Beyond Seeing: Evaluating Multimodal LLMs on Tool-Enabled Image Perception, Transformation, and Reasoning

Xingang Guo[1,2], Utkarsh Tyagi[1], Advait Gosai[1], Paula Vergara[1], Jayeon Park[1], Ernesto Gabriel Hernández Montoya[1], Chen Bo Calvin Zhang[1], Bin Hu[2], Yunzhong He[1], Bing Liu[1], Rakshith Sharma Srinivasa[1]

[1]Scale AI, [2]University of Illinois Urbana-Champaign

Correspondence: xingang.guo@scale.com

Abstract

Think-with-images, where multimodal LLMs (MLLMs) actively manipulate visual inputs as part of their reasoning, has become a focal research direction for agentic multimodal systems, with a surge of prompting, supervised, and RL methods teaching models to invoke vision tools such as cropping, zooming, and editing. Yet most established multimodal benchmarks use images whose critical content is already easy to perceive in a single pass, and many have saturated on frontier models. We introduce VisualToolBench (VTB), a benchmark deliberately curated so that critical visual content is dense, occluded, dispersed, rotated, or low-contrast, making active image manipulation a prerequisite for success. VTB contains 1,204 expert-authored, open-ended tasks (603 single-turn, 601 multi-turn) across five domains. Each task is paired with rubrics tagged by five competency categories (Visual Understanding, Truthfulness, Instruction Following, Reasoning, and Writing Style) that enable per-axis disentanglement. Across 26 evaluated MLLMs, even the strongest model reaches only 29% accuracy, with Visual Understanding the consistent bottleneck and visual misinterpretation accounting for 32% to 39% of failures. Top models, moreover, converge to comparable accuracy through markedly different tool-use strategies, which we analyze in detail.

1 Introduction

Thinking-with-images has emerged as a focal research direction for multimodal LLMs (MLLMs) towards agentic era (Qwen Team, 2026; Team et al., 2026; OpenAI, 2025; Su et al., 2025c). Rather than encoding an image once and reasoning from a fixed representation, this paradigm equips models with vision tools, cropping, zooming, rotation, contrast adjustment, drawing, and more, and lets them actively manipulate the image during inference as part of the reasoning trace. The past year has seen a rapid surge of methods aimed at teaching MLLMs to do this well, including prompting-based pipelines (Zeng et al., 2022; Hu et al., 2022; Yang et al., 2023b; Shen et al., 2024; Wang et al., 2025b), supervised fine-tuning on tool-invocation trajectories (Liu et al., 2023; Ma et al., 2024b; Qi et al., 2024; Wang et al., 2025a), and reinforcement learning that incentivizes adaptive visual exploration (Zheng et al., 2025; Su et al., 2025a,b; Zhu et al., 2025; Huang et al., 2025; Zhang et al., 2025d,e; Ma et al., 2025; Liu et al., 2025c).

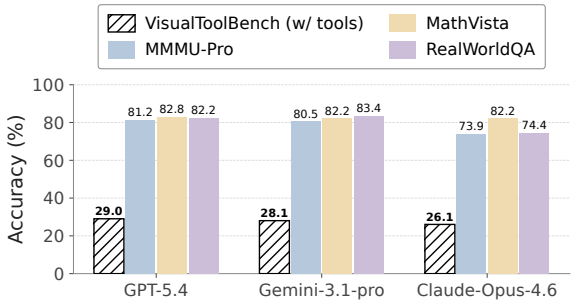


Figure 1: Accuracy of MLLMs across benchmarks.

However, evaluation has not kept pace with this paradigm shift. Most established multimodal benchmarks such as MMMU (Yue et al., 2024), MathVista (Lu et al., 2023), RealWorldQA (x.AI, 2024), ChartQA-style chart understanding (Wang et al., 2024b), and many of their successors evaluate MLLMs on images whose critical content is already *easy to perceive*: centered, well-lit, high-resolution, and large enough that a single forward pass extracts it without any active manipulation. Frontier closed-source models have either saturated or are rapidly approaching ceiling performance on many of these benchmarks as shown in Figure 1, leaving little headroom to differentiate between models or to detect whether a new think-with-images training recipe is actually helping. Even benchmarks that explicitly allow tool use such as GTA (Wang et al., 2024a) or m&m's (Ma

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 38981–39048
October 24-29, 2026 ©2026 Association for Computational Linguistics

et al., 2024a) restrict the available vision operations to a few canonical actions (mostly cropping) and use images that are not perception-hard in the first place, so the tool calls become decorative rather than load-bearing. As a result, we cannot reliably tell whether a model has learned to think with images, or is simply succeeding because there is nothing to actively perceive.

We propose VISUALTOOLBENCH (VTB), a benchmark designed from the ground up to fill this evaluation gap. Unlike existing multimodal benchmarks, where the relevant visual content is typically directly readable from the input image, VTB tasks are deliberately curated so that the critical visual content is *not easily perceived in a single pass*: it may be small relative to the overall image, occluded, dispersed across multiple regions of interest, rotated or flipped, low-contrast, or buried inside a busy scene. Solving a VTB task therefore requires the model to actively manipulate the image (crop the right region, zoom in, rotate, enhance, or compose several such operations) and integrate the resulting visual evidence with general-purpose tools such as web search, a Python interpreter, and a calculator[1].

To reflect real-world applications, we design five complementary task categories that capture diverse aspects of model performance. Two cases are tailored to single-turn tasks and three targeting multi-turn tasks (Sec.2.1). Each task is authored by qualified contributors with proper training and undergoes multiple review stages to ensure high-quality data samples (Sec.2.2). To further capture complex and realistic scenarios, tasks are presented in an open-ended format and carry rubric criteria tagged with the following five categories: Visual Understanding, Truthfulness, Instruction Following, Reasoning, and Writing Style. (Sec.2.3). Finally, due to the think-with-image nature of VTB, our evaluations support a specified vision tool for image manipulations and re-ingesting processed images into model's reasoning process, and a set of other general-purpose tools: web search, Python interpreter, calculator, and historical weather lookup, aiding in retrieval and computation for more advanced tasks (Sec.2.4). By releasing VTB and its accompanying agentic evaluation toolkit, we aim to catalyze the development of MLLMs that integrate perception, tool use, and reasoning into a unified competency stack. Our contributions are four-fold:

[1]More benchmark images are provided in Appendix E.1

STATISTIC	NUMBER
TOTAL QUESTIONS	1,204
- STEM	238 (19.7%)
- MEDICINE	238 (19.7%)
- FINANCE	243 (20.2%)
- SPORTS	240 (20.0%)
- GENERALIST	245 (20.4%)
SINGLE-TURN	603 (50.1%)
- REGION SWITCH Q&A	281 (46.6%)
- HYBRID TOOL-USE	322 (53.4%)
MULTI-TURN	601 (49.9%)
- FOLLOW-UP TEST	198 (32.9%)
- TEMPORAL REASONING	205 (34.2%)
- PROGRESSIVE REASONING	198 (32.9%)
TOTAL NUMBER OF RUBRICS	7,777
TOTAL NUMBER OF IMAGES	2,893
AVERAGE PROMPT LENGTH	48.41
AVERAGE ANSWER LENGTH	128.93

Table 1: Key statistics of VISUALTOOLBENCH.

1. The first think-with-image multimodal benchmark. VTB is the first benchmark to systematically evaluate MLLMs on tasks that require active visual manipulations.

2. Rubric-based, multi-dimensional evaluation. Moving beyond binary correctness or exact string matching, we design detailed rubrics annotated with five competency categories, enabling per-axis disentanglement of model strengths and weaknesses on the same task set.

3. Large-scale and systematic evaluations. We evaluate 26 representative MLLMs with function-calling capabilities, spanning reasoning and non-reasoning, open- and closed-source models under a consistent agentic loop that preserves complete tool-use trajectories. All evaluated models score below 30% overall pass rate, revealing substantial headroom.

4. In-depth experimental analysis. We profile each model along three axes and provide our insights: (i) *failure modes*: most failures stem from visual misinterpretation, not reasoning or coding; (ii) *tool-use behavior*: top models reach comparable accuracy through different tool-call strategies; and (iii) *ablations*: tool access consistently helps MLLMs.

2 VisualToolBench

In this section, we present VISUALTOOLBENCH (VTB), a multi-domain visual tool-use benchmark designed to evaluate MLLMs on challenging, real-world reasoning tasks. The benchmark includes

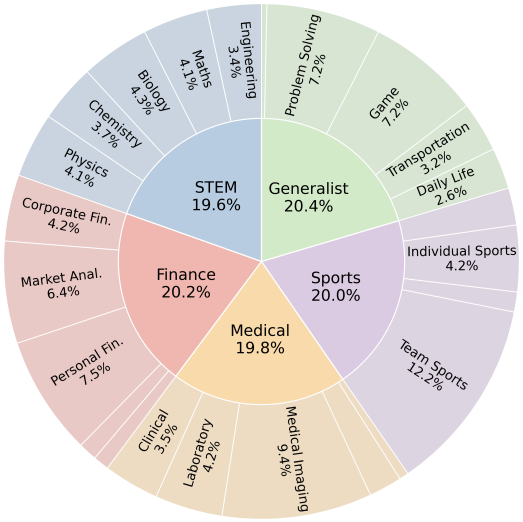


Figure 2: Topic distribution.

both single-turn and multi-turn tasks and incorporates five complementary task categories to probe different aspects of MLLM capabilities. Each task is designed by human experts and in an open-ended format to reflect realistic scenarios. In addition, each task is accompanied by detailed set of rubrics to support systematic evaluation. Table 1 summarizes the key statistics of VTB, and Figure 2 illustrates the topic distribution of tasks. Specifically, we cover five domains in VTB: STEM, Finance, Medicial, Sports, and Generalist, along with five realistic task categories.

2.1 Task Category

We design five complementary task categories, each targeting a critical aspect of real-world use case. Together, they assess not only visual perception but also the efficiency of tool use and the depth of multimodal reasoning. These categories are designed to mirror practical user scenarios, requiring models to think with images rather than relying solely on static perception.

Region-Switch Q&A (Single-Turn). The model answers a reasoning question that draws on information from multiple spatially distinct regions of interest within a single image, testing spatial selectivity and accurate RoI extraction via tool use. Figure 3 shows an example.

Hybrid Tool Reasoning (Single-Turn). The model must combine vision-specific tools with general-purpose tools (e.g., calculator, Python interpreter, web search) to solve multi-step tasks, testing the ability to orchestrate heterogeneous tool outputs into a coherent reasoning chain.

Follow-up Test (Multi-Turn). The first-round query is intentionally underspecified or ambiguous; the model should ask clarifying questions before answering, testing conversational proactivity and uncertainty management.

Temporal Visual Reasoning (Multi-Turn). The model reasons over a sequence of images across turns to detect temporal changes, track motion, or infer causal relationships.

Progressive Visual Reasoning (Multi-Turn). The model answers a series of interdependent questions about the same image, where later queries build on earlier answers, testing long-horizon reasoning and contextual memory. More benchmark examples are provided in Appendix G.

2.2 Task Generation Process

VTB tasks are authored by human experts through a multi-stage pipeline designed to ensure high-quality, realistic challenges. Specifically, contributors provide task prompts, images, golden answers, and evaluation rubrics. To maintain benchmark difficulty, we employ a model-in-the-loop filtering process: tasks are only included if at least two out of three representative models (o3, Gemini-2.5-pro, and o4-mini) fail to solve them with external tools. Each submission further undergoes two independent rounds of peer review to verify its necessity for dynamic image-based reasoning and overall correctness. Detailed descriptions of each pipeline stage are provided in Appendix E.3.

2.3 Rubric-based Evaluations

We adopt rubric-based evaluation to capture nuanced aspects of model performance beyond binary correctness alone (Arora et al., 2025; Starace et al., 2025; Scale AI, 2025; Lin et al., 2024; Sirdeshmukh et al., 2025; Fast et al., 2024; Gunjal et al., 2025; Guo et al., 2025). While designing each task, contributor also provides a comprehensive set of objective, atomic rubric criteria to assess model responses. A rubric item may range from specific factual requirements (e.g., providing the correct final short answer) to broader aspects of desirable behavior (e.g., presenting key intermediate steps). We categorize rubric items into the following five main categories for VTB:

1. Visual Understanding: Correct identification, extraction, and explanation of relevant visual elements such as text or objects;

2. Truthfulness: Accuracy of all factual statements and correctness of the final answer;

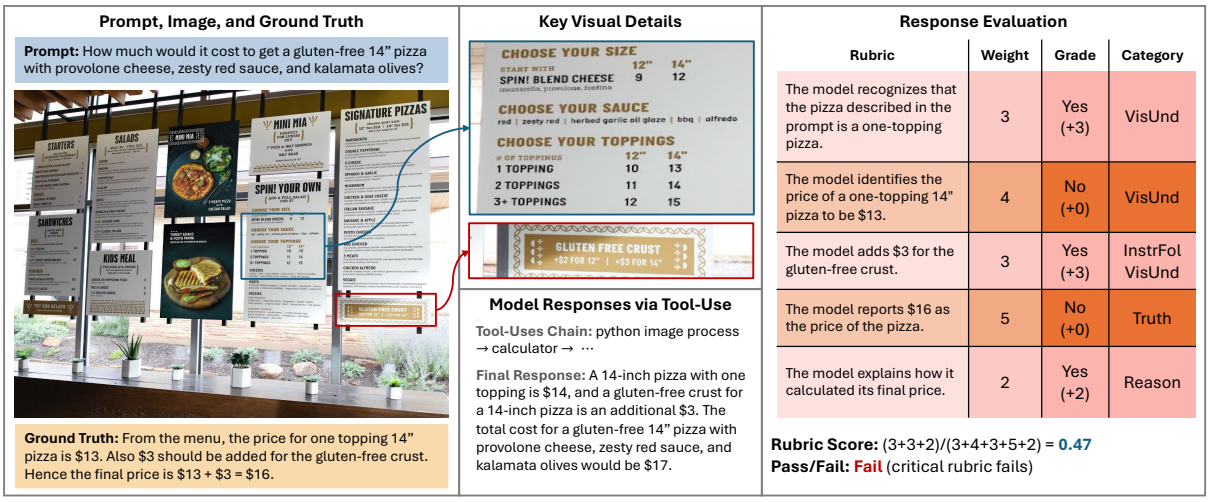


Figure 3: Demonstration example from VTB (single-turn, generalist domain, region switch Q&A). The key visual content needed to solve the task is distributed across different regions of the image, requiring the model to crop multiple regions for accurate perception and reasoning. Each task is paired with a detailed set of rubrics to evaluate model's responses. From these rubrics, we derive both a weighted rubric score between 0 and 1 and a binary pass/fail outcome, depending on whether critical rubrics are satisfied.

3. Instruction Following: Precise adherence to the task requirements specified in the prompt;
4. Reasoning: Use of clear, step-by-step logic with justified inferences and calculations;
5. Writing Style: Clarity, coherence, structure, and appropriate formatting of the response.

Each rubric criterion is assigned a weight $w \in \{1, 2, 3, 4, 5\}$ with higher weights indicate greater importance. Following by previous research (Arora et al., 2025), we utilize LLM-as-judge to evaluate model outputs by checking them against each rubric criterion. The weighted rubric score for a task is calculated as the sum of satisfied items, normalized by the total rubric weights. Rubric items with $w \geq 4$ are designated as **critical rubrics**. These rubrics typically correspond to essential aspects such as truthfulness and key visual understanding, and satisfying them indicates that the model has solved the task in a substantive way. Failure to meet any critical rubric results in an overall failure for that task, and this binary outcome is used to compute a task-level accuracy in VTB. All rubrics together contribute to the weighted rubrics score, enabling more fine-grained analysis.

2.4 Tool Set

To enable broad tool-augmented reasoning under the think-with-images paradigm, we expose six tools: python-image-processing, python-interpreter, web-search, calculator, browser-get-page-text, and historical-weather. This compact yet diverse set spans the core capabilities of image manipulation, computation, and information retrieval. The vision-specific python-image-processing tool is the cornerstone: it accepts arbitrary Python code for cropping, rotation, brightness/contrast adjustment, and other transformations, letting the model use the image as an interactive scratchpad.

3 Experiment Results

Implementation Details. We conduct large-scale evaluations using the LiteLLM function-calling interface (LiteLLM). Each supported tool is registered as a JSON-schema function, providing the model with a clear specification of tool signatures and required arguments within the system context. During inference, models follow an **agentic loop**: they evaluate the history messages and choose to either emit a tool call or provide a final response. Tool calls are executed in a secure Python sandbox, with the resulting observations (transformed images for vision tools) appended to the model's history message to inform its next action. This iterative process continues until the model terminates the loop (i.e., there is no more tool call) or reaches a maximum of 20 interaction rounds, ensuring a balance between thorough reasoning and computational efficiency.

Baseline Models. We evaluated 26 representative MLLMs with function-calling capabilities, covering both reasoning and non-reasoning as well as open- and closed-source models. A complete list

Model	Overall	Single-T	Multi-T	VisUnd	Truth	InstrFol	Reason	Present	STEM	Medicine	Finance	Sports	Gen
Closed-Source Models													
Gemini-3.1-pro	28.05	39.14	16.97	56.58	58.33	60.04	59.88	89.08	30.98	25.45	**36.50**	24.27	23.13
Gemini-3-pro	26.06	36.98	15.14	53.98	55.32	57.33	54.70	77.66	30.57	20.45	32.35	24.67	22.33
Gemini-3-flash	28.17	39.40	16.93	**59.68**	**60.35**	61.86	58.95	**92.02**	**33.69**	26.58	32.05	23.10	25.70
Gemini-2.5-pro	10.61	16.58	4.64	35.65	38.33	42.65	38.73	83.54	11.56	11.42	9.83	9.77	10.51
Gemini-2.5-flash	4.40	6.47	2.34	18.91	22.83	25.55	22.39	56.67	4.21	5.09	2.03	6.41	4.61
GPT-5.4-think	**29.00**	**39.57**	18.43	57.64	60.30	**64.39**	**61.39**	90.24	31.32	22.97	34.44	25.26	**30.70**
GPT-5.4	12.20	17.58	6.82	36.66	37.96	43.11	40.80	80.44	17.40	11.21	11.48	10.46	10.55
GPT-5-think	18.43	26.04	10.82	43.04	45.05	48.49	46.30	73.79	20.04	16.24	19.28	17.56	19.05
GPT-5	16.41	23.88	8.93	42.82	45.25	48.91	46.99	79.14	21.98	14.23	17.80	12.63	15.46
o3	11.95	15.59	8.32	36.25	38.02	41.29	39.95	71.22	14.38	14.60	10.27	12.54	8.13
o4-mini	11.04	15.75	6.32	37.04	38.00	42.99	40.70	85.96	8.95	12.98	9.83	11.30	12.17
GPT-4.1	5.31	5.97	4.66	28.81	28.54	35.50	32.61	80.28	5.89	8.79	1.65	6.68	3.68
Claude-opus-4.6-think	28.97	38.64	**19.30**	58.81	59.13	60.93	59.25	81.20	31.76	**29.22**	33.19	**25.88**	24.78
Claude-opus-4.5-think	20.92	28.52	13.31	49.35	51.13	54.77	51.27	83.14	21.19	22.14	23.76	20.89	16.64
Claude-opus-4.1-think	4.59	7.30	1.89	21.72	23.21	27.09	25.18	66.85	2.97	5.40	3.68	7.11	3.70
Claude-opus-4.1	4.32	6.30	2.33	25.64	26.98	31.40	29.31	72.40	4.66	7.11	1.63	4.60	3.64
Claude-sonnet-4.5-think	5.50	8.68	2.33	26.90	28.28	33.45	31.16	65.12	5.56	8.20	3.67	3.10	6.73
Claude-sonnet-4.5	4.86	7.40	2.33	27.60	29.06	34.35	31.78	67.53	3.86	6.78	2.69	5.45	5.17
Claude-sonnet-4-think	3.73	5.47	2.00	24.03	24.81	30.47	29.04	70.09	1.70	4.60	3.28	4.61	4.46
Claude-sonnet-4	4.07	5.31	2.83	22.76	24.64	30.10	27.50	73.57	3.79	6.27	2.88	4.18	3.24
Grok-4.20-reasoning	5.81	8.13	3.49	28.72	29.08	34.69	32.55	73.62	7.59	7.55	2.46	7.13	4.45
Nova-Premier	1.84	2.65	1.03	18.36	20.00	25.72	22.01	78.42	3.43	3.38	0.00	1.67	0.81
Open-Source Models													
Qwen3-VL-235B-think	3.15	4.81	1.50	22.48	20.33	22.69	22.24	20.97	5.07	5.00	2.86	0.84	2.02
Qwen3-VL-30B-think	0.93	1.33	0.53	15.07	12.28	11.94	12.44	6.81	1.25	1.24	0.96	0.42	0.81
Llama4-Maverick	1.16	1.82	0.50	17.26	16.99	21.59	19.55	28.75	2.16	1.25	0.81	0.83	0.82
Llama4-Scout	1.29	2.16	0.43	14.97	15.78	20.61	18.96	43.03	0.86	2.40	0.41	1.68	1.21

Table 2: Main results on VISUALTOOLBENCH. We report ACC (%) for overall, single-turn, multi-turn, and per-domain-wise results, and report CPR (%) for per-rubric-category weighted satisfaction rate. **VisUnd** (Visual Understanding), **Truth** (Truthfulness), **InstrFol** (Instruction Following), **Reason** (Reasoning), and **Present** (Writing Style). The best result in each column is in **bold**.

of models, along with their endpoints and configuration details, is provided in Appendix D.1.

Evaluation Metrics. We assess final-answer quality with three metrics: Accuracy (ACC), Average Rubric Score (ARS), and Category Pass Rate (CPR), the per-rubric-category weighted satisfaction rate (i.e., for each rubric category, the fraction of category-specific rubric weight a model's response satisfies, averaged across tasks). To characterize tool-use behavior, we report three metrics: Tool-Call Proactivity (TCP), the fraction of tasks on which the model invokes at least one tool; Tool-Call Success Rate (TCSR), the fraction of tool calls that execute successfully; and Tool-Call Volume (TCV), the average number of tool calls per task. Formal definitions are deferred to Appendix B.

3.1 Model Performance

Table 2 reports each model's overall ACC, single/multi-turn ACC, per-rubric-category CPR, and per-domain ACC. We make the following key observations.

Observation 1. VISUALTOOLBENCH is highly challenging: no model crosses 30% ACC, and 14 of 26 evaluated MLLMs score below 10%.

Even the strongest model (GPT-5.4-think) reaches only 29.0% ACC, underscoring that current MLLMs still struggle on visual reasoning tasks where the critical content cannot be perceived in a single forward pass and must be extracted through active tool use. Figure 4 shows the promising year-over-year gains within each frontier family: Gemini +17 pts, OpenAI +24 pts, and Anthropic +22 pts. Yet no released model has crossed the 30% threshold, leaving substantial headroom for future think-with-images systems.

Observation 2. Multi-turn tasks are markedly harder than single-turn tasks for every model family.

Across every closed-source family, multi-turn ACC sits roughly 2–3× below single-turn ACC. Figure 9 (Appendix B.2) makes this gap explicit: single-turn formats (region-switch Q&A, hybrid-tool reasoning) routinely outperform multi-turn formats (follow-up test, temporal sequential reasoning, pro-

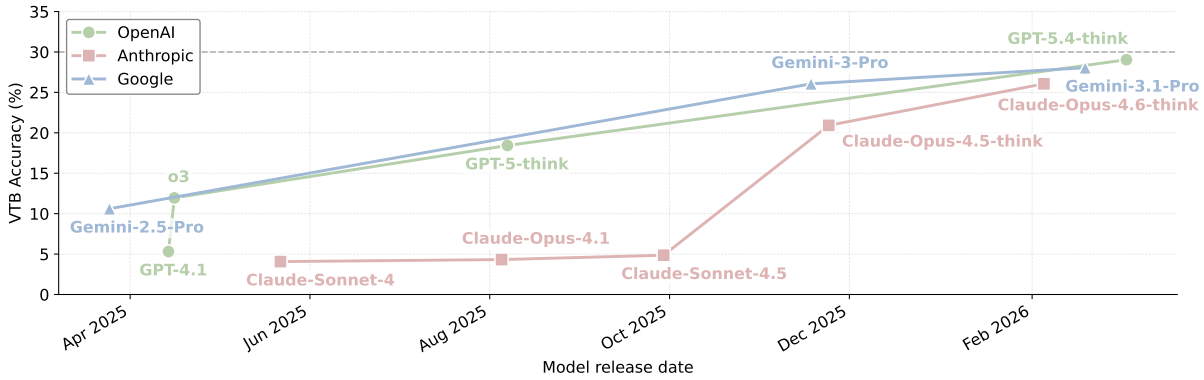


Figure 4: VTB Overall ACC over model release time. Each line connects one model family's chronologically ordered releases; the dashed reference line marks 30%. Every frontier family has improved rapidly (Gemini +17.4, OpenAI +23.7, Anthropic +22.0 pts in ∼12 months), yet no released model has crossed the 30% threshold.

gressive visual reasoning). This is expected as multi-turn tasks involve multiple conversational turns, compounding reasoning errors and multiplying opportunities for tool-use mistakes.

