times$ at concurrency 2 and $1.66\times$ at concurrency 4. At concurrency 8, mean per-sample latency reaches 116.04 seconds and speedup decreases to $1.08\times$, while accuracy and the valid-output rate fall to 94.24% and 95.81%, respectively. Because each speedup is measured against a baseline at the same concurrency, this trend reflects a narrowing relative latency advantage under increasing load. It also distinguishes concurrency across samples from asynchrony within a sample: overlapping draft exploration and main-model assessment does not eliminate their competition for shared serving resources.

Serving overload affects task completion. At concurrency 16, the valid-output rate drops to 84.29% because requests that exceed the vLLM response timeout are marked invalid. Overall accuracy falls to 83.25%, and mean end-to-end latency increases to 152.47 seconds. Serving logs provide evidence of increased request backlogs. At concurrency 8, the main-model waiting queue is nonempty in 17 of 284 status snapshots (6.0%), with a peak queue length of five, while no waiting requests are observed in the draft-model snapshots. At concurrency 16, these proportions increase to $171/196$ (87.2%) for the main model and $43/176$ (24.4%) for the draft model, with peak queue lengths of 13 and 7, respectively. These percentages describe the fraction of recorded server snapshots with a nonempty queue, not the fraction of requests that experience waiting. Together, the queue observations and confirmed response timeouts indicate substantial serving pressure under the current configuration. The resulting accuracy degradation therefore reflects operational failures as well as any errors in returned answers.

Implications for asynchronous execution. These results show that the latency benefit of asynchronous exploration depends on the serving system's ability to process assessment requests promptly. Main-model assessment is nonblocking with respect to draft exploration, but it still consumes inference capacity. As concurrent samples generate competing requests, queueing and longer response times can erode the benefit of overlap and eventually cause task-completion failures.

Although speedup increases from $1.08\times$ at concurrency 8 to $1.33\times$ at concurrency 16, neither absolute latency nor the valid-output rate improves. Since timed-out samples are included, this ratio measures relative time to termination, including unsuccessful termination, rather than time to a successful answer. High-load efficiency should therefore be assessed jointly with completion reliability.

Algorithm 1: SpecAgent

Require: $x, D, M, \mathcal{T}$; budgets B_D, B_M; thresholds τ_Q, τ_A; failure limits B_Q, B_E
Ensure: Answer y or failure

$\mathcal{P}, C \leftarrow \varnothing$; $y \leftarrow \perp$; failure counts $\leftarrow 0$
▷ **Exploration and initial assessment**
Start D to execute tools and populate $\mathcal{P}$
M assesses $x \rightarrow (\hat{y}_0, u_0)$ ▷ alongside D
if $u_0 = 1$ **then** $y \leftarrow \hat{y}_0$
▷ **Online assessment and takeover**
while $y = \perp$ **do**
  if a new entry e_j is available **then**
    if e_j has no tool error **then**
      M assesses (x, e_j) ▷ dual assessment yielding $v_j, (\hat{y}_j, u_j)$ in parallel
      Store v_j; update quality-failure count
      if $u_j = v_j = 1$ **then** $y \leftarrow \hat{y}_j$; **break**
    end if
    Update tool-error count ▷ reset on success
    if quality/error count reaches B_Q/B_E **then**
      $C \leftarrow$ approved complete earlier turns
      break ▷ takeover
    end if
  else if draft exploration ends **then**
▷ **Terminal review**
    if a draft candidate y_D is available **then**
      M verifies $(x, \mathcal{P}, y_D) \rightarrow A_D$
      if $A_D > \tau_A$ **then** $y \leftarrow y_D$
    end if
    $C \leftarrow \mathcal{P}$; **break** ▷ reference context
  end if
end while
▷ **Answering and context reuse**
Request draft cancellation
if $y = \perp$ **then** M explores from (x, C) within B_M to obtain y ▷ reuse context
Wait for outstanding draft work; **return** y