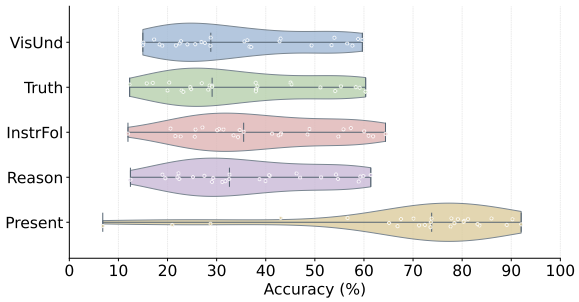


Figure 5: Per-rubric-category accuracy across the 26 evaluated MLLMs. Each violin shows the distribution of category-level CPR.

Observation 3. Per-category accuracy is widely spread ($\sigma \approx 14$–15) with Visual Understanding consistently the lowest. VTB discriminates models cleanly on every competency axis.

Figure 5 reports the distribution of per-category accuracy across the 26 models. The three hardest categories: Visual Understanding (median 28.8%), Truthfulness (29.1%), and Reasoning (32.6%), cluster within roughly 4 points; Instruction Following (35.5%) is slightly easier, and Writing Style (73.8%) is by far the easiest to satisfy. More importantly, every primary category exhibits a wide model-to-model spread ($\sigma \approx 14$–15), with the strongest models reaching $\approx 60\%$ CPR and the weakest near 12%. This large variance indicates that VTB discriminates MLLMs on every competency axis, while leaving substantial headroom for future improvements.

3.2 Tool-Use Analysis

We now turn from *what* the top models score to *how* they use tools to get there. We focus on the three highest-performing models (GPT-5.4-think, Gemini-3.1-Pro, Claude-Opus-4.6-think) and analyze tool-use at two levels of granularity: aggregate call statistics (Table 3) and the distribution of image-manipulation operations the models invoke (Figure 6).

Model	TCP (%)	TCSR (%)	TCV
GPT-5.4-think	99.3	83.8	6.36
Gemini-3.1-Pro	66.1	**98.1**	1.80
Claude-Opus-4.6-think	**99.8**	97.2	**7.91**

Table 3: Tool-call metrics for the three top models, aggregated across all 1,204 tasks (single + multi-turn).

Top models converge on accuracy but diverge sharply on tool-use style. Table 3 reveals three distinct strategies. Claude-Opus-4.6-think and GPT-5.4-think call tools on essentially every task (TCP $\geq$ 99%) and call them often (7.9 and 6.4 calls per task, respectively), whereas Gemini-3.1-Pro is markedly more selective (TCP 66.1%, only 1.8 calls per task). Execution quality also differs: Gemini and Claude succeed on almost every call they make (TCSR $\geq$ 97%), while GPT-5.4-think hits a noticeably higher failure rate (TCSR 83.8%), driven mostly by infrastructure-level errors (e.g., payload-too-large from large image outputs) rather than code bugs. These divergent profiles likely reflect different training emphases: Gemini-3.1-Pro appears tuned to invoke tools only when necessary, while GPT-5.4-think and Claude-Opus-4.6-think appear tuned to be highly proactive callers that

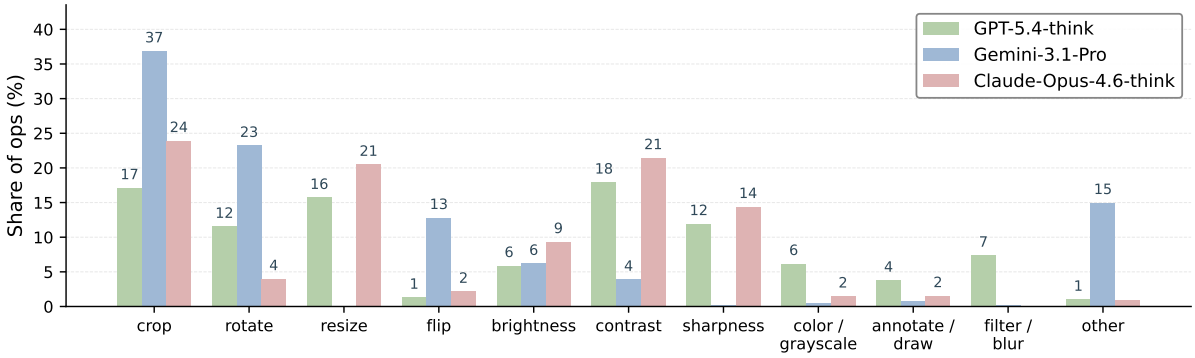


Figure 6: Distribution of image-manipulation operations across the three top models.

default to tool use across nearly all tasks.

Gemini's less tool use reflects stronger native vision. To understand why Gemini models invoke tools so much less frequently than GPT and Claude, we disabled tool access and compared per-rubric-category accuracy across families. Without any tools, Gemini-3-pro reaches 49.59% Visual Understanding CPR, compared to at most 26.73% for the OpenAI models and 20.33% for the Anthropic models we tested. This shows that Gemini's lower TCP and TVP are not a deficit in tool orchestration but a consequence of stronger native visual grounding. This finding also points to a future direction for think-with-image benchmark designs: as native vision continues to improve, perception-focused evaluations will saturate, and tool-use benchmarks must move toward challenges that perception alone cannot solve, such as tasks requiring *active image editing* or *iterative visual refinement*. Nevertheless, VTB still poses a significant challenge for the current frontier, no model crosses 30% ACC, leaving substantial headroom for the next generation of think-with-image MLLMs.

Models exhibit diverse patterns of image manipulation. We do not constrain models to a predefined operation set; the vision tool accepts arbitrary Python, and we recover operations *post hoc* by parsing the code into ten categories (cropping, resizing, rotation, flipping, brightness, contrast, sharpness, color/grayscale, annotation/drawing, filter/blur), plus a catch-all *other* (Figure 6). Gemini-3.1-Pro relies almost exclusively on geometric transforms (37% crop, 23% rotate, 13% flip), barely touching pixel-level enhancement, while GPT-5.4-think and Claude-Opus-4.6-think combine cropping with heavy contrast, sharpness, and resize use — they invest in enhancing image quality, not just framing the right region. This matches the aggregate metrics in Table 3: Gemini uses fewer calls *and* fewer operation types, whereas the other two pursue a broader, enhancement-heavy recipe (full per-model breakdown in Table 9, Appendix B.5).

3.3 Ablation Study

We conduct four targeted ablations to understand how (i) the tool-call budget, (ii) tool access itself, (iii) tool design (general vs. atomic), and (iv) prompting strategy shape model behavior on VTB.

3.3.1 Tool-Call Budget

We cap the agentic loop at a maximum of $\{5, 10, 15, 20\}$ tool-call rounds per task and rerun two representative closed-source models (GPT-5 and Gemini-3-pro) on the full benchmark. Table 4 reports overall ACC (averaged across single + multi-turn) at each budget.

Model	5	10	15	20
GPT-5	11.8	14.9	16.4	16.4
Gemini-3-pro	23.2	26.0	26.0	26.0

Table 4: Tool-call budget ablation.

Both models exhibit clear diminishing returns. Gemini-3-pro reaches 26.0% ACC at budget$= 10$, and GPT-5 gains only 0.5 points between budget 10 and budget 15. Our default budget of 20 rounds therefore provides sufficient tool call budget.

3.3.2 With Tools vs. Without Tools

We rerun five frontier models in a no-tool setting (i.e., the agentic loop is disabled and the model must answer directly from the original image). Figure 7 compares single-turn ACC with and without tools. Tools deliver the largest gain for GPT-5-think ($11.8\% \rightarrow 26.0\%$), with smaller but still positive

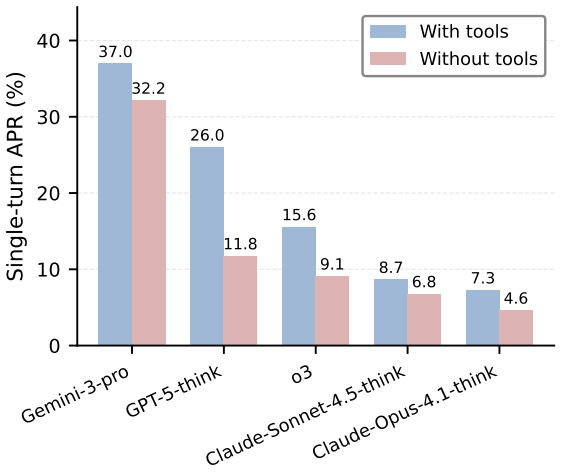


Figure 7: Single-turn ACC with vs. without tool access for five frontier models.

gains for o3 (+6.5 pts), Gemini-3-pro (+4.8 pts), and the Claude family (+1.9–2.7 pts).

3.3.3 Atomic vs. General Vision Tools

To probe whether raw coding ability is a confound, we compare our general python-image-processing tool (which accepts arbitrary Python) against a restricted set of *atomic* vision tools (crop, rotate, brightness, etc.). The atomic setup removes the need for code generation entirely; if coding skill were the dominant bottleneck on VTB, atomic tools should help.

As shown in Figure 8, atomic tools provide only a marginal change for GPT-4.1, indicating that its primary bottleneck is visual reasoning rather than code generation. GPT-5, however, suffers a notable drop (23.88% → 18.15%) when restricted to atomic tools. We attribute this to a loss of expressive flexibility: in the general setting GPT-5 routinely composes multi-step, creative image operations within a single call, and the atomic interface forces it into a rigid action vocabulary that limits this strategy. Further discussion is provided in Appendix F.2.

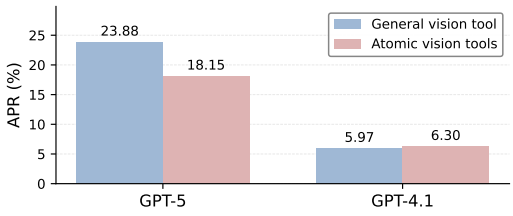


Figure 8: Atomic vs. general vision-tool comparison on the single-turn split.

3.3.4 Selective-Tool-Use Prompt

Finally, we probe a prompting intervention: we prepend the system prompt with an instruction asking the model to first decide whether tool use is necessary, and only invoke tools when the relevant visual evidence is not already readable. We compare this selective prompt against our default tool-use prompt on four frontier models. Table 5 reports single-turn ACC under both conditions.

Model	Default	Selective	Δ
Gemini-3-pro	36.98	35.38	−1.60
GPT-5	23.88	25.08	+1.20
o3	15.59	21.77	**+6.18**
Claude-Opus-4.1-think	7.30	7.77	+0.47

Table 5: ACC (%) on selective-tool-use prompt ablation.

Overall, we see no significant difference between default system prompt and selective prompt. Specifically. the selective prompt delivers a gain for o3 (+6.18 pts) and small positive gains for GPT-5 (+1.2) and Claude-Opus-4.1-think (+0.47), while slightly hurting Gemini-3-pro (−1.6). This pattern is consistent with the tool-call proactivity in Table 3: Gemini already invokes tools selectively by default (TCP 66.1%), so asking it to be more selective is counter-productive, whereas the OpenAI models are over-eager callers (TCP ≥ 99%) and clearly benefit from being nudged to skip tools when unnecessary. This result suggests training-time work on when to call tools is at least as important as work on how to call them.

3.4 Error Analysis

To understand why models fail, we conduct a fine-grained failure-mode analysis on failed tasks per model for GPT-5, Gemini-2.5-pro, and Claude-opus-4.1. We define eleven categories of vision tool-use error spanning visual misinterpretation, incorrect cropping, rotation, flipping, resizing, enhancing, or editing operations, OCR errors, distraction-induced mistakes, incorrect tool selection, tool-execution failures, and a catch-all Other. For each failed task we prompt GPT-5 with the question, the model's tool-use chain, the golden answer, and the model's actual answer to generate an initial category prediction; a human annotator then reviews and corrects each label. Table 6 shows three consistent patterns. First, visual misinterpretation is the dominant failure mode across all three models (32–39% of failures). Second, incorrect cropping and tool-execution errors together

account for another 20–30% of failures. Third, the three models share the broad shape of the failure distribution but differ in emphasis: Claude-opus-4.1 is the most perception-bottlenecked (39.46% misinterpretation + 17.94% incorrect cropping), GPT-5 shows the highest tool-execution error rate (15.14%). We provide detailed model failure examples in Appendix I with full model trajectories.

Failure Type	GPT	Gemini	Claude
Vis Misinterpret	**34.40**	**32.56**	**39.46**
Tool Execution	15.14	12.79	6.28
Incorrect Cropping	14.68	12.21	17.94
OCR	11.01	12.21	7.62
Distraction	8.26	4.07	10.76
Rotation / Flipping	5.05	5.23	5.38
Tool Selection	2.75	3.49	5.83
Enhancing	1.83	0.58	1.35
Resizing	0.46	0.00	0.00
Editing	0.00	0.00	0.45
Other	6.42	16.86	4.93
Total failures	218	172	223

Table 6: Fine-grained tool-use failure breakdown.

4 Related Work

Most existing multimodal benchmarks remain limited to *think about images*: passive visual Q&A without active interactions (Rahmanzadehgervi et al., 2024; Yue et al., 2024; Wang et al., 2024b; Lu et al., 2023; Zou et al., 2024; Wu and Xie, 2024; Scale AI, 2025), restricted-tool settings such as basic cropping (Wang et al., 2024a; Ma et al., 2024a), or tool-agnostic multi-turn dialogues (Liu et al., 2024a; Yan et al., 2025); see (Li et al., 2024) for a broader survey. Recent work teaches MLLMs to *think with images* via prompting (Zeng et al., 2022; Hu et al., 2022; Yang et al., 2023b; Wu et al., 2024a; Liu et al., 2025a; Shtedritski et al., 2023; Zhang et al., 2025b; Shen et al., 2024; Wang et al., 2025b; Yang et al., 2023a; Wu et al., 2024b; Li et al., 2025), supervised fine-tuning (Liu et al., 2023; Shao et al., 2024; Ma et al., 2024b; Bai et al., 2025b; Qi et al., 2024; Bai et al., 2025a; Chung et al., 2025), or reinforcement learning (Ma et al., 2025; Wang et al., 2025c; Liu et al., 2025c; Fan et al., 2025; Ni et al., 2025; Liu et al., 2025b; Huang et al., 2025; Zhu et al., 2025; Zheng et al., 2025; Hong et al., 2025; Su et al., 2025a; Wu et al., 2025; Su et al., 2025b). Despite these advances, evaluation remains anchored on visually passive benchmarks. VTB fills this gap with a rigorous testbed for genuine think-with-images intelligence; see Appendix A for further discussion.

5 Conclusion

We introduced VISUALTOOLBENCH, the first benchmark explicitly designed for the think-with-images paradigm, which requires *active visual manipulation* through tool use across diverse domains and task types, and is paired with an agentic interaction loop for systematic, reproducible evaluation. Our results show that current MLLMs still struggle with dynamic visual manipulation, exposing a substantial gap between passive understanding and interactive visual reasoning. We hope VTB serves as a catalyst for models that can more effectively think with images in complex real-world scenarios.

LLM Use Acknowledgment

We used large language models as a writing aid for prose polishing. All experimental results, analysis, and conclusions are produced by the authors.

Limitations

VISUALTOOLBENCH has several limitations that we want to make explicit. First, the task pool focuses on five English-language domains (STEM, Medicine, Finance, Sports, Generalist) and may under-represent other settings such as legal documents, music notation, scientific figures with specialized notation, or non-English prompts and signage. Second, our tool set is centered on python-image-processing: while this design choice gives models maximum expressive flexibility, benchmarks with more constrained or modality-diverse toolsets (e.g., dedicated OCR engines, 3D scene tools, video frames) may reveal different aspects of model behavior. Third, our rubric scoring relies on an LLM judge; although we use atomic, single-criterion prompts and validate agreement against human annotators, residual judge biases cannot be fully ruled out and may slightly distort absolute scores, even though they leave the overall ranking stable. Finally, our fine-grained failure-mode and tool-use analyses are necessarily conducted on a small set of top-performing models due to the cost of human-supervised labeling and per-call code inspection; broader replication across more open-source and frontier models is left to future work.

References

Rahul K Arora, Jason Wei, Rebecca Soskin Hicks, Preston Bowman, Joaquin Quiñonero-Candela, Foivos Tsimpourlas, Michael Sharman, Meghan Shah, Andrea Vallone, Alex Beutel, and 1 others. 2025. Healthbench: Evaluating large language models towards improved human health. *arXiv preprint arXiv:2505.08775*.

Sule Bai, Mingxing Li, Yong Liu, Jing Tang, Haoji Zhang, Lei Sun, Xiangxiang Chu, and Yansong Tang. 2025a. Univg-r1: Reasoning guided universal visual grounding with reinforcement learning. *arXiv preprint arXiv:2505.14231*.

Tianyi Bai, Zengjie Hu, Fupeng Sun, Jiantao Qiu, Yizhen Jiang, Guangxin He, Bohan Zeng, Conghui He, Binhang Yuan, and Wentao Zhang. 2025b. Multi-step visual reasoning with visual tokens scaling and verification. *arXiv preprint arXiv:2506.07235*.

Jing Bi, Junjia Guo, Susan Liang, Guangyu Sun, Luchuan Song, Yunlong Tang, Jinxi He, Jiarui Wu, Ali Vosoughi, Chen Chen, and 1 others. 2025. Verify: A benchmark of visual explanation and reasoning for investigating multimodal reasoning fidelity. *arXiv preprint arXiv:2503.11557*.

Zihui Cheng, Qiguang Chen, Xiao Xu, Jiaqi Wang, Weiyun Wang, Hao Fei, Yidong Wang, Alex Jinpeng Wang, Zhi Chen, Wanxiang Che, and 1 others. 2025. Visual thoughts: A unified perspective of understanding multimodal chain-of-thought. *arXiv preprint arXiv:2505.15510*.

Jiwan Chung, Junhyeok Kim, Siyeol Kim, Jaeyoung Lee, Min Soo Kim, and Youngjae Yu. 2025. Don't look only once: Towards multimodal interactive reasoning with selective visual revisitation. *arXiv preprint arXiv:2505.18842*.

Yue Fan, Xuehai He, Diji Yang, Kaizhi Zheng, Ching-Chen Kuo, Yuting Zheng, Sravana Jyothi Narayanaraju, Xinze Guan, and Xin Eric Wang. 2025. Grit: Teaching mllms to think with images. *arXiv preprint arXiv:2505.15879*.

Dennis Fast, Lisa C Adams, Felix Busch, Conor Fallon, Marc Huppertz, Robert Siepmann, Philipp Prucker, Nadine Bayerl, Daniel Truhn, Marcus Makowski, and 1 others. 2024. Autonomous medical evaluation for guideline adherence of large language models. *NPJ Digital Medicine*, 7(1):358.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Bing Liu, and Sean Hendryx. 2025. Rubrics as rewards: Reinforcement learning beyond verifiable domains. *arXiv preprint arXiv:2507.17746*.

Xingang Guo, Yaxin Li, Xiangyi Kong, Yilan Jiang, Xiayu Zhao, Zhihua Gong, Yufan Zhang, Daixuan Li, Tianle Sang, Beixiao Zhu, Gregory Jun, Yingbing Huang, Yiqi Liu, Yuqi Xue, Rahul Dev Kundu, Qi Jian Lim, Yizhou Zhao, Luke Alexander Granger, Mohamed Badr Younis, and 46 others. 2025. Toward engineering AGI: Benchmarking the engineering design capabilities of llms. In *Neural Information Processing Systems Datasets and Benchmarks Track*.

Jack Hong, Chenxiao Zhao, ChengLin Zhu, Weiheng Lu, Guohai Xu, and Xing Yu. 2025. Deepeyesv2: Toward agentic multimodal model. *arXiv preprint arXiv:2511.05271*.

Yushi Hu, Hang Hua, Zhengyuan Yang, Weijia Shi, Noah A Smith, and Jiebo Luo. 2022. Promptcap: Prompt-guided task-aware image captioning. *arXiv preprint arXiv:2211.09699*.

Zeyi Huang, Yuyang Ji, Anirudh Sundara Rajan, Zefan Cai, Wen Xiao, Haohan Wang, Junjie Hu, and Yong Jae Lee. 2025. Visualtoolagent (vista): A reinforcement learning framework for visual tool selection. *arXiv preprint arXiv:2505.20289*.

Geng Li, Jinglin Xu, Yunzhen Zhao, and Yuxin Peng. 2025. Dyfo: A training-free dynamic focus visual search for enhancing lmms in fine-grained visual understanding. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 9098–9108.

Lin Li, Guikun Chen, Hanrong Shi, Jun Xiao, and Long Chen. 2024. A survey on multimodal benchmarks: In the era of large ai models. *arXiv preprint arXiv:2409.18142*.

Bill Yuchen Lin, Yuntian Deng, Khyathi Chandu, Faeze Brahman, Abhilasha Ravichander, Valentina Pyatkin, Nouha Dziri, Ronan Le Bras, and Yejin Choi. 2024. Wildbench: Benchmarking llms with challenging tasks from real users in the wild. *arXiv preprint arXiv:2406.04770*.

LiteLLM. Litellm documentation: Function call. https://docs.litellm.ai/docs/completion/function_call. Accessed: 2025-09-17.

Dairu Liu, Ziyue Wang, Minyuan Ruan, Fuwen Luo, Chi Chen, Peng Li, and Yang Liu. 2025a. Visual abstract thinking empowers multimodal reasoning. *arXiv preprint arXiv:2505.20164*.

Shilong Liu, Hao Cheng, Haotian Liu, Hao Zhang, Feng Li, Tianhe Ren, Xueyan Zou, Jianwei Yang, Hang Su, Jun Zhu, Lei Zhang, Jianfeng Gao, and Chunyuan Li. 2023. Llava-plus: Learning to use tools for creating multimodal agents. *arXiv:2311.05437*.

Yuqi Liu, Bohao Peng, Zhisheng Zhong, Zihao Yue, Fanbin Lu, Bei Yu, and Jiaya Jia. 2025b. Seg-zero: Reasoning-chain guided segmentation via cognitive reinforcement. *arXiv preprint arXiv:2503.06520*.

Yuqi Liu, Tianyuan Qu, Zhisheng Zhong, Bohao Peng, Shu Liu, Bei Yu, and Jiaya Jia. 2025c. Visionreasoner: Unified visual perception and reasoning via reinforcement learning. *arXiv preprint arXiv:2505.12081*.

Ziyu Liu, Tao Chu, Yuhang Zang, Xilin Wei, Xiaoyi Dong, Pan Zhang, Zijian Liang, Yuanjun Xiong, Yu Qiao, Dahua Lin, and 1 others. 2024a. Mmdu: A multi-turn multi-image dialog understanding benchmark and instruction-tuning dataset for lvlms. *arXiv preprint arXiv:2406.11833*.

Zuyan Liu, Yuhao Dong, Yongming Rao, Jie Zhou, and Jiwen Lu. 2024b. Chain-of-spot: Interactive reasoning improves large vision-language models. *arXiv preprint arXiv:2403.12966*.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. 2023. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. *arXiv preprint arXiv:2310.02255*.

Yan Ma, Linge Du, Xuyang Shen, Shaoxiang Chen, Pengfei Li, Qibing Ren, Lizhuang Ma, Yuchao Dai, Pengfei Liu, and Junjie Yan. 2025. One rl to see them all: Visual triple unified reinforcement learning. *arXiv preprint arXiv:2505.18129*.

Zixian Ma, Weikai Huang, Jieyu Zhang, Tanmay Gupta, and Ranjay Krishna. 2024a. m & m's: A benchmark to evaluate tool-use for m ulti-step m ulti-modal tasks. In *European Conference on Computer Vision*, pages 18–34. Springer.

Zixian Ma, Jianguo Zhang, Zhiwei Liu, Jieyu Zhang, Juntao Tan, Manli Shu, Juan Carlos Niebles, Shelby Heinecke, Huan Wang, Caiming Xiong, Ranjay Krishna, and Silvio Savarese. 2024b. Taco: Learning multi-modal action models with synthetic chains-of-thought-and-action. *Preprint*, arXiv:2412.05479.

Minheng Ni, Zhengyuan Yang, Linjie Li, Chung-Ching Lin, Kevin Lin, Wangmeng Zuo, and Lijuan Wang. 2025. Point-rft: Improving multimodal reasoning with visually grounded reinforcement finetuning. *arXiv preprint arXiv:2505.19702*.

OpenAI. 2025. Thinking with images. https://openai.com/index/thinking-with-images/. Accessed: 2025-04-16.

Ji Qi, Ming Ding, Weihan Wang, Yushi Bai, Qingsong Lv, Wenyi Hong, Bin Xu, Lei Hou, Juanzi Li, Yuxiao Dong, and Jie Tang. 2024. Cogcom: Train large vision-language models diving into details through chain of manipulations. *arXiv preprint arXiv:2402.04236*.

Qwen Team. 2026. Qwen3.5: Scaling up, toward native omni-modal agi.

Pooyan Rahmanzadehgervi, Logan Bolton, Mohammad Reza Taesiri, and Anh Totti Nguyen. 2024. Vision language models are blind. In *Proceedings of the Asian Conference on Computer Vision*, pages 18–34.

Scale AI. 2025. Vista: Visual–language understanding leaderboard. https://scale.com/leaderboard/visual_language_understanding.

Hao Shao, Shengju Qian, Han Xiao, Guanglu Song, Zhuofan Zong, Letian Wang, Yu Liu, and Hongsheng Li. 2024. Visual cot: Unleashing chain-of-thought reasoning in multi-modal language models. *Preprint*, arXiv:2403.16999.

Haozhan Shen, Kangjia Zhao, Tiancheng Zhao, Ruochen Xu, Zilun Zhang, Mingwei Zhu, and Jianwei Yin. 2024. Zoomeye: Enhancing multimodal llms with human-like zooming capabilities through tree-based image exploration. *arXiv preprint arXiv:2411.16044*.

Aleksandar Shtedritski, Christian Rupprecht, and Andrea Vedaldi. 2023. What does clip know about a red circle? visual prompt engineering for vlms. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11987–11997.

Ved Sirdeshmukh, Kaustubh Deshpande, Johannes Mols, Lifeng Jin, Ed-Yeremai Cardona, Dean Lee, Jeremy Kritz, Willow Primack, Summer Yue, and Chen Xing. 2025. Multichallenge: A realistic multi-turn conversation evaluation benchmark challenging to frontier llms. *arXiv preprint arXiv:2501.17399*.

Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, and 1 others. 2025. Paperbench: Evaluating ai's ability to replicate ai research. *arXiv preprint arXiv:2504.01848*.

Alex Su, Haozhe Wang, Weiming Ren, Fangzhen Lin, and Wenhu Chen. 2025a. Pixel reasoner: Incentivizing pixel-space reasoning with curiosity-driven reinforcement learning. *arXiv preprint arXiv:2505.15966*.

Zhaochen Su, Linjie Li, Mingyang Song, Yunzhuo Hao, Zhengyuan Yang, Jun Zhang, Guanjie Chen, Jiawei Gu, Juntao Li, Xiaoye Qu, and 1 others. 2025b. Openthinkimg: Learning to think with images via visual tool reinforcement learning. *arXiv preprint arXiv:2505.08617*.

Zhaochen Su, Peng Xia, Hangyu Guo, Zhenhua Liu, Yan Ma, Xiaoye Qu, Jiaqi Liu, Yanshu Li, Kaide Zeng, Zhengyuan Yang, and 1 others. 2025c. Thinking with images for multimodal reasoning: Foundations, methods, and future frontiers. *arXiv preprint arXiv:2506.23918*.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, SH Cai, Yuan Cao, Y Charles, HS Che, Cheng Chen, Guanduo Chen, and 1 others. 2026. Kimi k2. 5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*.

Jiacong Wang, Zijiang Kang, Haochen Wang, Haiyong Jiang, Jiawen Li, Bohong Wu, Ya Wang, Jiao Ran, Xiao Liang, Chao Feng, and 1 others. 2025a. Vgr: Visual grounded reasoning. *arXiv preprint arXiv:2506.11991*.

Jize Wang, Ma Zerun, Yining Li, Songyang Zhang, Cailian Chen, Kai Chen, and Xinyi Le. 2024a. GTA: a benchmark for general tool agents. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Yikun Wang, Siyin Wang, Qinyuan Cheng, Zhaoye Fei, Liang Ding, Qipeng Guo, Dacheng Tao, and Xipeng Qiu. 2025b. Visuothink: Empowering lvlm reasoning with multimodal tree search. *arXiv preprint arXiv:2504.09130*.

Zifu Wang, Junyi Zhu, Bo Tang, Zhiyu Li, Feiyu Xiong, Jiaqian Yu, and Matthew B Blaschko. 2025c. Jigsaw-r1: A study of rule-based visual reinforcement learning with jigsaw puzzles. *arXiv preprint arXiv:2505.23590*.

Zirui Wang, Mengzhou Xia, Luxi He, Howard Chen, Yitao Liu, Richard Zhu, Kaiqu Liang, Xindi Wu, Haotian Liu, Sadhika Malladi, and 1 others. 2024b. Charxiv: Charting gaps in realistic chart understanding in multimodal llms. *Advances in Neural Information Processing Systems*, 37:113569–113697.

Junfei Wu, Jian Guan, Kaituo Feng, Qiang Liu, Shu Wu, Liang Wang, Wei Wu, and Tieniu Tan. 2025. Reinforcing spatial reasoning in vision-language models with interwoven thinking and visual drawing. *arXiv preprint arXiv:2506.09965*.

Penghao Wu and Saining Xie. 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*, pages 13084–13094.

Wenshan Wu, Shaoguang Mao, Yadong Zhang, Yan Xia, Li Dong, Lei Cui, and Furu Wei. 2024a. Mind's eye of llms: visualization-of-thought elicits spatial reasoning in large language models. *Advances in Neural Information Processing Systems*, 37:90277–90317.

Yixuan Wu, Yizhou Wang, Shixiang Tang, Wenhao Wu, Tong He, Wanli Ouyang, Jian Wu, and Philip Torr. 2024b. Dettoolchain: A new prompting paradigm to unleash detection ability of mllm. *arXiv preprint arXiv:2403.12488*.

x.AI. 2024. Grok-1.5 vision preview.

Dawei Yan, Yang Li, Qing-Guo Chen, Weihua Luo, Peng Wang, Haokui Zhang, and Chunhua Shen. 2025. Mmcr: Advancing visual language model in multimodal multi-turn contextual reasoning. *arXiv preprint arXiv:2503.18533*.

Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. 2023a. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v. *arXiv preprint arXiv:2310.11441*.

Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. 2023b. Mm-react: Prompting chatgpt for multimodal reasoning and action. *arXiv*.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, and 1 others. 2024. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9556–9567.

Andy Zeng, Maria Attarian, Brian Ichter, Krzysztof Choromanski, Adrian Wong, Stefan Welker, Federico Tombari, Aveek Purohit, Michael Ryoo, Vikas Sindhwani, and 1 others. 2022. Socratic models: Composing zero-shot multimodal reasoning with language. *arXiv preprint arXiv:2204.00598*.

Guanghao Zhang, Tao Zhong, Yan Xia, Zhelun Yu, Haoyuan Li, Wanggui He, Fangxun Shu, Mushui Liu, Dong She, Yi Wang, and 1 others. 2025a. Cmmcot: Enhancing complex multi-image comprehension via multi-modal chain-of-thought and memory augmentation. *arXiv preprint arXiv:2503.05255*.

Jianshu Zhang, Dongyu Yao, Renjie Pi, Paul Pu Liang, and Yi R. Fung. 2025b. Vlm2-bench: A closer look at how well vlms implicitly link explicit matching visual cues. *Preprint*, arXiv:2502.12084.

Jiarui Zhang, Mahyar Khayatkhoei, Prateek Chhikara, and Filip Ilievski. 2025c. Mllms know where to look: Training-free perception of small visual details with multimodal llms. *arXiv preprint arXiv:2502.17422*.

Xintong Zhang, Zhi Gao, Bofei Zhang, Pengxiang Li, Xiaowen Zhang, Yang Liu, Tao Yuan, Yuwei Wu, Yunde Jia, Song-Chun Zhu, and 1 others. 2025d. Chain-of-focus: Adaptive visual search and zooming for multimodal reasoning via rl. *arXiv preprint arXiv:2505.15436*.

Yi-Fan Zhang, Xingyu Lu, Shukang Yin, Chaoyou Fu, Wei Chen, Xiao Hu, Bin Wen, Kaiyu Jiang, Changyi Liu, Tianke Zhang, and 1 others. 2025e. Thyme: Think beyond images. *arXiv preprint arXiv:2508.11630*.

Jinliang Zheng, Jianxiong Li, Sijie Cheng, Yinan Zheng, Jiaming Li, Jihao Liu, Yu Liu, Jingjing Liu, and Xianyuan Zhan. 2024. Instruction-guided visual masking. *arXiv preprint arXiv:2405.19783*.

Ziwei Zheng, Michael Yang, Jack Hong, Chenxiao Zhao, Guohai Xu, Le Yang, Chao Shen, and Xing Yu. 2025. Deepeyes: Incentivizing" thinking with images" via reinforcement learning. *arXiv preprint arXiv:2505.14362*.

Muzhi Zhu, Hao Zhong, Canyu Zhao, Zongze Du, Zheng Huang, Mingyu Liu, Hao Chen, Cheng Zou, Jingdong Chen, Ming Yang, and 1 others. 2025. Active-o3: Empowering multimodal large language models with active perception via grpo. *arXiv preprint arXiv:2505.21457*.

Chengke Zou, Xingang Guo, Rui Yang, Junyu Zhang, Bin Hu, and Huan Zhang. 2024. Dynamath: A dynamic visual benchmark for evaluating mathematical reasoning robustness of vision language models. *arXiv preprint arXiv:2411.00836*.

A More on Related Work

A.1 Existing MLLM Benchmarks

Existing Multi-modal Benchmarks. Existing visual reasoning benchmarks are typically limited: (1) Passive Visual Q&A: focusing on "look-and-answer" type of tasks without any active interactions (Bi et al., 2025; Rahmanzadehgervi et al., 2024; Yue et al., 2024; Wang et al., 2024b; Lu et al., 2023; Zou et al., 2024; Wu and Xie, 2024; Scale AI, 2025). (2) Superficial Tool Use: including only basic tools like cropping, failing to test deeper image manipulation (Wang et al., 2024a; Ma et al., 2024a). (3) Tool-Agnostic Conversations: evaluation multi-turn dialogue but without integrating essential tools (Liu et al., 2024a; Yan et al., 2025). See (Li et al., 2024) for a more comprehensive survey of multimodal benchmarks.

A.2 Learning to think with images

Prompting Methods Prompt-based methods enable MLLMs to coordinate predefined visual tools without parameter updates, turning static inputs into actively explorable workspaces via in-context learning. Early work like Socratic Models (Zeng et al., 2022), PromptCap (Hu et al., 2022), and MM-REACT (Yang et al., 2023b) showed that language can mediate collaboration, allowing text-only LLMs to function as visual reasoners by orchestrating vision experts through dialogue or targeted captions. Other approaches manipulate inputs directly: visual prompt engineering (e.g., red circles) (Shtedritski et al., 2023; Zhang et al., 2025b), Visualization-of-Thought (Wu et al., 2024a), and Visual Abstract Thinking (Liu et al., 2025a) enhance perception by highlighting, abstracting, or structuring images, while ZoomEye (Shen et al., 2024), ViCrop (Zhang et al., 2025c), Chain-of-Spot (Liu et al., 2024b), and VisuoThink (Wang et al., 2025b) extend this into systematic zooming and multimodal tree search. Finally, specialized experts can be integrated via prompting: Set-of-Mark (Yang et al., 2023a) leverages segmentation, DetToolChain (Wu et al., 2024b) structures detection reasoning, DyFO (Li et al., 2025) uses MCTS for adaptive focus, and Visual Thoughts (Cheng et al., 2025) frames expert outputs as cached "visual thoughts." Collectively, these works demonstrate that careful prompting—through language mediation, input manipulation, or expert integration—can significantly enhance multimodal reasoning without retraining.

Think with Image via SFT Supervised fine-tuning (SFT) is a primary method for teaching MLLMs to use external tools or internal visual skills by training on datasets that demonstrate tool invocation and integration. For external orchestration, models like LLaVA-Plus (Liu et al., 2023), TACO (Ma et al., 2024b), and VTS-V (Bai et al., 2025b) learn to compose tools (e.g., OCR, calculators) and follow procedural chains of reasoning. For internal manipulation, frameworks such as CogCoM (Qi et al., 2024), VGR (Wang et al., 2025a), and UniVG-R1 (Bai et al., 2025a) show how SFT can endow models with intrinsic capabilities like cropping, grounding, or fine-grained perception. Finally, SFT also cultivates dynamic visual attention: Visual CoT (Shao et al., 2024), IVM (Zheng et al., 2024), CMM-CoT (Zhang et al., 2025a), selective revisitation (Chung et al., 2025), and V* (Wu and Xie, 2024) demonstrate how training with attentional annotations transforms attention into an active, controllable skill. Across these directions, SFT provides the supervision that converts high-level reasoning into executable visual actions.

Think with Image via Reinforcement Learning. Reinforcement learning (RL) advances beyond supervised imitation by enabling models to optimize policies for visual reasoning through interaction and feedback. Foundational studies such as Jigsaw-R1 (Wang et al., 2025c), V-Triune (Ma et al., 2025), and VisionReasoner (Liu et al., 2025c) established that RL improves generalization over SFT and supports unified frameworks for diverse perception tasks. Building on this, GRIT (Fan et al., 2025), Point-RFT (Ni et al., 2025), and Seg-Zero (Liu et al., 2025b) demonstrated policies that embed spatial cues (e.g., bounding boxes, positional prompts) into reasoning, forming multimodal chains of thought. RL has also enabled active tool orchestration: VisTA (Huang et al., 2025) learns tool-selection policies, Chain-of-Focus (Zhang et al., 2025d) and ACTIVE-o3 (Zhu et al., 2025) develop adaptive zooming and region proposals, while DeepEyes (Zheng et al., 2025) achieves interleaved multimodal reasoning without SFT. Exploration is further incentivized by Pixel-Reasoner (Su et al., 2025a), while VILASR (Wu et al., 2025) leverages drawing-based reasoning,

and OpenThinkIMG (Su et al., 2025b) introduces the first open-source end-to-end RL framework for invoking diverse external tools. Collectively, these approaches move MLLMs from passive viewers to active visual agents.

B More on Evaluation Results

B.1 VTB Performance over Time

Figure 4 plots the Overall ACC of every major frontier model against its public release date, with one line per model family (OpenAI, Anthropic, Google). Three observations stand out. First, every family has improved rapidly on VISUALTOOLBENCH over the past year, with end-to-end gains of $\sim$ 17–24 accuracy points across roughly a 12-month window (Gemini-2.5-Pro $\rightarrow$ Gemini-3.1-Pro: $+17.4$ pts; GPT-4.1 $\rightarrow$ GPT-5.4-think: $+23.7$ pts; Claude-Sonnet-4 $\rightarrow$ Claude-Opus-4.6-think: $+22.0$ pts). Second, the families have converged: although early-2025 models from the three labs differed substantially in capability, their latest releases now cluster within 3 points of each other (26–29% ACC). Third, and most importantly, no released model has yet crossed the 30% ACC threshold (dashed line in the figure), indicating that VISUALTOOLBENCH remains far from saturation despite the rapid year-over-year gains. We expect this remaining headroom to continue narrowing as the next generation of think-with-image models is released.

B.2 Per Task-Category Performance

Figure 9 reports per-task-category accuracy for the top six models. Two patterns emerge. First, *multi-turn tasks are consistently harder than single-turn tasks* for every model: the two single-turn categories (Region-Switch Q&A and Hybrid Tool Reasoning) achieve substantially higher ACC than the three multi-turn categories (Follow-up Test, Temporal Sequential Reasoning, and Progressive Visual Reasoning). For the strongest models the multi-turn gap is roughly 2×, and it widens further for weaker models, reflecting how compounded reasoning errors and tool-use mistakes accumulate across conversational turns. Second, the relative *ranking of task categories is stable across models*: every model finds Region-Switch Q&A and Hybrid Tool Reasoning easier than the multi-turn categories, with the Follow-up Test format the hardest in nearly every case. The consistency of this ordering suggests that task-category difficulty is an intrinsic property of VTB rather than an artifact of how a particular model interacts with tools.

B.3 More on Evaluation Metrics

B.3.1 Final-Answer Evaluation Metrics

We score each model response against task-specific rubrics. For each task $i \in \{1, \ldots, N\}$:

1. **Obtain the model's final response.** The model is given the {prompt, image} pair and may invoke our predefined tool set while producing its answer.

2. **Per-rubric grading.** For each rubric criterion $j \in \{1, \ldots, N_i\}$, an LLM judge decides whether the criterion is met given the model's response.

3. **Weighted rubric score.** We compute task i's rubric score as

$$s_i = \frac{\sum_{j=1}^{N_i} \mathbb{1}_{r_{ij}}\, w_{ij}}{\sum_{j=1}^{N_i} w_{ij}}, \tag{1}$$

where $w_{ij} \in \{1, 2, 3, 4, 5\}$ is the rubric weight and $\mathbb{1}_{r_{ij}}$ indicates whether criterion j is met.

Average Rubric Score (ARS). The benchmark-level ARS is the mean of the per-task rubric scores:

$$\mathrm{ARS} = \frac{1}{N} \sum_{i=1}^{N} s_i. \tag{2}$$

Accuracy (ACC). We additionally report binary task-level accuracy. A task is considered *passed* iff every critical rubric (weight ≥ 4) is satisfied. ACC is the fraction of passed tasks; it is the conservative complement to ARS, sensitive to a single missed critical rubric.

Category Pass Rate (CPR). For each rubric category C (Visual Understanding, Truthfulness, Instruction Following, Reasoning, Writing Style), we define

$$\mathrm{CPR}_C = \operatorname*{mean}_i \left[\frac{\sum_{j:\, C \in \mathrm{cat}(r_{ij})} \mathbb{1}_{r_{ij}}\, w_{ij}}{\sum_{j:\, C \in \mathrm{cat}(r_{ij})} w_{ij}} \right], \tag{3}$$

i.e. the weighted fraction of category-C rubrics satisfied, averaged over the tasks that contain at least one category-C rubric. Because rubric category tagging is multi-label (a single rubric may belong to multiple categories), a rubric contributes at full weight to each of its categories; consequently the per-category CPRs do not sum to the overall ARS.

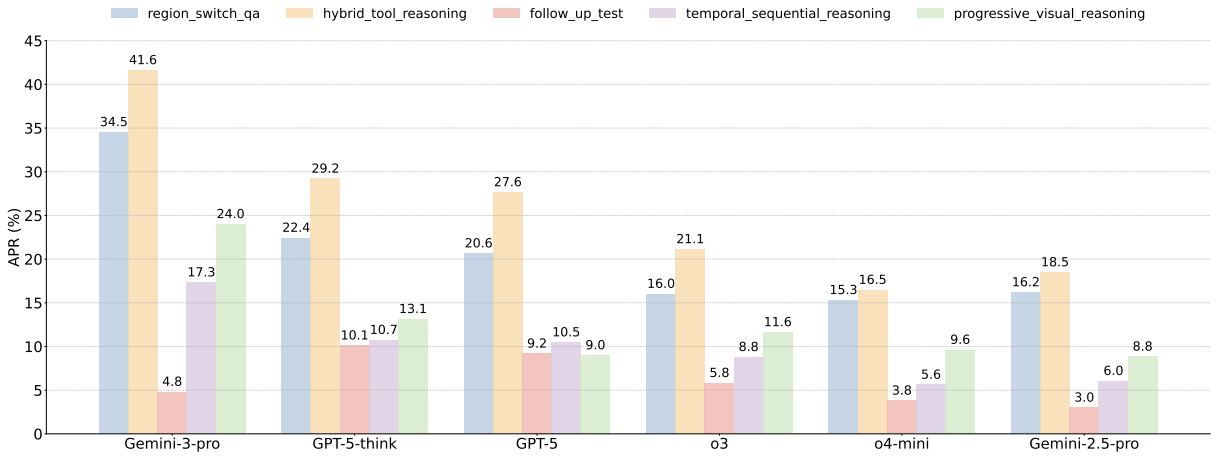


Figure 9: ACC across task categories for the top six models on VISUALTOOLBENCH. Single-turn categories (Region-Switch Q&A, Hybrid Tool Reasoning) sit substantially above the three multi-turn categories (Follow-up Test, Temporal Sequential Reasoning, Progressive Visual Reasoning) for every model.

Rubric weights. To ensure consistent rubric authoring, each rubric is assigned a weight $w \in \{1, 2, 3, 4, 5\}$ reflecting its importance. Table 7 summarizes the five weight levels, from *incidental* stylistic preferences to *critical* elements that determine overall task validity. Task contributors are instructed to assign weights in accordance with these guidelines.

B.3.2 Tool-Use Evaluation Metrics

We score tool-use behavior from execution traces recorded by the evaluation harness. Let $\mathcal{T}$ denote the set of tasks ($N = |\mathcal{T}|$), and let $\mathcal{T}_i = \{c_{i1}, c_{i2}, \dots\}$ denote the (ordered) multiset of tool invocations on task i.

Tool-Call Proactivity (TCP). The fraction of tasks on which the model invokes at least one tool:

$$\mathrm{TCP} = \frac{\left|\{\, i \in \mathcal{T} \mid |\mathcal{T}_i| > 0 \,\}\right|}{|\mathcal{T}|}. \tag{4}$$

Higher values indicate more frequent tool use, although the calls may be either beneficial or redundant.

Tool-Call Success Rate (TCSR). The fraction of valid tool invocations across all calls:

$$\mathrm{TCSR} = \frac{\sum_{i \in \mathcal{T}} \sum_{c \in \mathcal{T}_i} \mathbb{1}[\mathrm{valid}(c)]}{\sum_{i \in \mathcal{T}} |\mathcal{T}_i|}, \tag{5}$$

where $\mathbb{1}[\mathrm{valid}(c)] = 1$ if tool call c executes without an error message and 0 otherwise. TCSR measures how reliably a model adheres to tool specifications.

Tool-Call Volume (TCV). The average number of tool calls per task:

$$\mathrm{TCV} = \frac{1}{N} \sum_{i \in \mathcal{T}} |\mathcal{T}_i|. \tag{6}$$

B.4 Average Rubric Score Results

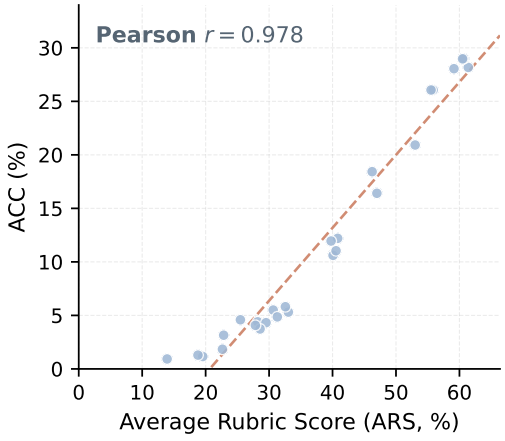


Figure 10: ACC and ARS are highly correlated across the 26 evaluated models (Pearson $r = 0.978$).

Although ACC is a strict binary outcome (a single missed critical rubric fails the task) while ARS is a continuous weighted score across all rubrics, the two metrics agree almost perfectly on which models are strong and which are weak (Figure 10). This high correlation does not mean the rubrics are redundant. On the contrary, the partial-credit scores let us track incremental progress for weaker models whose binary pass rate would otherwise floor at zero. For brevity we therefore report ACC as the primary headline accuracy in the main paper. In

Table 7: Rubric weights used in VISUALTOOLBENCH. Higher weights indicate greater importance; critical rubrics determine task-level pass/fail under ACC.

Weight	Description
Critical (5)	**Non-negotiable.** Must be present for the answer to count as valid; failing it implies task failure regardless of other strengths.
Significant (4)	**Central to success.** Omitting this degrades output quality or creates confusion about how the task was solved.
Moderate (3)	**Meaningfully important.** Affects clarity or correctness; its absence weakens the answer but does not make it invalid.
Minor (2)	**Adds polish.** Omitting it slightly lowers quality but does not break the core solution.
Incidental (1)	**Marginally relevant.** A nice-to-have detail or stylistic preference; does not affect whether the task is solved.

addition, Table 8 presents the detailed rubric score for both single-turn and multi-turn tasks. It can be seen that APR and ARS are positively correlated, models have higher APR also have higher ARS.

B.5 Tool-Use Analysis

Table 9 reports the detailed tool-use metrics for all evaluated models. By combining these results with model performance (APR in Table 2 and ARS in Table 8), we make the following key observations:

1. **More tool calls do not necessarily translate to better performance.** For instance, o3 makes the largest number of calls (16,116), yet performs worse than GPT-5 (10,213 calls) and GPT-5-think (13,429 calls).

2. **High proactivity does not guarantee strong results.** Claude models exhibit very high proactivity, yet their performance remains poor, with overall APR below 5% and ARS values under 0.3.

3. **Low proactivity and low call volume generally correlate with poor performance.** For example, the Llama models, Pixtral-large, GPT-4o, and GPT-4.1 all demonstrate relatively low proactivity and correspondingly weak performance.

B.6 LLM-as-Judge vs. Human-as-Judge

To enable large-scale evaluation of VISUALTOOLBENCH, we employ LLMs as automatic judges and compare their assessments against human annotations. Table 10 reports the overall alignment rate on a subset of VISUALTOOLBENCH, as well as breakdowns for objective and subjective rubrics across three judge models.

All models achieve high alignment on objective rubrics, while alignment on subjective rubrics is comparatively lower, reflecting the inherent ambiguity of subjective evaluation. Nevertheless, overall alignment rates remain close to 90%, underscoring the reliability of LLM-as-judge for our benchmark. We choose to use o4-mini as our judge model for the main experimental study.

C More on Error Analysis

In this section, we provide the fine-grained tool-use failure analysis. Specifically, our analysis covering eleven categories of common tool-related errors, including visual-perception mistakes, incorrect cropping, incorrect rotation/flip operations, OCR errors, distraction-induced mistakes, and incorrect tool selection as shown in Table 11. We perform such analysis on 100 common failure tasks on three evaluated models: GPT-5, Claude-opus-4.1, and Gemini-2.5-pro.

To support large-scale evaluation, we prompt GPT-5 with the question, the model's tool-use chain, the golden answer, and the model's actual answer to generate an initial failure-category prediction. Human supervision is then applied to filter out any inappropriate or incorrect categories. We examined three frontier models and select 100 common failure tasks for this fine-grained tool-use failure analysis.

We emphasize that reasonably good visual perception is essential for enabling effective tool use

Table 8: ARS results of the evaluated models across domains. The best results in each column are highlighted in bold.

Model	Overall	Single-Turn					Multi-Turn				
		STEM	Medicine	Finance	Sports	Generalist	STEM	Medicine	Finance	Sports	Generalist
Open-Source Models											
Qwen3-VL-30B-think	0.1393	0.0894	0.1073	0.0830	0.1152	0.1272	0.2370	0.1806	0.1874	0.1483	0.1177
Qwen3-VL-235B-think	0.2284	0.2077	0.2141	0.1643	0.1783	0.1784	0.3428	0.3458	0.2337	0.2305	0.1913
Llama4-Maverick	0.1955	0.1943	0.1627	0.1174	0.1293	0.1435	0.2821	0.3188	0.2199	0.2094	0.1821
Llama4-Scout	0.1879	0.1988	0.1430	0.1223	0.1759	0.1621	0.2728	0.2558	0.1969	0.1910	0.1543
Closed-Source Models											
Gemini-3.1-pro	0.5914	0.6208	0.5670	**0.7133**	0.5604	0.5526	0.6487	0.5455	0.6308	0.5709	0.5021
Gemini-3-pro	0.5586	0.5936	0.4972	0.6465	0.5117	0.5236	0.6041	0.5281	0.6029	0.5610	0.5160
Gemini-3-flash	**0.6139**	**0.6356**	**0.6346**	0.6707	0.5715	0.6321	**0.6618**	0.5793	0.6154	0.6077	0.5331
Gemini-2.5-pro	0.4008	0.3747	0.3270	0.3777	0.3661	0.3840	0.4352	0.4999	0.3979	0.4701	0.3802
Gemini-2.5-flash	0.2821	0.2274	0.1652	0.1755	0.2308	0.1857	0.4032	0.4762	0.2650	0.4226	0.3050
GPT-5.4-think	0.6067	0.5984	0.5473	0.6850	**0.5823**	**0.6460**	0.6519	0.5903	0.6211	0.5888	0.5493
GPT-5.4	0.4081	0.4606	0.3948	0.3179	0.3958	0.3787	0.4940	0.4529	0.4253	0.4115	0.3542
GPT-5-think	0.4624	0.4473	0.4113	0.3720	0.4410	0.4646	0.5677	0.5267	0.4471	0.5031	0.4433
GPT-5	0.4699	0.4604	0.4329	0.4574	0.4173	0.4319	0.5590	0.5229	0.4497	0.4956	0.4739
o3	0.3977	0.3342	0.3872	0.3153	0.3604	0.3155	0.4652	0.5289	0.4282	0.4526	0.3934
o4-mini	0.4055	0.3660	0.3927	0.3519	0.3840	0.4337	0.4464	0.4839	0.3749	0.4286	0.3925
GPT-4.1	0.3304	0.2819	0.2924	0.2112	0.3382	0.2836	0.4219	0.4392	0.3111	0.4150	0.3125
Claude-opus-4.6-think	0.6049	0.5941	0.6269	0.6588	0.5291	0.5608	0.6544	**0.6024**	**0.6474**	**0.6221**	**0.5509**
Claude-opus-4.5-think	0.5302	0.4972	0.5450	0.5930	0.4675	0.4759	0.5475	0.5606	0.5673	0.5617	0.4854
Claude-opus-4.1-think	0.2548	0.2424	0.2574	0.2671	0.2862	0.2736	0.2555	0.2956	0.2024	0.2439	0.2203
Claude-opus-4.1	0.2952	0.2521	0.2507	0.2233	0.2558	0.2656	0.3739	0.4351	0.2605	0.3395	0.2977
Claude-sonnet-4.5-think	0.3065	0.2652	0.3033	0.2638	0.2271	0.2967	0.3548	0.3890	0.3295	0.3359	0.2970
Claude-sonnet-4.5	0.3130	0.2775	0.2623	0.2554	0.3144	0.2878	0.3396	0.4121	0.3425	0.3499	0.2920
Claude-sonnet-4-think	0.2858	0.2280	0.2710	0.2359	0.2514	0.2547	0.3495	0.4034	0.3101	0.2865	0.2689
Claude-sonnet-4	0.2784	0.2560	0.2603	0.2163	0.2431	0.2421	0.3363	0.3857	0.2824	0.2988	0.2660
Nova-Premier	0.2265	0.1984	0.2082	0.1530	0.2076	0.1929	0.3377	0.3325	0.1907	0.2364	0.2214

in our benchmark. For example, some tasks require the model to first localize a small object or region. However, as our examples illustrate later, models often fail to propose accurate cropping coordinates in one shot, leading to zooming into irrelevant regions, and this makes models to perform multiple unnecessary tools calls, which may propagate downstream errors. Overall, we believe that this fine-grained error analysis provides process-level insight into where current MLLMs struggle, complementing our existing metrics and strengthening the evaluation framework.

D More on Experimental Setup

In this section, we provide more details on our experimental setup.

D.1 Baseline Models

We evaluated 26 representative MLLMs, covering both reasoning and non-reasoning as well as open and closed-sourc models. Whenever possible, we set the temperature to 0. For o3, o4-mini, GPT-5, and GPT-5-think, the temperature is set to 1. Since GPT-5 is inherently a reasoning model, we use its default reasoning effort (`medium`) for GPT-5, and GPT-5-thinking corresponds to the `high` reasoning effort. For o3 and o4-mini, we adopt their default reasoning effort (`medium`). For Claude's thinking mode, we set the reasoning budget to 5000 tokens. All other model API hyperparameters are kept at their default settings without further customization.

D.2 Prompts

In our main experimental study, we use a strong system prompt to encourage models to use tools towards solving VISUALTOOLBENCH tasks. In addition, we design a weaker system prompt for ablation study. The strong and weak system prompts are provided below.

System Prompt

You are a *proactive, tool-empowered* visual-reasoning assistant.

When user supplies an image and requests to solve a problem that requires visual content that are small, ambiguous, or not centered, you must:

1. Examine the image carefullly and mentally list the visual clues most likely to locate the target object.

2. Proactively use the image-processing tools - such as crop, zoom, or enhance - to isolate and

Table 9: Tool call results of the evaluated models. The best results in each column are highlighted in **bold**.

Model	Total #	Single-Turn			Multi-Turn		
		Proactivity	Success Rate	Volume	Proactivity	Success Rate	Volume
Open-Source Models							
Llama4-Maverick	315	0.1625	0.6609	0.19	0.2550	0.7236	0.33
Llama4-Scout	110	0.0729	0.6140	0.09	0.0760	0.4906	0.09
Pixtral-large	1315	0.3947	0.6733	0.50	0.3144	0.2857	1.14
Qwen3-VL-30B-think	949	0.3068	0.9429	0.41	0.5807	0.9233	1.17
Qwen3-VL-235B-think	2745	0.7413	0.9852	1.57	0.8552	0.9744	2.99
Closed-Source Models							
GPT-4o	1024	0.2056	0.6273	0.61	0.2463	0.5914	1.08
GPT-4.1	535	0.1359	0.8038	0.26	0.2728	0.6790	0.63
o3	**16116**	0.9453	0.8587	**9.98**	0.9750	0.8683	**16.80**
o4-mini	5337	0.8159	0.8199	3.34	0.8835	0.8281	5.53
GPT-5	10212	0.9652	0.8555	6.46	0.9883	0.8476	10.53
GPT-5-think	13429	0.9900	0.8660	7.45	0.9950	0.7873	13.79
GPT-5.4-think	7658	0.9950	0.7994	4.60	0.9917	0.8604	8.13
Gemini-2.5-flash	2313	0.6650	0.8605	1.87	0.4809	0.8742	1.96
Gemini-2.5-pro	3366	0.8060	0.7331	1.94	0.9251	0.7656	3.66
Gemini-3-flash	7887	0.9784	0.9494	4.53	**1.0000**	0.9465	8.58
Gemini-3-pro	5239	0.9784	**0.9644**	3.03	**1.0000**	**0.9601**	5.68
Gemini-3.1-pro	2163	0.5489	0.9938	1.07	0.7737	0.9756	2.52
Claude-sonnet-4	6941	0.9768	0.9252	4.52	0.9917	0.9299	7.01
Claude-sonnet-4-think	8704	0.9735	0.9352	3.84	0.7671	0.9328	3.91
Claude-opus-4-1	6538	0.9536	0.9524	3.83	0.9900	0.9452	7.03
Claude-opus-4-1-think	2082	0.9486	0.9524	3.45	0.9950	0.9583	6.72
Claude-opus-4.6-think	9529	**0.9983**	0.9782	5.77	0.9983	0.9691	10.07
Nova-premier	5109	0.9055	0.5444	2.88	0.9734	0.5163	5.61

Table 10: Alignment rates of different LLMs when serving as judges.

Judge Model	Overall	Objective Rubrics	Subjective Rubrics
o4-mini	88.07%	91.70%	73.96%
GPT-4.1	88.18%	89.83%	81.77%
GPT-4o	87.01%	89.16%	78.65%

clarify the relevant region.
3. Save each transformed image. The updated image will be appended to the conversation for your reference.
4. Iterate as needed. Call the tools repeatedly until the visual evidence is clear enough to answer the user's request.
5. Double-check your observations. Confirm that the final transformed image supports an accurate, confident response before replying to the user.
6. Use other general-purpose tools if needed to answer the user's question.
7. Please use the tools wisely as you have limited tool calls.

D.3 LLM-as-Judge Protocol

We employ an LLM-as-judge framework for large-scale evaluation. To improve reliability, each judgment considers one rubric at a time. All rubrics are atomic: each rubric targets a single, verifiable fact or behavior, thereby reducing ambiguity, limiting error propagation, and yielding clearer agreement signals.

The exact judge prompts are provided below. For each case, the judge receives: (i) the original question, (ii) the gold answer, (iii) a single rubric criterion, and (iv) the model's answer; it returns a verdict (met or not met) with a brief, evidence-grounded justification.

Table 11: Fine-grained categories of vision tool-use failure.

Failure Type	Description
Visual Misinterpretation	The model misperceives or misunderstands original or processed visual content.
OCR Error	The model incorrectly reads, extracts, or transcribes text from an image.
Distraction	The model's response is distracted by the tool-usage.
Tool Execution Error	The model applies it incorrectly, producing invalid tool observations.
Incorrect Tool Selection	The model chooses the wrong tool for the task.
Incorrect Cropping	The model crops the wrong region, uses incorrect crop boundaries.
Incorrect Rotation/Flipping	The model rotates or flips the image in the wrong direction or when rotation was not required.
Incorrect Enhancing	The model incorrectly applies enhancement operations (brightness, contrast, sharpness).
Incorrect Editing	The model performs an inappropriate editing action.
Incorrect Resizing	The model resizes the image incorrectly, including wrong scaling factors or unintended aspect ratio changes.
Other	Any failure modes not captured by the predefined categories.

LLM Judge Prompt

You are an expert evaluator tasked with judging whether a model's answer meets a specific rubric criterion. You will be provided with:
- a question
- a golden (reference) answer
- a rubric criterion
- the model's answer

Your task is to decide if the model's answer **meets** or **does not meet** the given rubric criterion, referencing the golden answer only as needed.

Inputs:
Question: question
Golden Answer: golden_answer
Rubric Criterion: rubric_criteria
Model Answer: model_answer

Important Notes:
- The model's answer does not need to be correct to meet the criterion if correctness is not required.
Example: If the rubric is "The model should show its reasoning process to answer the question," the answer can be incorrect but still meet the rubric if model's reasoning process is present.
- For writing style or presentation rubrics, apply leniency.
Example: If the rubric asks for conciseness, answers that are slightly longer than the golden answer but still reasonably length should be considered as meeting the rubric.
- The model's answer may satisfy the rubric implicitly without explicitly mentioning the exact term. This should still be considered as meeting the criterion if model's answer is reasonable and makes sense.
Example: If the rubric is "The model should demonstrate understanding of photosynthesis," and the model states "Plants make their own food using sunlight," without explicitly mentioning the term "photosynthesis," it still meets the criterion.

Output Format:
Return your judgment in the following JSON format:

```
{
    "explanation": "Brief explanation of your
    ↪  judgment",
    "judge_result": "Met" or "Not Met"
}
```

E More on VISUALTOOLBENCH

E.1 More Benchmark Visual Inputs

Figure 11 to Figure 15 show more benchmark images from VTB.

E.2 Task Domain and Sub-domain Taxonomy

In this section, we provide the complete statistics of task domains and subdomains distribution in Table 13.

E.3 More on Task Generation Process

All VISUALTOOLBENCH tasks are authored by human contributors with diverse domain expertise. To ensure benchmark quality and realism, we adopt a rigorous multi-stage data collection pipeline:

1. **Contributor Training.** Contributors are first introduced to the project scope, task cate-

Table 12: Model endpoints and hyperparameter setup.

Model Provider	Model Endpoint	Hyperparameter
OpenAI	gpt-4o	`temperature = 0.0`
	gpt-4.1	`temperature = 0.0`
	o3	`reason_effort = "medium"`
	o4-mini	`reason_effort = "medium"`
	gpt-5	`reason_effort = "medium"`
	gpt-5-think	`reason_effort = "high"`
	gpt-5.4-2026-03-05	`reason_effort = "medium"`
	gpt-5.4-2026-03-05-think	`reason_effort = "high"`
Anthropic	claude-sonnet-4-20250514	`temperature = 0.0`
	claude-sonnet-4-20250514 (thinking)	`thinking_budget = 5000`
	claude-sonnet-4.5	`temperature = 0.0`
	claude-sonnet-4.5 (thinking)	`thinking_budget = 5000`
	claude-opus-4.1	`temperature = 0.0`
	claude-opus-4.1 (thinking)	`thinking_budget = 5000`
	claude-opus-4.5 (thinking)	`thinking_budget = 5000`
	claude-opus-4.6 (thinking)	`thinking_budget = 5000`
Google	gemini-2.5-flash	`temperature = 0.0`
	gemini-2.5-pro	`temperature = 1.0`
	gemini-3-flash-preview	`temperature = 1.0`
	gemini-3-pro-preview	`temperature = 1.0`
	gemini-3.1-pro-preview	`temperature = 1.0`
Meta	llama4-maverick-instruct	`temperature = 0.0`
	llama4-scout-instruct	`temperature = 0.0`
Amazon	nova-premier-v1:0	`temperature = 0.0`
Mistral AI	pixtral-large-latest	`temperature = 0.0`
xAI	grok-4.20-reasoning	`reason_effort = "high"`
Tongyi Qianwen	Qwen3-VL-30B-think	`temperature = 0.0`
	Qwen3-VL-235B-think	`temperature = 0.0`

gories, and submission requirements. They are instructed to provide a task prompt, input image, reference answer, reference tool-use chain, and a set of evaluation rubrics.

2. **Initial Task Design.** Drawing on their domain expertise, contributors design tasks by selecting the appropriate domain and aligning with the specified task category. Each submission includes a text prompt and associated image(s), a golden answer, a reference tool-use chain that demonstrates a valid solution path, and well-defined evaluation rubrics.

3. **Initial Model Response Grading.** Contributors are presented with responses from three representative models (o3, Gemini-2.5-pro, and o4-mini) to their designed tasks. They then grade these responses against the proposed rubrics. A task is selected only if at least two of the three models fail, thereby ensuring that the benchmark captures genuinely challenging cases.

4. **First-Round Review.** A reviewer evaluates each task for realism, necessity of dynamic image-based reasoning, correctness of the reference answer, and appropriateness of the rubrics. Tasks with minor issues may be revised, while those that are fundamentally unsound (e.g., not requiring genuine visual-tool use) are discarded.

5. **Second-Round Review.** A second indepen-

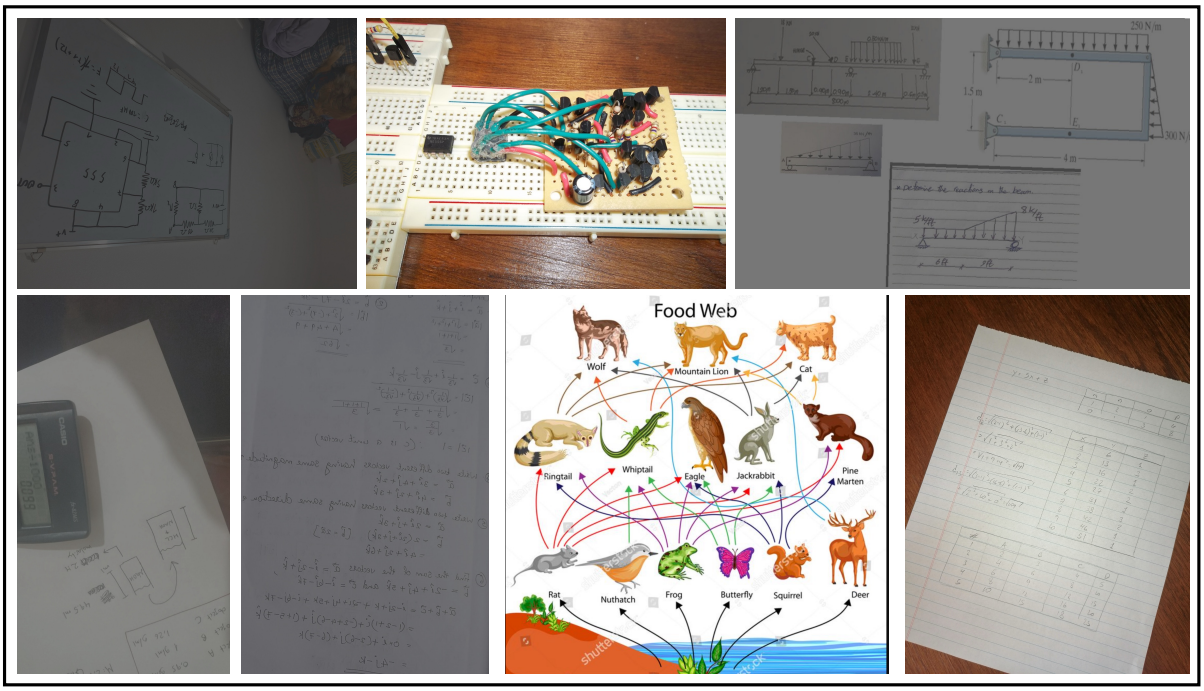


Figure 11: STEM domain representative images.

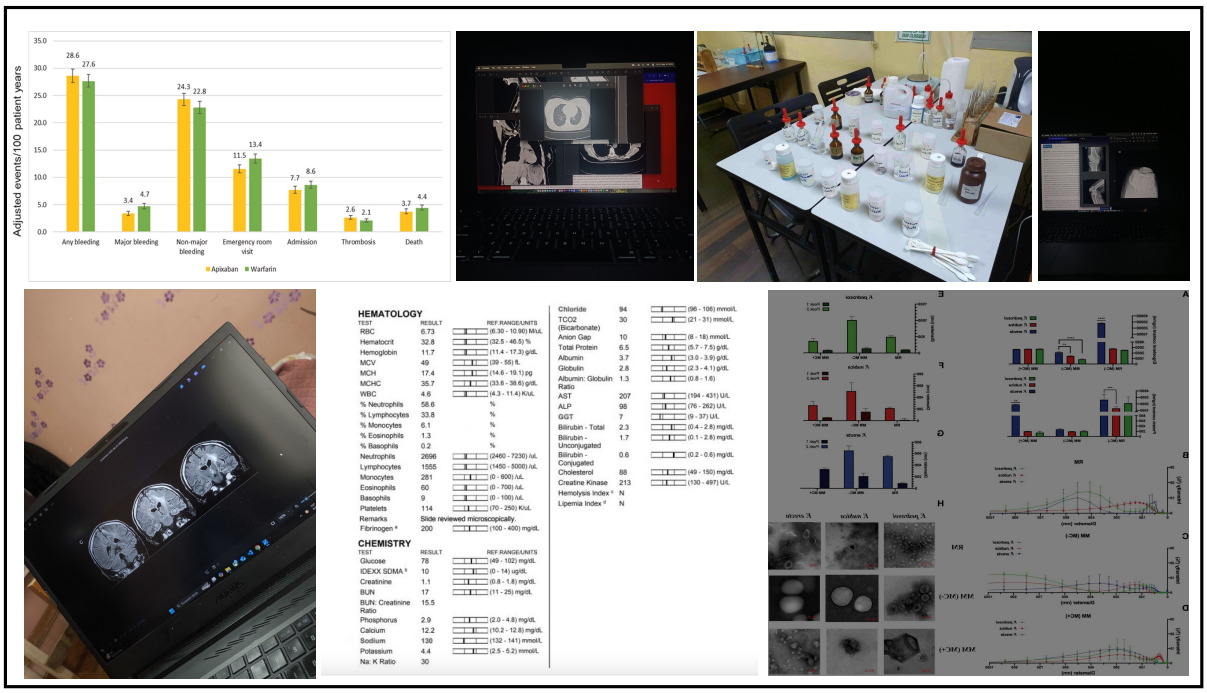


Figure 12: Medicine domain representative images.

dent reviewer validates the first-round decision, ensuring consistency and reliability across the benchmark.

6. **Final Integration.** Tasks that pass both review stages are incorporated into the benchmark, ensuring high quality, broad domain coverage, and diverse reasoning requirements.

This layered pipeline ensures that every task is original, realistic, and rigorously validated, resulting in a benchmark that robustly evaluates genuine visual intelligence.



Figure 13: Sports domain representative images.

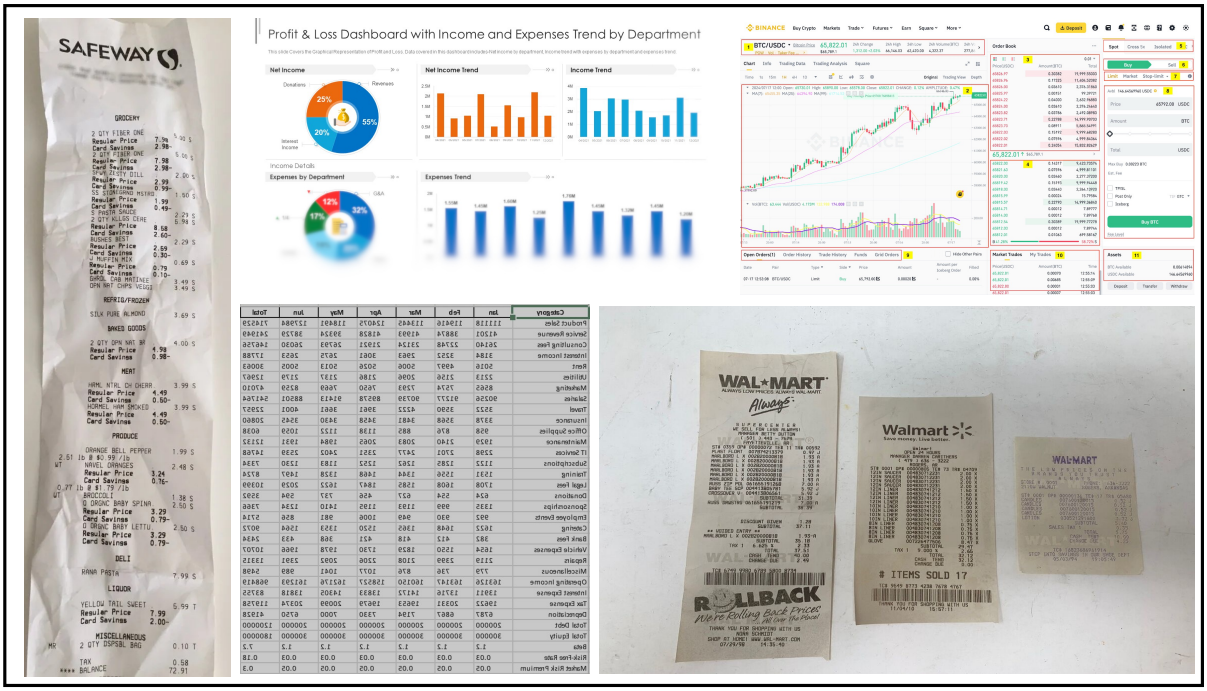


Figure 14: Finance domain representative images.

F More on Tools Analysis

In this section, we provide more details on the tools supported by VISUALTOOLBENCH and more examples on Tool-use of the evaluated models.

F.1 Tool Description

Table 14 provides tool description supported by VISUALTOOLBENCH.

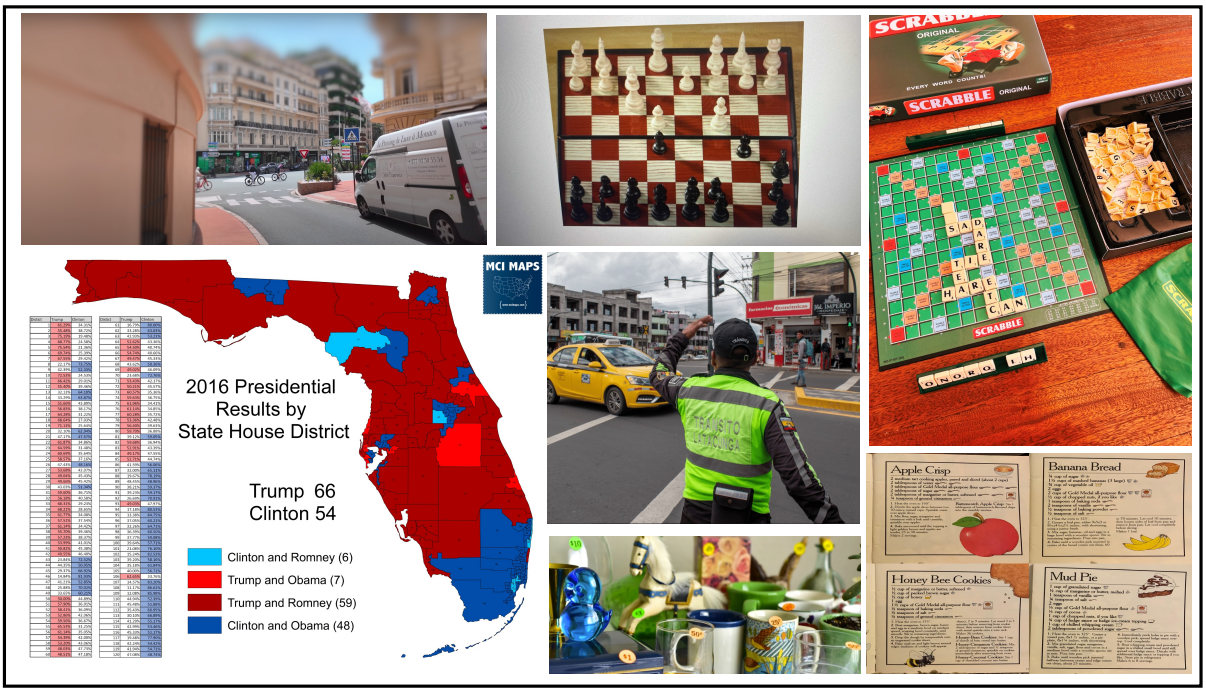


Figure 15: Generalist domain representative images.

F.2 Ablation on Atomic Vision Tools and General Vision Tool

Our evaluation toolkit includes a general-purpose vision tool, `python_image_processing`, which executes arbitrary image-processing operations based on model-generated code. To empirically study the effect of decoupling coding ability from visual reasoning, we perform an additional ablation experiment in which `python_image_processing` is replaced by an atomic toolset composed of common, predefined image-processing operations, including: crop-image, rotate-image, adjust-brightness, adjust-contrast, adjust-saturation, adjust-sharpness, resize-image, flip-image, convert-to-grayscale, and apply-filter.

We evaluated GPT-5 and GPT-4.1 on single turn tasks under two settings. The results are shown in Figure 8. We make the following key observations:

1. For GPT-4.1, the atomic tool setup provides a small improvement. However, the gain is marginal, suggesting that the model's failures stem mainly from visual reasoning rather than from coding issues.

2. On the other hand, GPT-5 shows a performance decline under atomic tools, which we hypothesize arises because atomic tools restrict the model's expressive image manipulation capabilities. GPT-5 frequently composes multi-step and creative operations in the general setting; atomic tools constrain this flexibility and can reduce its performance.

F.3 Model Tool-Call Behavior Examples

F.3.1 Image Manipulation Operations

In this section, we present the representative o3 generated python code snippets, where the model invokes the `python_image_processing` tool to perform diverse image manipulations in support of problem solving. Lines that execute the manipulations are highlighted in yellow.

Table 13: Task Distribution Across Domains and Subdomains in VISUALTOOLBENCH

Domain	Subdomain	Number of Tasks
STEM	Engineering	44
	Physics	49
	Maths	49
	Biology	52
	Chemistry	44
Finance	Market Analysis	77
	Personal Finance	90
	Corporate Finance	50
	Currency Trends	14
	Taxation Analysis	12
Medicine	Clinical Specialties	42
	Laboratory Medicine	51
	Medical Imaging	113
	Pharmaceutical Science	25
	Public Health	7
Sports	Team Sports	147
	Individual Sports	50
	Racket Sports	15
	General Sports	28
Generalist	Daily Life	31
	Transportation	38
	Game & Entertainment	86
	Problem Solving	86
	Support & Services	4
Total	—	**1,245**

Table 14: Supported tools supported in VISUALTOOLBENCH.

Tool	Functionality
python-image-processing	Vision-specific manipulation, including image cropping, editing, rotation, brightness/contrast adjustment, and enhancement. Enables iterative refinement of visual inputs.
python-interpreter	General code execution.
web-search	Open-domain information retrieval from the web.
browser-get-page-text	Extraction of textual content from online sources.
historical-weather	Weather records for temporal and geographic look-ups.
calculator	Arithmetic operations for quick computations.

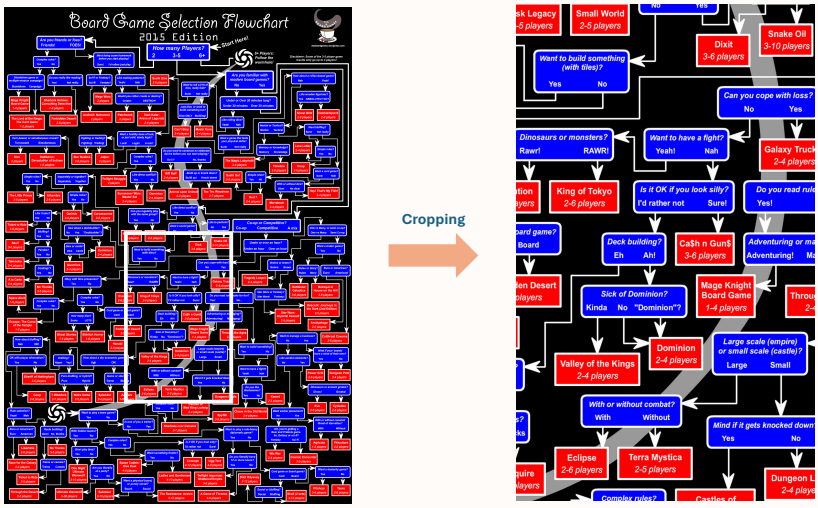
Image Operations: Crop and Resize

Task ID: 68658a7116039839194326l5

- line 9-15: image cropping
- line 17-18: image resize

```
1  from PIL import Image
2  import os
3  import numpy as np
4  import cv2
5
6  img_path = '/mnt/efs/.../vision_tool_use/Kb2JElJ87npHXeG.jpg'
7  img = Image.open(img_path)
8
9  # Crop bottom middle area where Dixit might be
10 w, h = img.size
11 left = int(w * 0.35)
12 top = int(h * 0.45)
13 right = int(w * 0.65)
14 bottom = int(h * 0.8)
15 crop = img.crop((left, top, right, bottom))
16
17 # Enlarge cropped region
18 crop_large = crop.resize((crop.size[0]*2, crop.size[1]*2))
```

```
19
20 # Save
21 os.makedirs('eval_results_single_turn/...', exist_ok=True)
22 crop_large.save('eval_results_single_turn/...', 'PNG')
```



Original Image Transformed Image

Image Operations: Rotation, Brightness and Contrast Adjustment

Task ID: 68658a7116039839194326l3

- line 5-6: image rotation
- line 7-9: brightness adjustment
- line 10-12: contrast adjustment

```
1 from PIL import Image, ImageEnhance
2 import os, numpy as np
3 img_path = '/mnt/.../vision_tool_use/PtZEjeYmkRmW-Lf.jpg'
4 img = Image.open(img_path)
5 # rotate 180 degrees
6 img_rot = img.rotate(180, expand=True)
7 # enhance brightness
8 enhancer = ImageEnhance.Brightness(img_rot)
9 img_enh = enhancer.enhance(1.8)
10 # increase contrast
11 enhancer_c = ImageEnhance.Contrast(img_enh)
12 img_enh2 = enhancer_c.enhance(1.5)
13 # save
14 save_dir = 'eval_results_single_turn/...'
15 os.makedirs(save_dir, exist_ok=True)
16 save_path = os.path.join(save_dir, 'transformed_image_0.png')
17 img_enh2.save(save_path, 'PNG')
```



Original Image

Brightness Adjustment

Rotation



Transformed Image 0

Image Operations: Flip

Task ID: 68658a711603983919432613

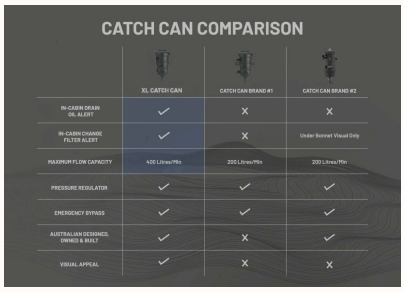
- line 4-5: image flip

```
1 from PIL import Image
2 import os, numpy as np
3 img = Image.open('/.../transformed_image_0.png')
4 # flip horizontally to correct mirroring
5 img_flipped = img.transpose(Image.FLIP\_LEFT\_RIGHT)
6 # maybe crop to table area? Keep full for readability
7 save_dir='eval_results_single_turn/...'
8 img_flipped.save(save_dir,'PNG')
```



Transformed Image 0

Flip



Transformed Image 1

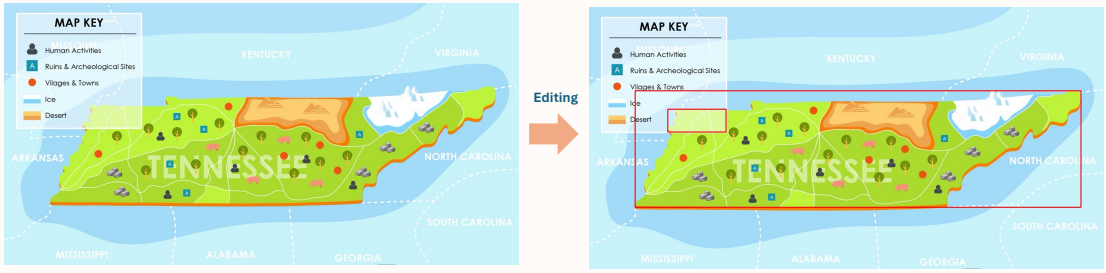
Image Operations: Editing, Others

Task ID: 68757ac94c044bac23c9bd8b

- line 23-26: image editing

```
1 from PIL import Image
2 import numpy as np
3 import cv2
4 import os
5 # Load image
6 img_path='/mnt/.../SZJPcfAUQgyEVGe.jpg'
7 img=cv2.imread(img_path)
8 # Convert to HSV
9 hsv=cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
10 # Define green range (for tree foliage), adjust
11 lower=np.array([35,40,40])
12 upper=np.array([85,255,255])
```

```
mask=cv2.inRange(hsv, lower, upper)
# Morphological operations to remove noise
kernel=np.ones((5,5),np.uint8)
mask=cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel)
# Find contours
contours,_=cv2.findContours(mask, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
# Filter contours by area to avoid tiny bits
tree_contours=[cnt for cnt in contours if cv2.contourArea(cnt)>500]
print('Totalcontourslarge', len(tree_contours))
# Draw bounding boxes to visualize
for cnt in tree_contours:
  x,y,w,h=cv2.boundingRect(cnt)
  cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),2)
# Save transformed
out_dir='/...'
os.makedirs(out_dir, exist_ok=True)
Image.fromarray(cv2.cvtColor(img, cv2.COLOR_BGR2RGB)).
save(out_dir+'/transformed_image_0.png','PNG')"
```



Original Image | Transformed Image 0

F.3.2 Vision Tool Call for GPT-5

From the first column of Table 9, we observe that although GPT-5 and GPT-5-think invoke the vision tool less frequently than o3, they perform more image manipulations overall. A closer inspection shows that GPT-5 and GPT-5-think often execute multiple image-processing operations within a single vision-tool call, thereby achieving higher tool-call efficiency than o3. We provide two demonstration examples below.

GPT-5 Vision Tool Call Output

Task ID: 68658a7116039839194326l3

-line 12-14, 21-26: saved three plus transformed images

```
from PIL import Image, ImageEnhance, ImageFilter
import os
base='eval_results_single_turn/.../transformed_image_0.png'
img=Image.open(base)
# Let's crop central table area
w,h=img.size
# Crop left column, middle, right ... for readability
# First overall crop removing margins
crop=img.crop((int(w*0.06), ..., int(h*0.92)))
# enhance
crop=ImageEnhance.Contrast(crop).enhance(1.4)
#Save
out_dir=os.path.dirname(base)
crop.save(out_dir+'/transformed_image_1.png','PNG')
# Crop each column region roughly
# Determine thirds
cw=crop.size[0]
ch=crop.size[1]
col_width=cw//3
cols=[]
for i in range(3):
```

```
  c=crop.crop((i*col_width,0,(i+1)*col_width,ch))
  c=ImageEnhance.Brightness(c).enhance(1.2)
  c=c.filter(ImageFilter.UnsharpMask(radius=1.5,percent=180,threshold=2))
  c.save(out_dir+f'/transformed_image_{2+i}.png','PNG')
  print("saved 3+ images")
```

GPT-5-think Vision Tool Call Output

Task ID: 68658a7116039839194326l6

-line 24, 27, 32, 35, 38, 41: saved six transformed images in one tool call

```
from PIL import Image, ImageEnhance, ImageFilter, ImageOps
import numpy as np
import os

# Load the source image
src_path = '/mnt/efs/.../ymHX_cj7Vg3lDk8.jpg'
im = Image.open(src_path)
# Step 1: Auto-orient and slightly crop borders
im = ImageOps.exif_transpose(im)
# Step 2: Enhance brightness and contrast
bright = ImageEnhance.Brightness(im).enhance(1.8)
contrast = ImageEnhance.Contrast(bright).enhance(2.0)
# Step 3: Convert to grayscale for clarity
gray = ImageOps.grayscale(contrast)
# Step 4: Increase sharpness
sharp = ImageEnhance.Sharpness(gray).enhance(2.5)
# Step 5: Apply slight denoise via median filter, then unsharp mask
median = sharp.filter(ImageFilter.MedianFilter(size=3))
unsharp = median.filter(ImageFilter.UnsharpMask(radius=2, percent=200,
    threshold=3))

# Save the enhanced full image
save_dir = 'eval_results_single_turn/...'
os.makedirs(save_dir, exist_ok=True)
unsharp.save(os.path.join(save_dir, 'transformed_image_0.png'), 'PNG')
# Also create a higher-contrast inverted version which sometimes helps
inv = ImageOps.invert(unsharp)
inv.save(os.path.join(save_dir, 'transformed_image_1.png'), 'PNG')
# Crop likely totals area: lower half of the long receipt\nw, h = im.size
# Long receipt is on right; crop right area
right_x0 = int(w*0.35)
right = unsharp.crop((right_x0, 0, w, h))
right.save(os.path.join(save_dir, 'transformed_image_2.png'), 'PNG')
# Crop bottom of right receipt where totals usually appear
bottom_right = unsharp.crop((right_x0, int(h*0.55), w, h))
bottom_right.save(os.path.join(save_dir, 'transformed_image_3.png'), 'PNG')
# Crop mid section (items + discounts)
mid_right = unsharp.crop((right_x0, int(h*0.30), w, int(h*0.60)))
mid_right.save(os.path.join(save_dir, 'transformed_image_4.png'), 'PNG')
# Crop the smaller left slip (maybe PWD details)
left = unsharp.crop((0, int(h*0.1), int(w*0.38), int(h*0.7)))
left.save(os.path.join(save_dir, 'transformed_image_5.png'), 'PNG')
print('Saved enhanced and cropped images.')"
```

F.4 Tool APIs

In this section, we provide more details on the tool API description.

Tool Description: python_image_processing

```
{
    "type": "function",
    "function": {
```

```
        "name": "python_image_processing",
        "description": (
            "Generate arbitrary Python code for image manipulation and save the transformed
            ↪  image as PNG.\n"
            f"-Read one source image (your choice) from the working-directory file list:
            ↪  {image_list}.\n"
            f"-Perform any image processing with PIL, NumPy, or OpenCV. You cannot use
            ↪  matplotlib to show the image.\n"
            f"-You **must save** the transformed image as PNG to {processed_image_save_path}
            ↪  using the filename pattern "
            "\"transformed_image_i.png\", where the counter **i starts at 0 and increments on
            ↪  each invocation** "
            "so files are never overwritten. Example:\n"
            f"    img.save(f\"{processed_image_save_path}/transformed_image_{{i}}.png\",
            ↪  \"PNG\")\n"
        ),
        "parameters": {
            "type": "object",
            "properties": {
                "code": {
                    "type": "string",
                    "description": "Python code to run.",
                    "minLength": 1,
                    "maxLength": 5000
                }
            },
            "required": ["code"]
        }
    }
}
```

Tool Description: python_processing

```
{
    "type": "function",
    "function": {
        "name": "python_interpreter",
        "description": (
            "General-purpose Python interpreter. Run arbitrary Python code and capture stdout
            ↪  via print(). "
            "Any exceptions are returned in stderr.\n\n"
            "Pre-installed packages:\n"
            "  • numpy\n"
            "  • pandas\n"
            "  • requests\n"
            "  • scipy\n"
            "  • scikit-learn\n"
            "  • simpy\n"
            "  • tabulate\n"
            "  • beautifulsoup4\n"
            "  • yfinance"
        ),
        "parameters": {
            "type": "object",
            "properties": {
                "code": {
                    "type": "string",
                    "description": "Python code to run.",
                    "minLength": 1,
                    "maxLength": 5000
                }
            },
            "required": ["code"]
        }
    }
}
```

Tool Description: web_search

```
{
    "type": "function",
    "function": {
        "name": "web_search",
        "description": (
            "Perform a Google search and return relevant results. "
            "Useful for finding current information, news, or facts about topics."
        ),
        "parameters": {
            "type": "object",
            "properties": {
                "query": {
                    "type": "string",
                    "description": "The search query to look up"
                },
                "num_results": {
                    "type": "integer",
                    "description": "Number of results to return (1-10)",
                    "default": 5
                }
            },
            "required": ["query"]
        }
    }
}
```

Tool Description: browser_get_page_text

```
{
    "type": "function",
    "function": {
        "name": "browser_get_page_text",
        "description": (
            "Fetch a web page and extract its text content. "
            "Useful for reading articles, documentation, or any web page content."
        ),
        "parameters": {
            "type": "object",
            "properties": {
                "url": {
                    "type": "string",
                    "description": "The URL of the web page to fetch"
                }
            },
            "required": ["url"]
        }
    }
}
```

Tool Description: historical_weather

```
{
    "type": "function",
    "function": {
        "name": "historical_weather",
        "description": (
            "Get historical weather data for a specific location and date. "
            "Useful for analyzing past weather patterns or events."
        ),
        "parameters": {
            "type": "object",
            "properties": {
                "location": {
                    "type": "string",
```

```
                    "description": "City name or coordinates (e.g., 'New York, NY' or
                    ↪  '40.7128,-74.0060')"
                },
                "date": {
                    "type": "string",
                    "description": "Date in YYYY-MM-DD format"
                }
            },
            "required": ["location", "date"]
        }
    }
}
```

Tool Description: calculator

```
{
    "type": "function",
    "function": {
        "name": "calculator",
        "description": "A calculator tool that can perform basic arithmetic operations
        ↪  including +, -, *, /, %, ^, sqrt, sin, cos, tan, log, exp, and parentheses.",
        "parameters": {
            "type": "object",
            "properties": {
                "expression": {"type": "string", "description": "The expression to evaluate,
                ↪  e.g. \"2 * 3.14 * 5\"."}
            },
            "required": ["expression"]
        }
    }
}
```

G More Benchmark Examples

Below, we present additional benchmark examples covering both single-turn and multi-turn settings across all five task categories. Each example includes the task prompt, image input, golden answer, reference tool trajectory, and evaluation rubrics.

Single-Turn, Generalist, Hybrid-Reasoning

Prompt: Check out the menu from this new coffee shop I'm meeting a date at. I want to have a hot coffee ready for him when he gets here. Give me a suggestion for someone who doesn't like steamed milk or sweet-tasting drinks. I don't want it to be the cheapest type of coffee on the menu.



Golden Answer: That's a great way to start your date! Looking at the menu, an Americano would meet your requirements. It doesn't contain steamed milk, and it's not sweet. At $4, it's not the cheapest item on the menu, either. Let me know if you need any other suggestions!

Reference Tool Trajectory:

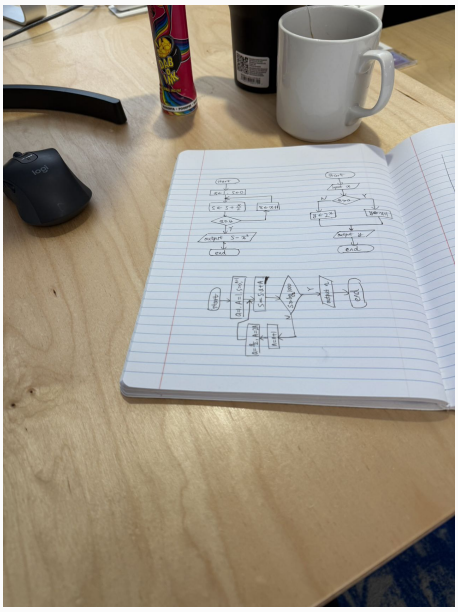
1. `python_image_processing`: Load image to enhance it by flipping horizontally.
2. `python_image_processing`: Crop the image for items that qualify as coffee and are not the cheapest on the menu.
3. `web_search`: Perform a web search to identify which types of coffee contain steamed milk and are sweet-tasting.

Rubrics:

Description	Weight	Type	Category	Critical
Suggests Americano coffee as the beverage.	5	Objective	Reasoning, Instruction following, Truthfulness, Visual Understanding	Yes
States that the Americano is not sweet and does not contain milk.	5	Objective	Instruction following, Truthfulness	Yes
States that the Americano is not the cheapest coffee on the menu	4	Objective	Visual Understanding	No

Single-Turn, STEM, Region Switch Q&A

Prompt: For the top-left flowchart, what operation updates 'S' inside the loop, and what condition terminates it? Also, for the top-right flowchart, what are the two possible calculations for 'y', and what condition decides which one is performed?



Golden Answer: The flowchart on the top left defines a loop. The process inside this loop updates the variable 'S' with the specific mathematical operation $S \leftarrow S + x/2$. The loop is controlled by the diamond shape, which sets the exact termination condition to $x \geq 4$. The flowchart on the top-right describes a conditional process based on the input 'x.'. The diamond shape, $x > 0$, is the condition that determines which path is taken. If the condition is true, the calculation performed is $y \leftarrow x + 1$; if it is false, the alternate calculation performed is $y \leftarrow 2^x$.

Reference Tool Trajectory:

1. `python_image_processing`: Crop the top-left flowchart and read the text inside the loop to identify the iterative operation on the 'S' variable ($S \leftarrow S + x/2$) and the text in the diamond shape to identify the loop's termination condition ($x \geq 4$).

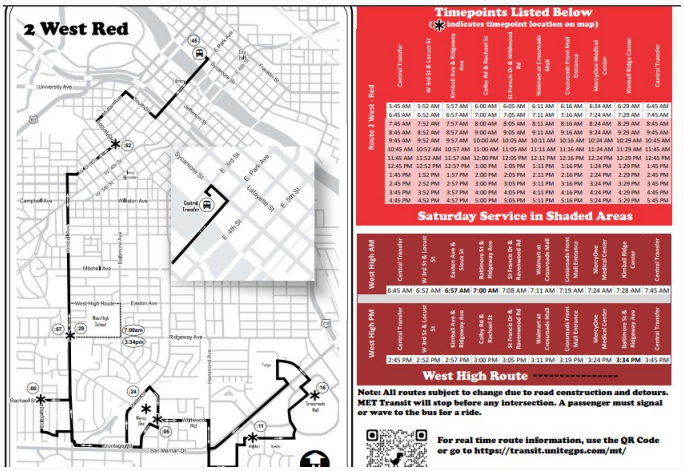
2. `python_image_processing`: Crop the the top-right flowchart and read the text in the diamond to find the deciding condition ($x > 0$) and trace its "Yes" and "No" paths to find the two possible calculations for the output 'y' ($y \leftarrow x + 1$ and $y \leftarrow 2^x$).

Rubrics:

Description	Weight	Type	Category	Critical
States that the operation updating S inside the loop is $S \leftarrow S + x/2$.	5	Objective	Reasoning, Instruction following, Truthfulness, Visual Understanding	Yes
States that the loop terminates when $x > 4$.	5	Objective	Reasoning, Instruction following, Truthfulness, Visual Understanding	Yes
The response states both possible calculations for the output y as $y \leftarrow x + 1$ and $y \leftarrow 2^x$.	4	Objective	Reasoning, Instruction following, Truthfulness, Visual Understanding	Yes
The response states that the deciding condition from the top-right flowchart is $x > 0$.	3	Objective	Reasoning, Instruction following, Truthfulness, Visual Understanding	No

Single-Turn, Generalist, Region Switch Q&A

Prompt: I need to get to Crossroads Mall entrance from the Central Transfer station by 8:30. When do I need to be ready to be picked up? Also, which road will the route follow for most of the trip?



Golden Answer:

- You will need to be picked up at the Central Transfer station at 7:45 AM, since this bus arrive at Crossroads Mall by 8:16.
- The route mainly follows Kimball Avenue from the image.

Reference Tool Trajectory:

1. `python_image_processing`: Zoom-in the bus schedule area, find a bus leaving the Central Transfer station to arrive at Crossroads Mall by 8:30. Also use this tool to identify the main road the route follows..
2. `python_image_processing`: Crop the map area to find the main road name for the bus route.

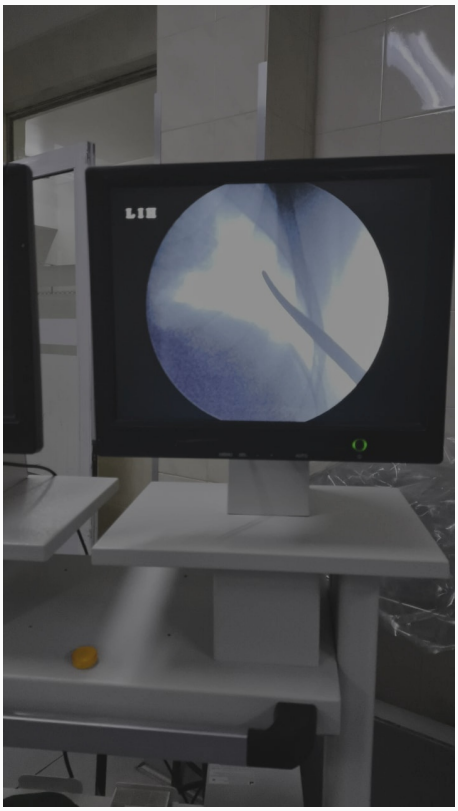
Rubrics:

Description	Weight	Type	Category	Critical
The model identifies the main road the route follows as Kimball Avenue.	5	Objective	Reasoning, Instruction following, Truthfulness, Visual Understanding	Yes
The response states that the user should be picked up at 7:45 AM to catch the bus arriving at Crossroads Mall entrance by 8:30 AM.	5	Objective	Reasoning, Instruction following, Truthfulness, Visual Understanding	Yes

Multi-Turn, Medicine, Temporal Sequential Reasoning

Turn 1

Prompt: The medical team is treating a young patient admitted to the emergency room with a diagnosis of pneumothorax after a MVA. During surgery, a chest drain is placed under fluoroscopic guidance. Based on this image, describe the correct trajectory and placement of the instruments. Also, describe the criteria for confirming the safety of the procedure, as well as the possible complications of improper drain placement.



Golden Answer: The image shows the introduction of scissors into the thoracic cavity of a patient with pneumothorax. The instrument's path is carefully directed toward the pleural area, taking great care to avoid direct contact with central structures such as the heart. The orientation is appropriate, as it follows a lateral and slightly superior path, allowing safe access to the pleural space. Correct placement and safety criteria:

- The instrument is directed toward the pleural cavity and not the mediastinum.
- It is kept away from the cardiac silhouette and large vessels.
- The progressive opening of the tract facilitates subsequent placement of the drainage tube.

Possible complications of incorrect placement:

1. Cardiac or large vessel injury (cardiac tamponade, massive hemothorax).
2. Lung injury with subcutaneous emphysema or hemothorax.
3. Ineffective extrapleural placement.

In conclusion, the image shows a technically correct procedure in the scissors dissection phase, maintaining safety by avoiding vital structures before definitive placement of the chest drain.

Reference Tool Trajectory:

1. `python_image_processing`: Apply a brightness and contrast adjustment to clearly display the image.
2. `python_image_processing`: Apply a crop operation to extract only the area of the monitor containing the image.
3. `web_image_processing`: Search for information about the pneumothorax intervention procedure and its potential complications from improper drain placement.

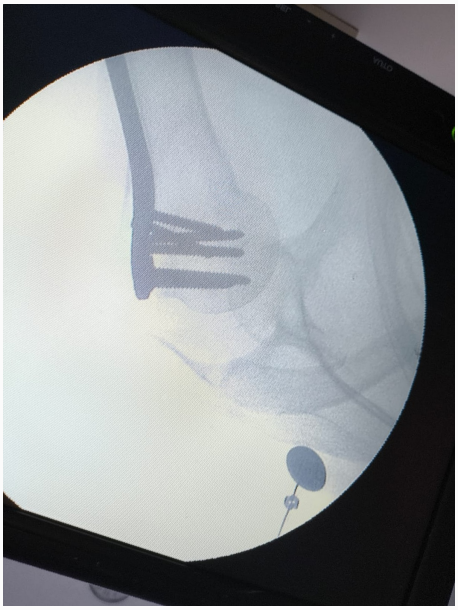
Rubrics:

Description	Weight	Type	Category	Critical
Identify that the chest drain orientation follows a lateral and slightly superior path to safely access the pleural space.	5	Objective	Instruction following, Truthfulness, Visual Understanding	Yes
Identify safety criteria: instrument trajectory into the pleural cavity while avoiding the cardiac silhouette and large vessels.	5	Objective	Reasoning, Instruction following, Truthfulness	Yes
Identify possible complications of improper drain placement, including cardiac or vessel injury, lung injury, and ineffective extrapleural placement.	4	Objective	Reasoning, Instruction following, Truthfulness	Yes

==========

Turn 2

Prompt: The patient also has a fracture due to the MVA, which was treated along with the pneumothorax. The patient returns for routine follow-up after surgical fixation of a fracture. The radiograph shows the surgical site with the internal fixation hardware in place. Identify the position of the fracture and the hardware holding it. Also, identify the progress of bone healing and if there are any potential complications, such as infection, loosening, hardware migration, or malunion.



Golden Answer: The follow-up radiograph of the proximal humerus shows internal fixation with a plate and multiple screws appropriately placed in the fracture. The hardware is intact, with no signs of loosening, migration, or fracture. The fracture is still slightly appreciable but shows early union with callus formation, consistent with a normal healing process at this stage. There is no evidence of infection, osteolysis, or malunion. Overall, the postoperative outcome of the proximal humerus fracture is favourable, with correct alignment and satisfactory progression.

Reference Tool Trajectory:

1. `python_image_processing`: Apply a flip operation to flip it vertically and extract it in the correct orientation.

2. `python_image_processing`: Apply a rotation operation using the estimated coordinates to extract only the laptop screen area with the imaging findings.

Rubrics:

Description	Weight	Type	Category	Critical
Identifies absence of infection and other complications.	5	Objective	reasoning, truthfulness, visual understanding	Yes
Identify that the proximal humerus fracture is treated with internal fixation hardware (a plate and multiple screws) appropriately placed.	5	Objective	visual understanding	Yes
Identifies that callus is forming at the fracture site.	4	Objective	Reasoning, Visual Understanding	Yes
Identify that this is a shoulder radiograph.	2	Objective	Visual Understanding	No
Identifies that the radiograph image quality is sufficient for evaluation	1	Objective	Visual Understanding	No

Multi-Turn, Generalist, Follow-up Test

Turn 1

Prompt: Shown is a map of multiple train lines throughout a city, with each station/stop marked by a white dot (with intersecting lines indicating where one can transfer lines). Suppose I start at the northmost station of Line Seven; then, without using external sources, what is the least number of required stops to reach the southmost station of Line One? Give all the reasoning necessary to solve the question.



Golden Answer: I am sorry, but it appears the numbers corresponding to each train line are not listed in the image given. To determine both the northmost station of Line Seven, as well as the southmost station of Line One, the line number corresponding to each color (or at least the colors associated with Lines One and Seven) must be given. Can you please provide this information?

Reference Tool Trajectory:

1. `python_image_processing`: Flip the image horizontally, and increase brightness and contrast as needed.
2. `python_image_processing`: Locate the train line map on the screen of the computer in the upper/middle portion of the image. Identify that the line numbers are not listed/are missing from the picture.

Rubrics:

Description	Weight	Type	Category	Critical
The model identifies that the line numbers corresponding to each color are not provided in the image and are necessary to solve the problem.	5	Objective	Visual Understanding	Yes
The model asks a follow-up clarification question requesting the mapping between line numbers and their colors for Lines One and Seven.	4	Objective	Instruction following	Yes

==========

Turn 2

Prompt: That's my bad, I forgot to provide you with the line numbers on the map. Can you now please solve the same question? Shown is a map of multiple train lines throughout a city, with each station/stop marked by a white dot (with

intersecting lines indicating where one can transfer lines). Suppose I start at the northmost station of Line Seven; then, without using external sources, what is the least number of required stops to reach the southmost station of Line One? Give all the reasoning necessary to solve the question.



Golden Answer: Using the compass at bottom right (north is up), identify Line 1 (yellow) and Line 7 (light green). The northernmost Line 7 station is Malvern Centre and the southernmost Line 1 station is Union. The minimum-stop route is: Malvern Centre → Neilson → west via Washburn → Sheppard McCowan → Kennedy → south to Warden → Pape → Garrard → Queen → Union, avoiding longer branches via Brenyon or Ionview. Counting stops on this path (excluding the starting station) gives 25.

Reference Tool Trajectory:

1. `python_image_processing`: Flip the image horizontally, and increase brightness and contrast as needed.
2. `python_image_processing`: Locate the train line map on the computer screen in the middle/upper portion of the image. Further, zoom into the upper portion of the screen to identify the train line color pairing as: light green (Line Seven) and yellow (Line One). Next, locate the compass symbol near the bottom right portion of the screen, indicating that the top of the image is the northmost portion. Finally, identify all possible station stops (marked as white dots) between the northmost station of Line Seven (Malvern Centre) and the southmost station of Line One (Union).

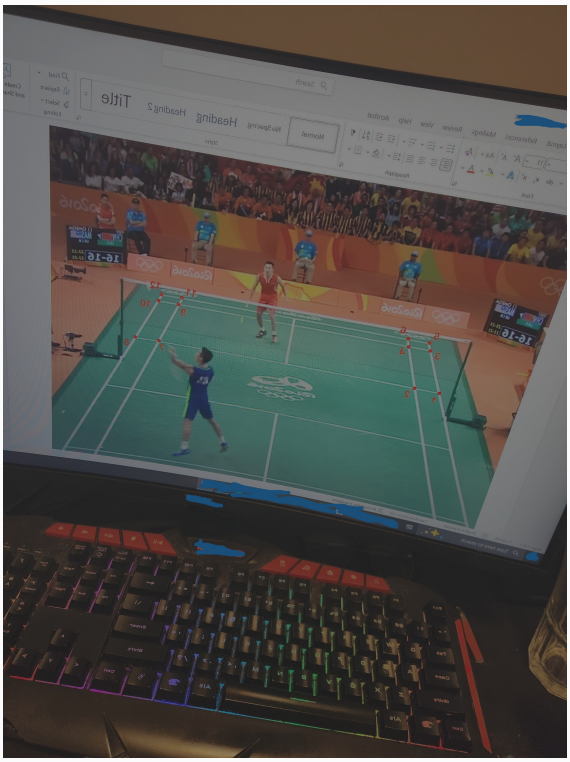
Rubrics:

Description	Weight	Type	Category	Critical
Model correctly identifies the northmost station of Line Seven as Malvern Centre.	5	Objective	Reasoning, Truthfulness, Visual Understanding	Yes
Model correctly identifies the southmost station of Line One as Union Station.	5	Objective	Reasoning, Instruction following, Truthfulness, Visual Understanding	Yes
The model states that the minimum number of stops required when travelling from Malvern Centre to Union Station is 25.	4	Objective	Reasoning, Instruction following, Truthfulness, Visual Understanding	Yes
The model outlines a path listing stations from Malvern Centre to Union Station that follows the specified route, allowing equivalent station names.	3	Objective	Reasoning, Instruction following, Truthfulness, Visual Understanding	No

Multi-Turn, Sports, Progressive Visual Reasoning

Turn 1

Prompt: A badminton tournament during the 2016 Rio Olympics between Lin Dan (in red) and Lee Chong Wei (in blue) is shown, which is a best-of-three-game series. The middle of a round is currently in play. Who is currently hitting/about to hit the shuttlecock, and how do you know? Further, if the shuttlecock is missed by the opponent after being hit by the player from the answer in the first question, what numbered point on the ground labelled in red will be the farthest point away from the receiver (player who missed), such that the shuttlecock is not considered out?



Golden Answer: Who is hitting/about to hit the shuttlecock? Since the player in blue is jumping, this is a clear indication that he is currently hitting/about to hit the shuttlecock. Further, if one looks more carefully, a white shuttlecock can be identified near the top right of the blue player's racket. Thus, the blue player (Lee Chong Wei) is currently hitting/about to hit the shuttlecock.
What point labelled in red is farthest away from the receiving player if he misses the shot made by Lee Chong Wei, not considered out? Since the game shown is between two players (singles), any shots where the shuttlecock lands in the farthest left or farthest right rectangles are considered out (everything else is considered in, including the backmost rectangle). Thus, this eliminates the points numbered 1, 3, 5, 8, 10, and 12 as possible answers. Of the remaining points, the point numbered 6 is furthest away from Lin Dan (red player). Thus, the point labelled 6 is the farthest point away from the receiving player (Lin Dan) considered in, assuming he misses the shot made by Lee Chong Wei.

Reference Tool Trajectory:

1. `python_image_processing`: Flip the image horizontally.
2. `python_image_processing`: Adjust the brightness and contrast of the image.

Rubrics:

Description	Weight	Type	Category	Critical
The model correctly identifies Lee Chong Wei (in blue) as the player currently hitting or about to hit the shuttlecock.	5	Objective	Instruction following, Truthfulness, Visual Understanding, Reasoning	Yes
The model justifies its identification by referencing either the player's posture/position or the visible shuttlecock location near the racket.	4	Objective	Visual Understanding, Reasoning	Yes
The model must reason that the game shown is a singles match between two players.	3	Objective	Reasoning	No
The model must state that the farthest most left and farthest most right rectangles are considered out if the shuttlecock lands there.	3	Objective	Reasoning	No
Identify the 12 points labelled in red (1 through 12) on Lin Dan's side of the court.	1	Objective	instruction following, Visual Understanding	No
State that the shuttlecock is out if it lands in any of the labelled points 1, 3, 5, 8, 10, or 12.	2	Objective	instruction following, reasoning, truthfulness	No
State that point number 6 is the farthest allowed point from Lin Dan still considered in..	5	Objective	Instruction following, Truthfulness, Visual Understanding, Reasoning	Yes
Response includes explicit reasoning rather than only providing a ground truth answer.	3	Objective	Reasoning, Presentation	No

==========

Turn 2

Prompt: Suppose now, instead of hitting the shuttlecock to the point labelled 6 on the ground, Lee Chong Wei instead hits the shuttlecock to the point labelled 8 on the ground. Assuming Lin Dan does not receive (hit) the shuttlecock and lets it hit the ground, what will the scoreboard read after this point? State the player's name associated with the number of points as well.

Golden Answer: Current score according to the scoreboard: From the scoreboard, it reads 16-16, where the left number is the score associated with the team from China (from the flag shown), whereas the right number is the score associated with the team from Malaysia (from the flag shown).
New score, assuming the shuttlecock lands at the point labelled 8: Since the game is a singles match, the outer left and outer right-most rectangles are considered out. Since the point labelled 8 is in the outer right-most rectangle, this shot made by Lee Chong Wei would be considered out. Since Lin Dan did not receive the shot as stated by the prompt, Lin Dan will win this round, increasing the score to 17-16. (Here, it is important to note that the left number on the scoreboard has increased, and not the right). Further, Lin Dan will have 17 points, while Lee Chong Wei will still have 16 points.

Reference Tool Trajectory:

1. `python_image_processing`: Identify the current match score as 16-16 (middle top right of the image), where the left number is associated with the score of the team from China, whereas the right number is associated with the score of the team from Malaysia, according to the flags.

2. `web_search`: Look up and determine that Lin Dan represents the team from China, whereas Lee Chong Wei represents the team from Malaysia.

Rubrics:

Description	Weight	Type	Category	Critical
Identify the current score as 16-16.	3	Objective	Truthfulness, Visual Understanding	Yes
Associate the left number with the team from China and the right number with the team from Malaysia based on the flags.	1	Objective	Truthfulness, Visual Understanding	No
Model correctly assigns Lin Dan to the left scoreboard number (team from China) and Lee Chong Wei to the right scoreboard number (team from Malaysia).	2	Objective	Reasoning, Visual Understanding	No
Model correctly reasons that because the shuttlecock landing point labelled 8 is out in singles matches and Lin Dan does not hit it back, Lin Dan receives the point	5	Objective	Reasoning, Visual Understanding	Yes
The model states the new score after Lin Dan receives the point as: 17-16 (17 for Lin Dan, and 16 for Lee Chong Wei).	5	Objective	Reasoning	Yes
The model states the new score after Lin Dan receives the point as: 17-16 (17 for Lin Dan, and 16 for Lee Chong Wei).	5	Objective	Truthfulness, Instruction Following	Yes
The reasoning explains why point 8 is considered out.	4	Objective	Reasoning, Presentation	Yes
The reasoning explains why the left number on the scoreboard is associated with Lin Dan and not Lee Chong Wei.	1	Objective	Reasoning, Presentation	No

==========

Turn 3

Prompt: Now, since the tournament is a best-of-three-game series, if Lee Chong Wei in fact does hit the shuttlecock out at the point labelled 8 on the ground as discussed, how many additional points will he need to win the match (assuming the score does not tie at 20-20)? If Lee Chong Wei wins this match/game, is he guaranteed to win the series? If so, what were the scores of the previous two matches played against each other?

Golden Answer: How many points does Lee Chong Wei need to win the match after shooting the shuttlecock out? The new score after Lee Chong Wei hits the shuttlecock out is 17-16 (17 for Lin Dan, 16 for Lee Chong Wei). Since a standard badminton game is up to 21 points, and assuming the score does not tie to 20-20 (in which case you can only win if you get two points back-to-back), Lee Chong Wei thus needs 21-16=5 points more to win.
Scores of the previous two matches: One can deduce that the previous two games have already been finished from the scoreboard, where it states that Lin Dan won one game with a final score of 21-15, whereas Lee Chong Wei won the other game with a score of 11-21. Thus, the game being played currently is the final game to determine the winner of the three-game series. Hence, if Lee Chong Wei wins this match, he is guaranteed to win the series.

Reference Tool Trajectory:

1. `python_image_processing`: Identify the scores of the previous two matches/games in the middle top right of the image as: 21-15 (for Lin Dan), and 11-21 (for Lee Chong Wei).

2. `web_search`: Lookup the amount of points needed to win a standard badminton game as 21 (assuming a 20-20 point tie does not occur).

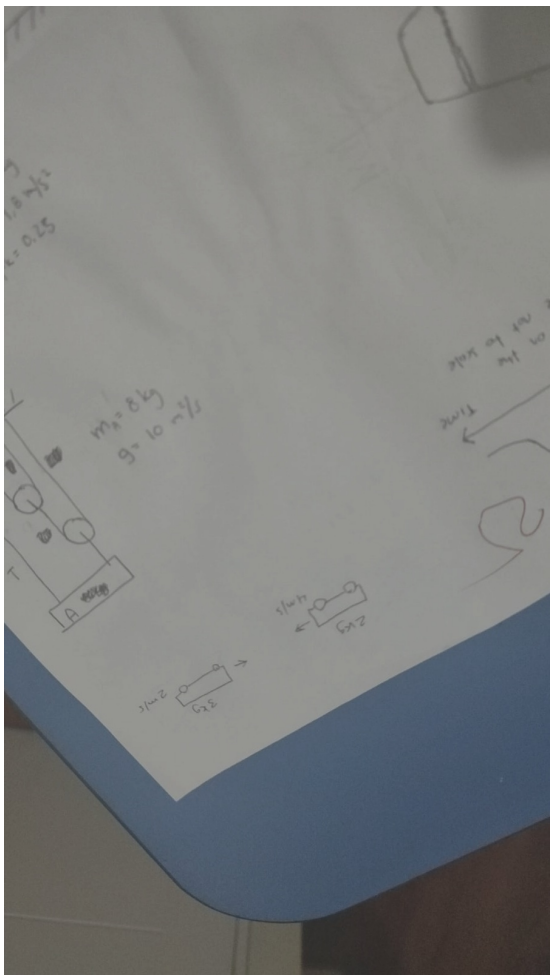
Rubrics:

Description	Weight	Type	Category	Critical
Identifies the final scores of the previous two matches as 21-15 (Lin Dan) and 11-21 (Lee Chong Wei).	3	Objective	Instruction following, Truthfulness, Visual Understanding	No
The model reasons that Lee Chong Wei must reach a total point count of 21 to win the match (assuming a 20-20 tie does not occur).	4	Objective	Reasoning	Yes
The model correctly calculates that Lee Chong Wei needs 5 additional points to win the current match.	5	Objective	Reasoning, Instruction following, Truthfulness	Yes
The model correctly states that if Lee Chong Wei wins this match, he is guaranteed to win the best-of-three series and provides the scores of the previous two matches.	5	Objective	Reasoning, Presentation	Yes
The response provides intermediate reasoning that addresses each question posed by the prompt: (1) how many additional points Lee Chong Wei needs to win the match, and (2) whether he is guaranteed to win the series and what the scores of the previous two matches were.	2	Objective	Reasoning, Presentation	No

H Model Success Examples

H.1 GPT-5 Success Example

Prompt: Can you help me with the car problem shown in the image? It gives the initial conditions. What will be the final speed of the cars if they stick together after the perfectly inelastic collision with no friction?



Original Image

Full model trajectory with intermediate output:

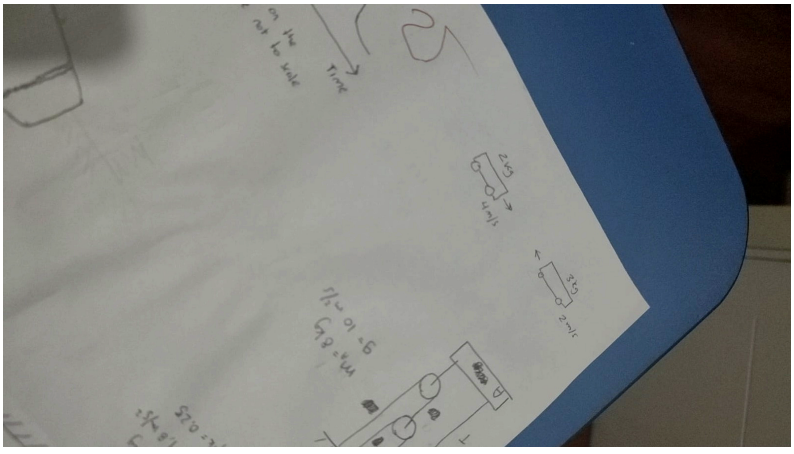
- **Model Intermediate Output:** I'll zoom into the small drawings and text so I can read the masses and velocities. I'll rotate, crop, and enhance the bottom region where the two cars and their speeds are drawn. Then I'll compute the final speed using momentum conservation.

- **Tool Call 1**

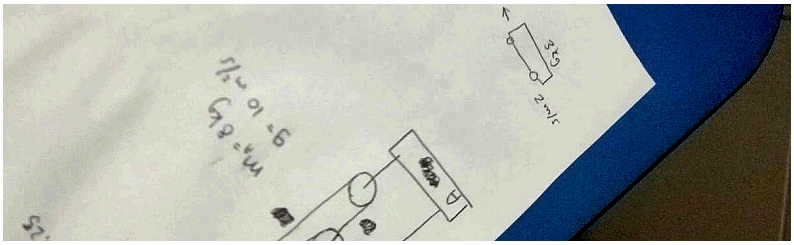

```
{
  "function_name": "python_image_processing",
  "function_args": {
```

```
        "code": "from PIL import Image, ImageOps, ImageEnhance, ImageFilter\nimport os\n\n# Load
        ↪  the source image\nsrc_path = '../B5OYlWps8AlthY3.jpg'\nimg =
        ↪  Image.open(src_path)\n\n# Step 1: Rotate so text is more horizontal. The image seems
        ↪  tilted; try rotating 90 degrees counterclockwise first.\nrot = img.rotate(90,
        ↪  expand=True)\n\n# Step 2: Enhance contrast/sharpness\nrot_enh =
        ↪  ImageEnhance.Contrast(rot).enhance(1.6)\nrot_enh =
        ↪  ImageEnhance.Sharpness(rot_enh).enhance(2.0)\n\n# Save first transformed
        ↪  image\nout_dir = 'eval_results_single_turn/openai/gpt-5_w_tool_system_high_max_tool
        ↪  _calls_20_trial_1/chunk_28_task_6870357d15624e798bfc83bf'\nos.makedirs(out_dir,
        ↪  exist_ok=True)\nrot_enh.save(os.path.join(out_dir, 'transformed_image_0.png'),
        ↪  'PNG')\n\n# Step 3: Crop the bottom center area where small car sketches are located
        ↪  (based on visual estimation).\nw, h = rot_enh.size\n# Crop lower-middle area\ncrop =
        ↪  rot_enh.crop((int(0.2*w), int(0.55*h), int(0.95*w), int(0.95*h)))\n# Further
        ↪  enhance\ncrop = ImageEnhance.Brightness(crop).enhance(1.2)\ncrop =
        ↪  ImageEnhance.Contrast(crop).enhance(1.7)\ncrop =
        ↪  ImageEnhance.Sharpness(crop).enhance(2.5)\n\ncrop.save(os.path.join(out_dir,
        ↪  'transformed_image_1.png'), 'PNG')\n\n# Step 4: Another crop focusing even more on
        ↪  the small cars at the lower edge\nw2, h2 = crop.size\ncrop2 = crop.crop((int(0.0*w2),
        ↪  int(0.40*h2), int(0.9*w2), int(0.95*h2)))\ncrop2 =
        ↪  ImageEnhance.Contrast(crop2).enhance(1.4)\ncrop2 =
        ↪  ImageEnhance.Sharpness(crop2).enhance(2.0)\n\ncrop2.save(os.path.join(out_dir,
        ↪  'transformed_image_2.png'), 'PNG')\n\n# Step 5: A crop on left small car\nw3, h3 =
        ↪  crop2.size\nleft_car = crop2.crop((int(0.0*w3), int(0.15*h3), int(0.35*w3),
        ↪  int(0.9*h3)))\nleft_car = ImageEnhance.Contrast(left_car).enhance(1.6)\nleft_car =
        ↪  ImageEnhance.Sharpness(left_car).enhance(2.0)\nleft_car.save(os.path.join(out_dir,
        ↪  'transformed_image_3.png'), 'PNG')\n\n# Step 6: A crop on right small car\nright_car
        ↪  = crop2.crop((int(0.38*w3), int(0.05*h3), int(0.9*w3), int(0.9*h3)))\nright_car =
        ↪  ImageEnhance.Contrast(right_car).enhance(1.6)\nright_car =
        ↪  ImageEnhance.Sharpness(right_car).enhance(2.0)\nright_car.save(os.path.join(out_dir,
        ↪  'transformed_image_4.png'), 'PNG')\n\nprint('done', rot_enh.size, crop.size,
        ↪  crop2.size)"
    },
    "observation": {
        "ok": "true",
        "stdout": "",
        "stderr": "",
        "output_path": "../transformed_image_0.png",
        "output_paths": [
            "../transformed_image_0.png",
            "../transformed_image_3.png",
            "../transformed_image_1.png",
            "../transformed_image_2.png",
            "../transformed_image_4.png"
        ]
    }
}
```

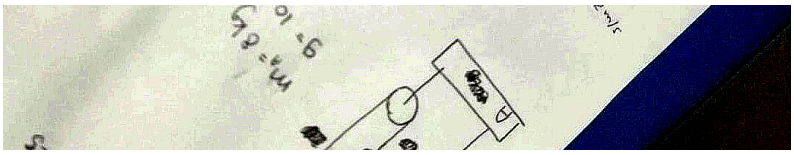

Processed Images



Transformed Image 0



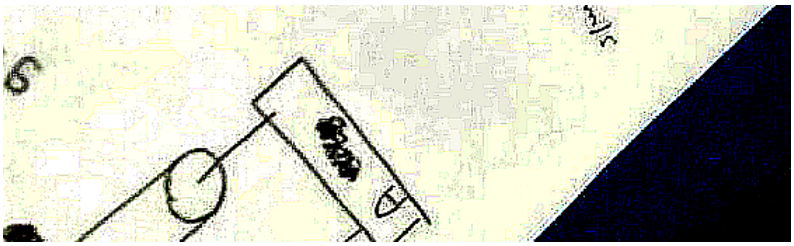
Transformed Image 1



Transformed Image 2



Transformed Image 3



Transformed Image 4

Model Answer. From the image:

Car A has a mass of 2 kg and is moving to the right at 4 m/s. Car B has a mass of 3 kg and is moving to the left at 2 m/s. They stick together (perfectly inelastic collision), and with no friction, momentum is conserved.

To compute the final velocity v:

$$v = \frac{m_1 v_1 + m_2 v_2}{m_1 + m_2} = \frac{2 \cdot (+4) + 3 \cdot (-2)}{2 + 3} = \frac{8 - 6}{5} = \frac{2}{5} = 0.4 \text{ m/s to the right.}$$

Answer: 0.4 m/s to the right.

Judge Results.

1.
   - **Rubric Content**: Identifies that the speed and mass of one of the cars are 4 m/s and 2 kg
   - **Explanation**: The model's answer explicitly states that Car A has mass 2 kg and speed 4 m/s, satisfying the criterion.
   - **Met or Not Met**: Met

2.
   - **Rubric Content**: Identifies that the speed and mass of the other car are 2 m/s (in the opposite direction) and 3 kg
   - **Explanation**: The model answer explicitly identifies the other car as 3 kg moving at 2 m/s in the opposite direction.

- **Met or Not Met**: Met

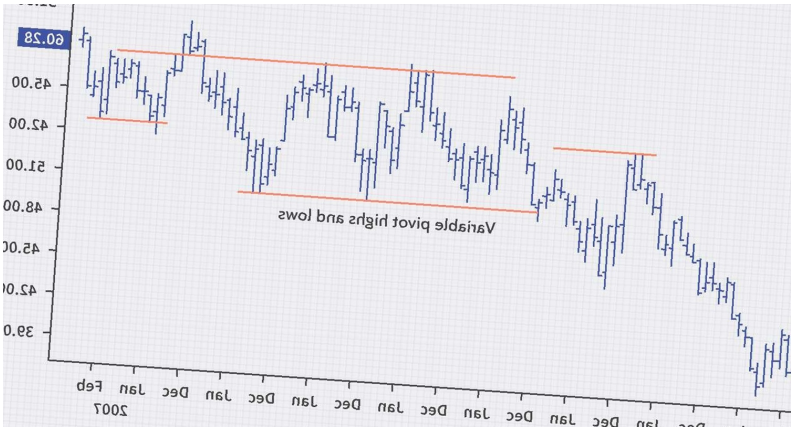
3.
   - **Rubric Content**: The response states that the final speed of the cars is 0.4 m/s in the same direction as the car that was initially moving at 4 m/s.
   - **Explanation**: The model answer explicitly gives the final speed as 0.4 m/s to the right, matching the direction of the initially faster car (4 m/s to the right).
   - **Met or Not Met**: Met

Overall Score: 1.0
Accuracy: 1

Insights. Although the model successfully solved the task, it performed unnecessary image-processing operations. Future work should focus not only on improving final-answer accuracy but also on enhancing the efficiency of the model's tool use.

H.2 Claude-Opus-4.1 Success Example

Prompt: How many orange lines acting as a resistance in its span are drawn? How many orange lines acting as a support in its span are drawn? Is the current price of the stock shown in the image at a key level support or key level resistance?



Original Image

Tool-call trajectory with intermediate output:

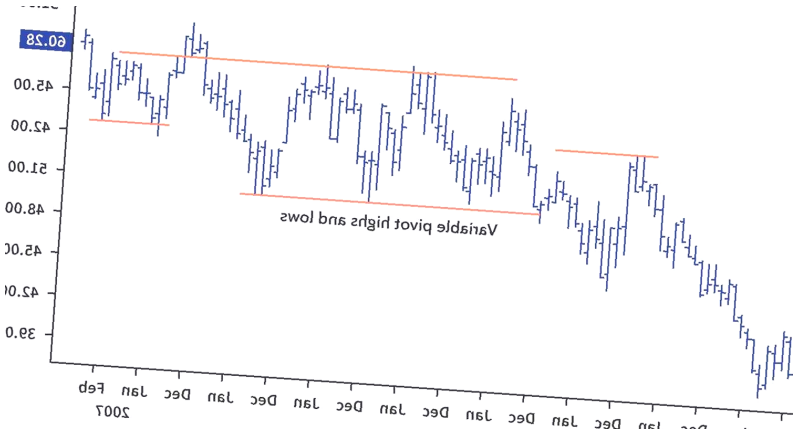
- **Model Intermediate Output**
  I'll analyze this stock chart to identify the orange support and resistance lines and determine the current price position. Let me first enhance the image to get a clearer view of the details.

- **Tool Call 1**


```
{
    "function_name": "python_image_processing",
    "function_args": {
        "code": "from PIL import Image\nimport numpy as np\n\n# Load the image\nimg =
        ↪  Image.open('../2Fe-HqOtQKYj54l.jpg')\n\n# Convert to RGB if needed\nif img.mode !=
        ↪  'RGB':\n    img = img.convert('RGB')\n\n# Enhance contrast and brightness for better
        ↪  visibility\nimg_array = np.array(img)\n\n# Increase contrast\nimg_array =
        ↪  np.clip(img_array * 1.3 - 30, 0, 255).astype(np.uint8)\n\n# Convert back to PIL
        ↪  Image\nenhanced_img = Image.fromarray(img_array)\n\n# Save the enhanced
        ↪  image\nenhanced_img.save('../transformed_image_0.png', 'PNG')\n\nprint(\"Enhanced
        ↪  image saved\")"
    },
    "observation": {
        "ok": "true",
        "stdout": "Enhanced image saved\n",
        "stderr": "",
        "output_path": "../transformed_image_0.png",
```

```
        "output_paths": [
            "../transformed_image_0.png"
        ]
    }
}
```


Processed Images



Transformed Image 0

Model Answer. Now I can clearly see the stock chart. Let me analyze the orange lines and their roles as support or resistance, as well as identify the current price position.

Orange Lines Acting as Resistance. Two orange lines function as resistance over their spans:

1. The top descending orange line (starting around the 60.28 level) — this line acts as resistance throughout its span, as the price stays below it.

2. The middle descending orange line (around the 48.00–51.00 level) — this also acts as resistance for most of its duration.

Orange Lines Acting as Support. Two orange lines function as support:

1. The short horizontal orange line on the left (around the 42.00 level) — acts as support in its brief span.

2. The horizontal orange line on the right (around the 51.00 level) — acts as support for recent price action.

Current Price Position. The current price shown in the blue box is **60.28**. From the chart:

- The current price appears to be **at or very near a key resistance level**.
- It is testing the upper descending orange resistance line.
- The price has risen from lower levels but is now confronting this significant resistance zone.

Thus, the current price of 60.28 is at a **key resistance level**, not a support level. This represents a critical technical zone where the stock encounters selling pressure from the descending trendline resistance.

Golden Answer. There are 2 orange lines acting as a resistance in its span. There are 2 orange lines acting as a support in its span. The current price of 60.28 is currently at a key level resistance.

Judge Results.

1.
   - **Rubric Content**: The response states that there are 2 orange lines acting as a resistance in its span.
   - **Explanation**: The model answer clearly states that there are 2 orange lines acting as resistance in their spans.

- **Met or Not Met**: Met

2. - **Rubric Content**: The response states that there are 2 orange lines acting as a support in its span.
   - **Explanation**: The model explicitly states that there are 2 orange lines acting as support in their spans, satisfying the rubric criterion.
   - **Met or Not Met**: Met

3. - **Rubric Content**: The response states that the current price is at a key level resistance.
   - **Explanation**: The model explicitly states that the current price of 60.28 is at a key level resistance.
   - **Met or Not Met**: Met

4. - **Rubric Content**: The response states that the current price is 60.28.
   - **Explanation**: The model's answer explicitly states that the current price shown is 60.28, satisfying the rubric criterion.
   - **Met or Not Met**: Met

Overall Score: 1.0

Accuracy: 1

Insights. In some cases, the model can correctly identify key visual content even without rotating the image into a human-readable orientation.

H.3 Gemini-2.5-pro Success Example

Prompt: I'm traveling from Monumento station to Antipolo station. What station from my current line should I get off at if I want to go through exactly one interchange only? What station is in the interchange of the yellow line and the LRT 2 line? My friend is at Calamba station, and he's going to Madrigal station. What is the color of the line he should transfer to in order to get to Madrigal station in the least number of stations passed?



Original Image

Tool-call trajectory with intermediate output:

- **Tool Call 1**


```
{
  "function_name": "python_image_processing",
  "function_args": {
    "code": "\nfrom PIL import Image\nimport numpy as np\n\ni = 0\nimg =
    ↪  Image.open('../qVnlVdcHA0Etj5Q.jpg')\nflipped_img = img.transpose(Image.FLIP_LEFT_RIG
    ↪  HT)\nflipped_img.save(f\"../transformed_image_{i}.png\",
    ↪  \"PNG\")\n"
  },
  "observation": {
    "ok": "true",
    "stdout": "",
    "stderr": "",
    "output_path": "../transformed_image_0.png",
    "output_paths": [
      "../transformed_image_0.png"
    ]
  }
}
```


Processed Images



Transformed Image 0

Model Answer. Here are the answers to your questions based on the Metro Manila Rail Transit Map:

1. To get from Monumento station to Antipolo station with only one interchange, you should get off at **Doroteo Jose** station. From there, you can transfer to the LRT 2 line at Recto station to continue your journey to Antipolo.

2. The interchange station for the yellow line (NSCR) and the LRT 2 line (Purple) is **Tutuban** station.

3. Your friend traveling from Calamba station to Madrigal station should transfer to the **Pink** line (MRT 4) at Alabang station. This is the most direct route with the least number of stations.

Golden Answer. If you are traveling from Monumento Station to Antipolo Station and you want to go through only one interchange, then the station that you should get off at your current line is Doroteo Jose Station.

The station that is located at the interchange of the yellow line and the LRT 2 line is Tutuban Station.

If your friend is traveling from Calamba Station to Madrigal Station, and your friend wants to pass through the fewest number of stations, then your friend should transfer to the pink line (at the first possible opportunity).

Judge Results.

1. - **Rubric Content**: The response states that if the user is traveling from Monumento Station to Antipolo Station and wants to go through only one interchange, then the station that the user should get off at (on the user's current line) is Doroteo Jose Station.
   - **Explanation**: The model's answer explicitly states that the user should get off at Doroteo Jose station for only one interchange, matching the rubric criterion.
   - **Met or Not Met**: Met

2. - **Rubric Content**: The response states that the station that is located at the interchange of the yellow line and the LRT 2 line is Tutuban Station.

- **Explanation**: The model's answer explicitly states that the interchange station for the yellow line and the LRT 2 line is Tutuban station.
- **Met or Not Met**: Met

3.
- **Rubric Content**: The response indicates that to minimize the number of stations passed from Calamba Station to Madrigal Station, the friend should transfer to the pink line.
- **Explanation**: The model's answer clearly states that the friend should transfer to the Pink line to minimize the number of stations passed from Calamba to Madrigal.
- **Met or Not Met**: Met

Overall Score: 1.0
Accuracy: 1

I Model Failure Examples

I.1 GPT-5 Failure Example

Prompt: In this game between Kasparov and Karpov, white has just played Kxh6. Can black recapture with the g7-pawn? Which piece is threatened by the knight on e4?



Original Image

Tool-call trajectory with intermediate output:

- **Model Intermediate Output**
  I'll zoom in on the chessboard in the center to read the position clearly before answering.

- **Tool Call 1**


```
{
    "function_name": "python_image_processing",
    "function_args": {
        "code": "from PIL import Image\nimport os\n\n# Load the image\nimg_path =
        ↪  '../qZerK_Jz50p11e7.jpg'\nimg = Image.open(img_path)\n\n# The chessboard is
        ↪  centered; compute a crop around the central square area.\nw, h = img.size\n# Based on
        ↪  visual inspection, crop a central region where the board is located\nleft =
        ↪  int(w*0.33)\nupper = int(h*0.06)\nright = int(w*0.67)\nlower =
        ↪  int(h*0.56)\nboard_crop = img.crop((left, upper, right, lower))\n\n# Save the cropped
        ↪  image\nsave_dir = '../chunk_21_task_6883f999d3f384b2a9a3e641'\nos.makedirs(save_dir,
        ↪  exist_ok=True)\nboard_crop.save(os.path.join(save_dir, 'transformed_image_0.png'),
        ↪  'PNG')\n\nprint('Cropped image saved, size:', board_crop.size)"
    },
    "observation": {
        "ok": "true",
        "stdout": "Cropped image saved, size: (435, 360)\n",
        "stderr": "",
        "output_path": "../transformed_image_0.png",
        "output_paths": [
            "../transformed_image_0.png"
        ]
    }
}
```


- ...

- **Tool Call 5**

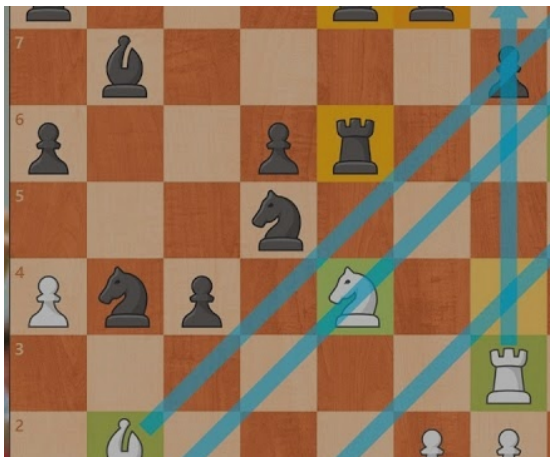

```
{
    "function_name": "browser_get_page_text",
    "function_args": {
        "url": "https://chesstempo.com/game-database/game/1856030"
    },
    "observation": {
        "url": "https://chesstempo.com/game-database/game/1856030",
        "title": "Garry Kasparov vs Anatoly Karpov (1990)",
        "text": "Garry Kasparov vs Anatoly Karpov (1990)",
        "status": "success",
        "content_length": 39
    }
}
```

- ...

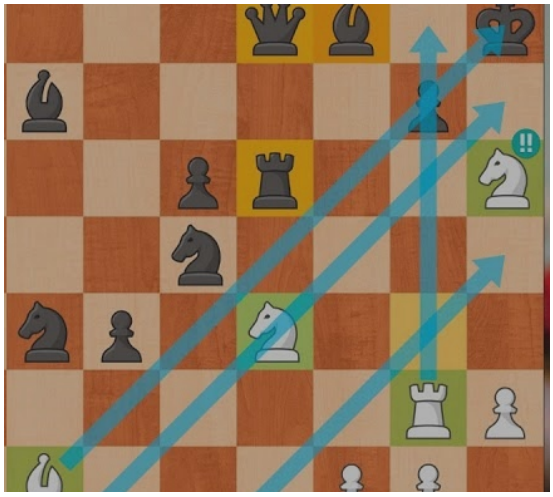
- **Tool Call 12**


```
{
    "function_name": "google_search",
    "function_args": {
        "query": "Kasparov Karpov Nxh6!! gxh6 Qg8+ which game",
        "num_results": 5
    },
    "observation": {
        "results": [
            {
                "title": "World Chess Championship 1990",
                "snippet": "The World Chess Championship 1990 was played between Garry Kasparov
                ↪  and Anatoly Karpov. It was the fifth and final Kasparov\u2013Karpov
                ↪  championship match.",
                "url": "https://en.wikipedia.org/wiki/World_Chess_Championship_1990",
                "display_link": "https://en.wikipedia.org \u203a wiki \u203a
                ↪  World_Chess_Champi..."
            },
            ...
            {
                "title": "Kasparov-Karpov: the Secret of the 24th Game",
                "snippet": "This article focuses on the decisive moment of their fourth match:
                ↪  the 24th and final game, played in Seville in 1987.",
                "url":
                ↪  "https://en.chessbase.com/post/kasparov-karpov-the-secret-of-the-24th-game",
                "display_link": "https://en.chessbase.com \u203a post \u203a
                ↪  kasparov-karpov-the-se..."
            }
        ],
        "query": "Kasparov Karpov Nxh6!! gxh6 Qg8+ which game",
        "status": "success",
        "total_results": 7980
    }
}
```

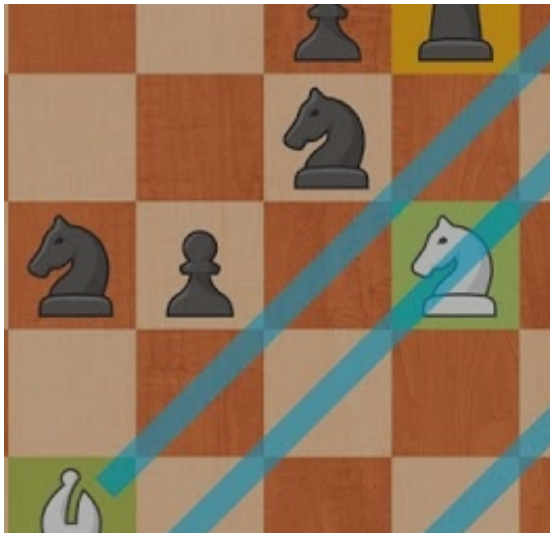

Processed Images



Transformed Image 0



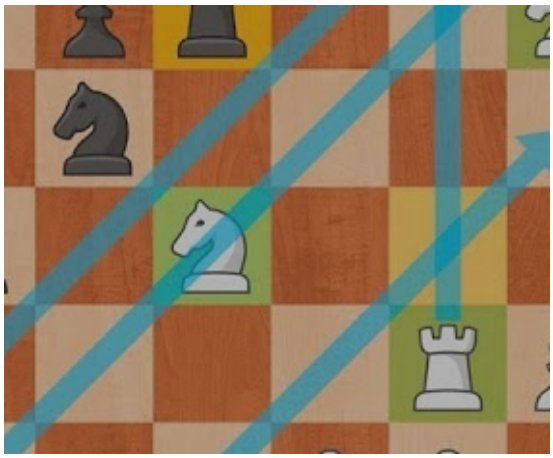
Transformed Image 1



Transformed Image 2



Transformed Image 3



Transformed Image 4

Model Answer: - No. If ...gxh6? then White plays Rxg8+ Kxg8 and Nf6+ with a royal fork, winning Black's queen. - The knight on e4 is threatening Black's queen on e8 (via Nf6+).

Golden Answer: The bishop on b2 pins the g7-pawn to the black king, so it cannot capture the knight. The e4 knight is threatening black's b6 pawn.

Judge Results.

1.
   - **Rubric Content**: The model states that black cannot recapture with the g7-pawn because it is pinned by the bishop on b2.
   - **Explanation**: The model answer explicitly states that the g7-pawn is pinned along the b2–h8 diagonal by White's bishop on b2, matching the rubric criterion.
   - **Met or Not Met**: Met

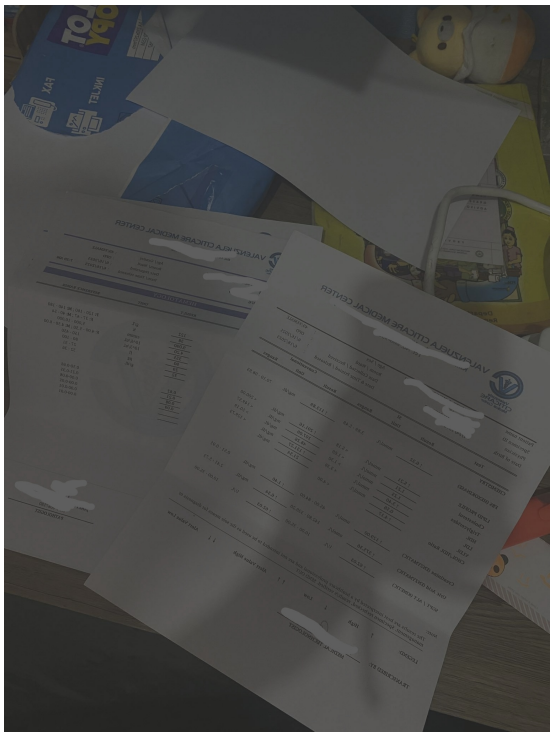
2.
   - **Rubric Content**: The model identifies that the e4-knight is threatening the d6-pawn.
   - **Explanation**: The model answer says the knight on e4 is attacking the rook on f6, not the d6-pawn as required by the rubric.
   - **Met or Not Met**: Not Met.

Overall Score: 0.5.

Accuracy: 0.

I.2 Claude-Opus-4.1 Failure Example

Prompt: For the results that were released on June 18, 2025, which tests have shown high results? How much greater (in %) is the Cholesterol result compared to the ceiling value for the Cholesterol test based solely on the contents of the image?



Original Image

Tool-call trajectory with intermediate output:

- **Model Intermediate Output**
  I can see this is a medical test results document, but the image is at an angle and the text is not clearly readable. Let me enhance and rotate the image to better read the specific test results and dates.

- **Tool Call 1**


```
{
    "function_name": "python_image_processing",
    "function_args": {
        "code": "from PIL import Image\nimport numpy as np\n\n# Load the image\nimg =
        ↪  Image.open('../faOCgknfLYxLiGs.jpg')\n\n# Rotate the image to make it more readable
        ↪  (appears to be at an angle)\nimg_rotated = img.rotate(-10, expand=True,
        ↪  fillcolor='white')\n\n# Crop to focus on the medical results document (the main
        ↪  white paper in center)\nwidth, height = img_rotated.size\n# Approximate crop to
        ↪  focus on the document\nleft = width * 0.15\ntop = height * 0.1\nright = width *
        ↪  0.85\nbottom = height * 0.9\nimg_cropped = img_rotated.crop((left, top, right,
        ↪  bottom))\n\n# Save the transformed image\nimg_cropped.save(\"eval_results_single_tu
        ↪  rn/anthropic/claude-opus-4-1_w_tool_system_high_max_tool_calls_20_trial_1/chunk_18_
        ↪  task_68757ac94c044bac23c9be59/transformed_image_0.png\", \"PNG\")\n\nprint(\"Image
        ↪  rotated and cropped to focus on the medical document\")"
    },
    "observation": {
        "ok": "true",
        "stdout": "",
        "stderr": "",
        "output_path": "../transformed_image_0.png",
        "output_paths": [
            "../transformed_image_0.png"
        ]
    }
}
```


- **Model Intermediate Output**
  The image is still not clear enough to read the specific details. Let me further enhance it by focusing on the test results section and improving the contrast.

- **Tool Call 2**

```
{
    "function_name": "python_image_processing",
    "function_args": {
        "code": "from PIL import Image, ImageEnhance, ImageOps\nimport numpy as np\n\n# Load the
        ↪  previously transformed image\nimg = Image.open('../transformed_image_0.png')\n\n#
        ↪  Further crop to focus on the middle section where the test results appear to
        ↪  be\nwidth, height = img.size\nleft = width * 0.1\ntop = height * 0.25\nright = width
        ↪  * 0.9\nbottom = height * 0.75\nimg_cropped = img.crop((left, top, right,
        ↪  bottom))\n\n# Enhance contrast and sharpness\nenhancer =
        ↪  ImageEnhance.Contrast(img_cropped)\nimg_enhanced = enhancer.enhance(1.5)\n\nenhancer
        ↪  = ImageEnhance.Sharpness(img_enhanced)\nimg_enhanced = enhancer.enhance(2.0)\n\n#
        ↪  Resize to make text more readable\nnew_width = int(img_enhanced.width *
        ↪  1.5)\nnew_height = int(img_enhanced.height * 1.5)\nimg_resized =
        ↪  img_enhanced.resize((new_width, new_height), Image.Resampling.LANCZOS)\n\n# Save the
        ↪  transformed image\nimg_resized.save(\"../transformed_image_1.png\",
        ↪  \"PNG\")\n\nprint(\"Image cropped, enhanced and resized to focus on test results\")"
    },
    "observation": {
        "ok": "true",
        "stdout": "",
        "stderr": "",
        "output_path": "../transformed_image_1.png",
        "output_paths": [
            "../transformed_image_1.png"
        ]
    }
}
```


- **Model Intermediate Output**
  I can see the document better now, but I need to focus more on the specific test results and values to answer your question accurately. Let me enhance the image further to clearly read the cholesterol values and reference ranges.

- **Tool Call 3**


```
{
    "function_name": "python_image_processing",
    "function_args": {
        "code": "from PIL import Image, ImageEnhance, ImageOps, ImageFilter\nimport numpy as
        ↪  np\n\n# Load the original image\nimg = Image.open('../faOCgknfLYxLiGs.jpg')\n\n#
        ↪  Convert to grayscale for better text clarity\nimg_gray = img.convert('L')\n\n#
        ↪  Rotate to correct orientation\nimg_rotated = img_gray.rotate(-8, expand=True,
        ↪  fillcolor='white')\n\n# Focus on the center-right portion where the test results
        ↪  appear to be\nwidth, height = img_rotated.size\nleft = width * 0.25\ntop = height *
        ↪  0.35\nright = width * 0.75\nbottom = height * 0.65\nimg_cropped =
        ↪  img_rotated.crop((left, top, right, bottom))\n\n# Apply adaptive
        ↪  enhancement\nimg_cropped = ImageOps.autocontrast(img_cropped)\n\n# Enhance sharpness
        ↪  significantly\nenhancer = ImageEnhance.Sharpness(img_cropped)\nimg_enhanced =
        ↪  enhancer.enhance(3.0)\n\n# Increase contrast\nenhancer =
        ↪  ImageEnhance.Contrast(img_enhanced)\nimg_enhanced = enhancer.enhance(2.0)\n\n#
        ↪  Resize for better readability\nnew_width = int(img_enhanced.width * 2)\nnew_height =
        ↪  int(img_enhanced.height * 2)\nimg_resized = img_enhanced.resize((new_width,
        ↪  new_height), Image.Resampling.LANCZOS)\n\n# Save the transformed
        ↪  image\nimg_resized.save(\"../transformed_image_2.png\", \"PNG\")\n\nprint(\"Image
        ↪  converted to grayscale, cropped to test results section, and enhanced for maximum
        ↪  text clarity\")"
    },
    "observation": {
        "ok": "true",
        "stdout": "Image converted to grayscale, cropped to test results section, and enhanced
        ↪  for maximum text clarity\n",
        "stderr": "",
        "output_path": "../transformed_image_2.png",
        "output_paths": [
            "../transformed_image_2.png"
        ]
    }
}
```

- **Model Intermediate Output**
  Good! Now I can see the test results more clearly. I can see this is a medical test report with various test results. Let me examine the visible values to identify which tests show high results and specifically look for the Cholesterol test details. From what I can see in the enhanced image:

  - The document appears to be dated 18/6/2025 (June 18, 2025)

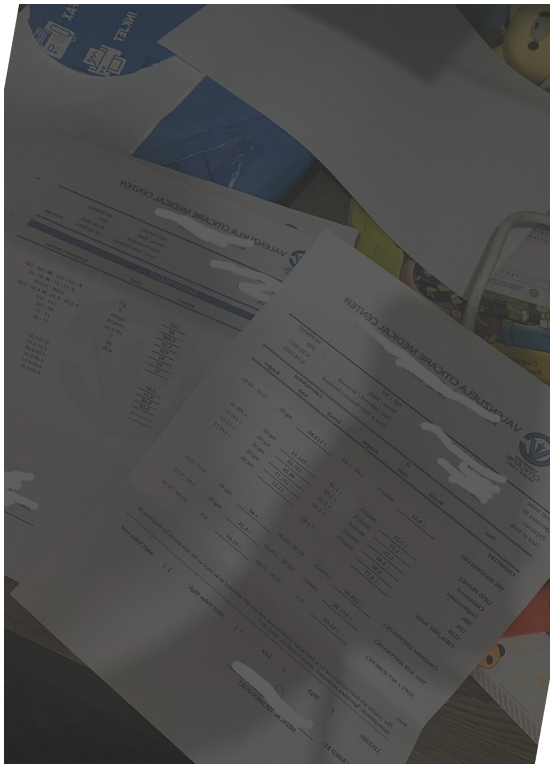
  - I can see several test results with their values and reference ranges

  - There are tests listed including what appears to be cholesterol-related measurements. Let me calculate the percentage for the Cholesterol result:

- **Tool Call 4**

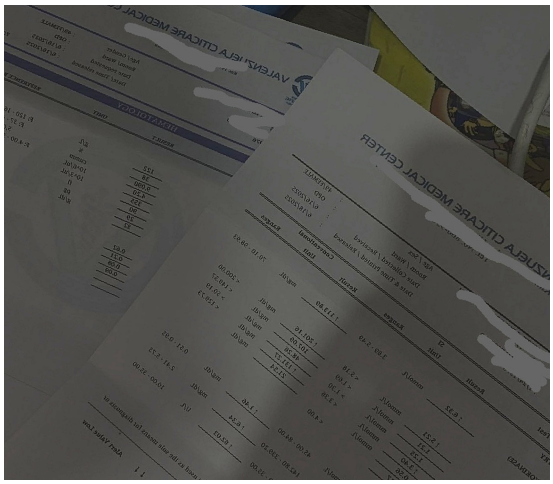

```
{
    "function_name": "calculator",
    "function_args": {
        "expression": "(261 - 200) / 200 * 100"
    },
    "observation": {
        "result": "30.5",
        "expression": "(261 - 200) / 200 * 100",
        "status": "success"
    }
}
```


Processed Images

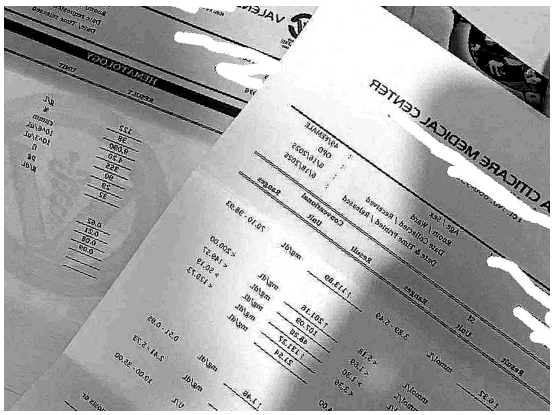


Transformed Image 0

39040



Transformed Image 1



Transformed Image 2

Model Answer: Based on the medical test results document dated June 18, 2025, I can identify the following:

Tests showing HIGH results: 1. **Cholesterol** - The result shows 261 mg/dL, which is marked as HIGH (the reference range appears to have an upper limit of 200 mg/dL)

Cholesterol percentage calculation: The Cholesterol result of 261 mg/dL is **30.5% greater** than the ceiling value of 200 mg/dL for the Cholesterol test.

From what's visible in the document, the Cholesterol test is the primary test showing a high result that exceeds the normal reference range. The result is significantly elevated above the recommended maximum value.

Golden Answer: The tests that were released on June 18, 2025, showing high results are FBS (HEXOKINASE), Cholesterol, LDL, CHOL/HDL Ratio, Creatinine (ENZYMATIC), Uric Acid (ENZYMATIC), and SGPT/ALT (KINETIC). The ceiling range for Cholesterol is given by < 5.18 mmol/L or < 200.00 mg/dL, while the results for the Cholesterol test are 5.21 mmol/L or 201.16 mg/dL. Relative to the ceiling range, the result of 5.31 mmol/L is greater than the ceiling range of 5.18 mmol/L by approximately [(5.31-5.18)/5.18] * 100% $\approx$ 2.51%.

Judge Results:

1.
- **Rubric Content**: The response lists the tests released on June 18, 2025 showing high results as FBS (HEXOKINASE), Cholesterol, LDL, CHOL/HDL Ratio, Creatinine (ENZYMATIC), Uric Acid (ENZYMATIC), and SGPT/ALT (KINETIC).
- **Explanation**: The model answer only lists Cholesterol as showing high results and omits FBS (HEXOKINASE), LDL, CHOL/HDL Ratio, Creatinine (ENZYMATIC), Uric Acid (ENZYMATIC), and SGPT/ALT (KINETIC), so it does not meet the criterion.
- **Met or Not Met**: Not Met.

2.
   - **Rubric Content**: The response does not include any tests other than FBS (HEXOKINASE), Cholesterol, LDL, CHOL/HDL Ratio, Creatinine (ENZYMATIC), Uric Acid (ENZYMATIC), and SGPT/ALT (KINETIC).
   - **Explanation**: The model's answer only mentions Cholesterol, which is on the allowed list, and does not introduce any tests outside the specified set.
   - **Met or Not Met**: Not Met.

3.
   - **Rubric Content**: The response states that the Cholesterol result is approximately 2.51% greater than the ceiling value for the Cholesterol test within a margin of error of 0.01%.
   - **Explanation**: The model answer states that the cholesterol result is 30.5% greater than the ceiling value, rather than approximately 2.51% ±0.01% as required by the rubric criterion.
   - **Met or Not Met**: Not Met.

4.
   - **Rubric Content**: The response states that the ceiling range for Cholesterol is given by < 5.18 mmol/L or < 200.00 mg/dL, while the results for the Cholesterol test are 5.21 mmol/L or 201.16 mg/dL.
   - **Explanation**: The model's answer does not state the ceiling range as <5.18 mmol/L or <200.00 mg/dL nor the result as 5.21 mmol/L or 201.16 mg/dL, instead giving different values (261 mg/dL vs 200 mg/dL). It fails to meet the specified criterion.
   - **Met or Not Met**: Not Met.

5.
   - **Rubric Content**: The response shows the calculation of the relevant percentage by finding the difference of the result and the ceiling value and dividing it by the ceiling value.
   - **Explanation**: The model answer states the percentage (30.5%) but does not show the subtraction (261–200) or the division by 200 step, so it fails to demonstrate the explicit calculation process required by the rubric.
   - **Met or Not Met**: Not Met.

Overall Score: 0.0.
Accuracy: 0.

I.3 Gemini-2.5-pro Failure Example

Prompt: I am currently at the bottom-most station of the orange line, and I am planning to get to Sullivan without transferring to another line. Based solely on the image, what's the first interchange that I will encounter? My mom is at Mass Ave, and she will meet me at Sullivan. Based solely on the image, what is the color of the line that my mom is currently in? At what color of line should my mom initially get on to in order to drop off at the fewest number of interchanges?



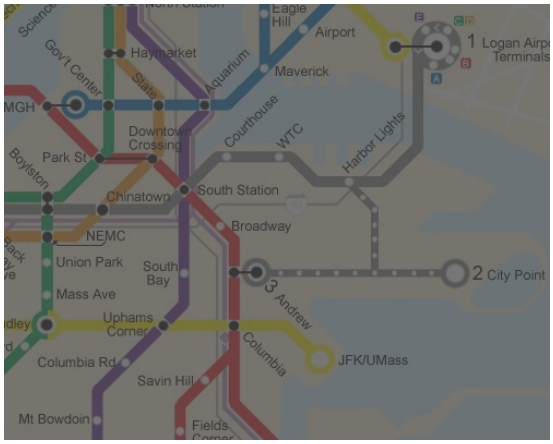
Original Image

Tool-call trajectory with intermediate output:

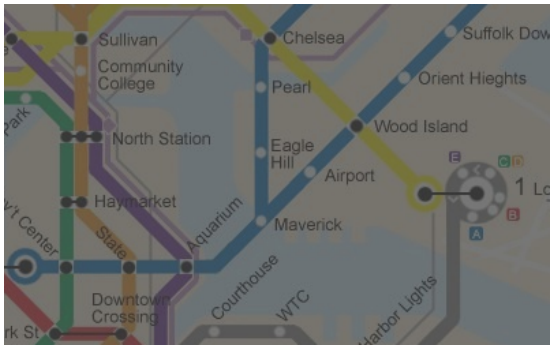
- **Tool Call 1**


```
{
  "function_name": "python_image_processing",
  "function_args": {
    "code": "\nfrom PIL import Image\n\n# Open the original image\nimg =
    ↪  Image.open('../AusOom2bUmenbDg.jpg')\n\n# Crop to the southern part of the Orange
    ↪  Line\n# Coordinates are (left, top, right, bottom)\ncrop_box_1 = (700, 600, 1200,
    ↪  1000)\ncropped_img_1 =
    ↪  img.crop(crop_box_1)\ncropped_img_1.save('../transformed_image_0.png', 'PNG')\n\n#
    ↪  Crop to the area around the two \"Mass Ave\" stations\ncrop_box_2 = (750, 500, 1150,
    ↪  750)\ncropped_img_2 =
    ↪  img.crop(crop_box_2)\ncropped_img_2.save('../transformed_image_1.png', 'PNG')\n\n#
    ↪  Crop to the path from Mass Ave to Sullivan\ncrop_box_3 = (800, 300, 1200,
    ↪  700)\ncropped_img_3 =
    ↪  img.crop(crop_box_3)\ncropped_img_3.save('../transformed_image_2.png', 'PNG')\n"
  },
  "observation": {
    "ok": "true",
    "stdout": "Model inference successful on trial 1\ncalling tool calculator\nfunction_args:
    ↪  {'expression': '38.25 * 0.05203711173860428'}\nobservation: {'result':
    ↪  '1.9904195240016136', 'expression': '38.25 * 0.05203711173860428', 'status':
    ↪  'success'}\n\nRound 7\nUsing model: gemini/gemini-2.5-pro (trial 1/1)\n",
    "stderr": "",
    "output_path": "../transformed_image_0.png",
    "output_paths": [
      "../transformed_image_0.png",
      "../transformed_image_1.png",
      "../transformed_image_2.png"
    ]
  }
}
```

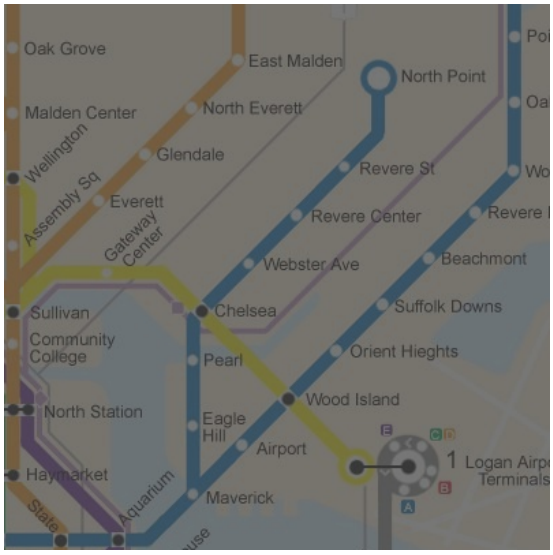

Processed Images



Transformed Image 0



Transformed Image 1



Transformed Image 2

Model Answer: Based on the images provided:

- The first interchange you will encounter traveling from the southernmost station of the Orange Line (Forest Hills) to Sullivan is Downtown Crossing.

- Your mom is at the Mass Ave station on the Orange line.

- To get to Sullivan with the fewest interchanges, your mom should get on the Orange line, which will take her directly to Sullivan with no transfers.

Golden Answer: If you are currently at the bottom-most station of the orange line and are planning to get to Sullivan without transferring to another line, then the first interchange that you will encounter is the interchange at Ruggles.

Since your mom is at Mass Ave, the color of the line that your mom is currently in is green. The color of the line that your mom should initially get on to in order to drop off at the fewest number of interchanges is orange.

Judge Results:

1.
   - **Rubric Content**: The response states that if the user (you) are currently at the bottom-most station of the orange line and are planning to get to Sullivan without transferring to another line, then the first interchange that the user (you) will encounter is the interchange at Ruggles.
   - **Explanation**: The model failed to provide any answer and did not state that the first interchange is Ruggles, so it does not meet the criterion.
   - **Met or Not Met**: Not Met.

2.
   - **Rubric Content**: The response correctly identifies that the user's mom is currently on the green line.
   - **Explanation**: The model answer does not identify that the user's mom is on the green line.
   - **Met or Not Met**: Not Met.

3.
   - **Rubric Content**: The response correctly identifies that the user's mom should get on the orange line to minimize the number of interchanges.
   - **Explanation**: The model identifies the orange line for the mom's route, so it meets the criterion.
   - **Met or Not Met**: Met.

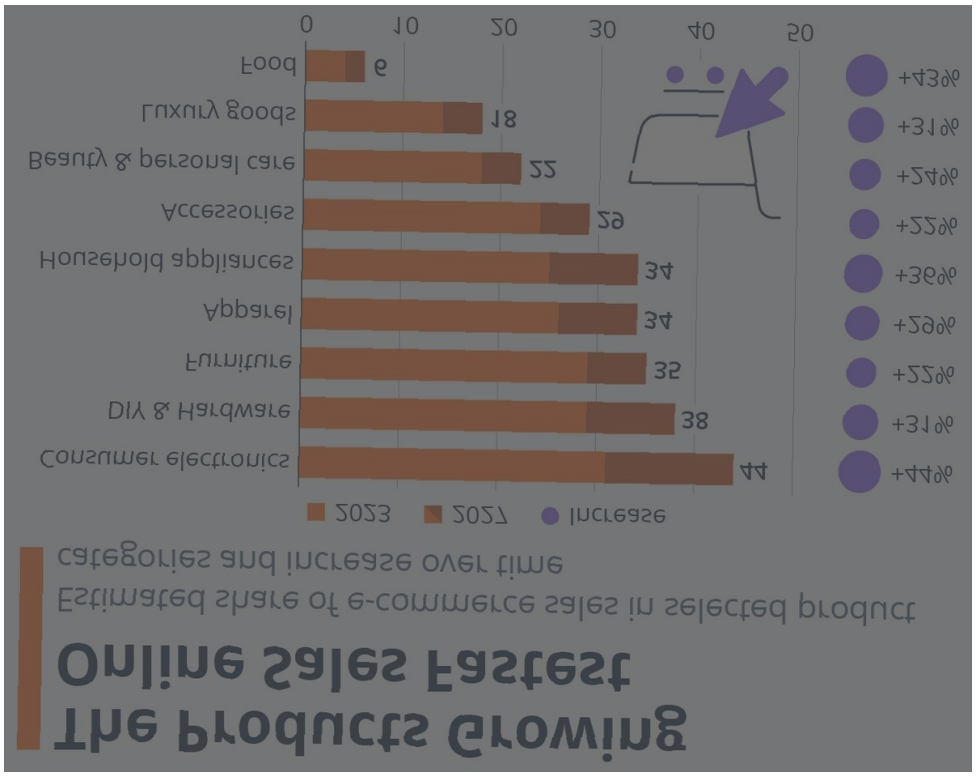
Overall Score: 0.3077.

Accuracy: 0.

J Examples of Processed Images via Vision Tools

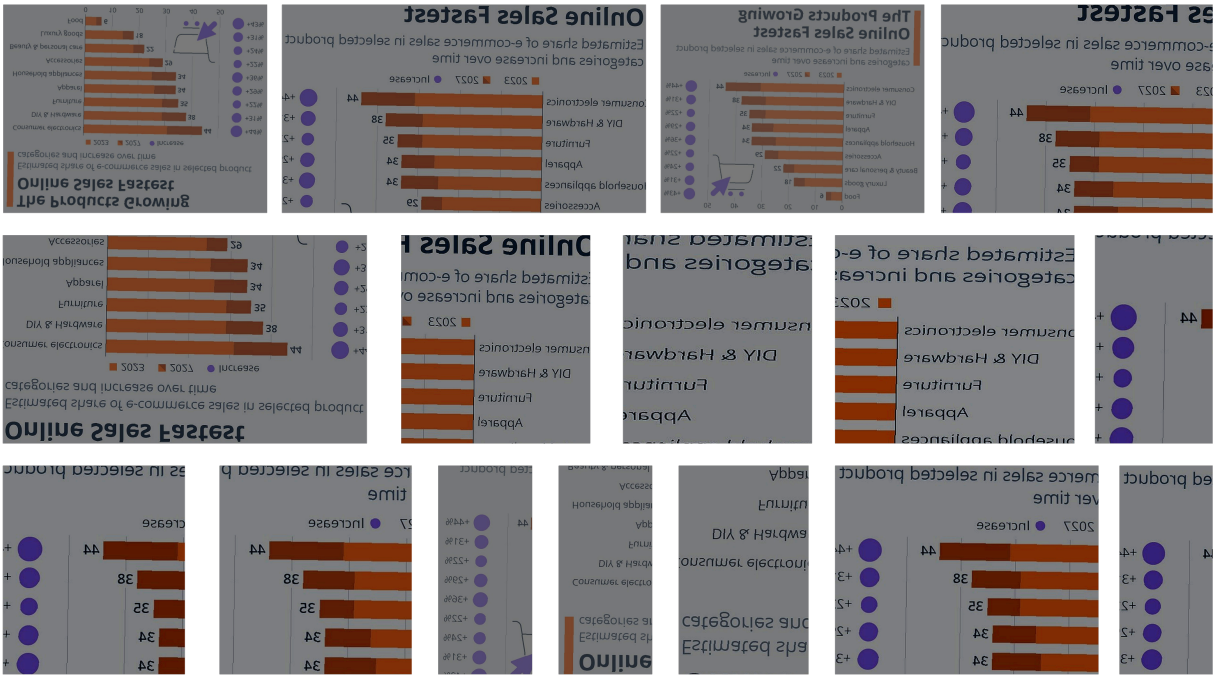
Based on our observation, models sometimes which may produce some unnecessary transformed images. Below we show the representative task prompt, original images, and process images from GPT-5, Gemini-2.5-pro, and Claude-opus-4.1.

Prompt: I need your help to analyze this chart. I need you to list the 3 categories with the highest percentage increase, and the 3 with the lowest. Include each category's percentage increase in the answer.



Original Image

Processed Images



Transformed Images by GPT-5.

Prompt: If the player nearest to the goalkeeper in the image was moved to the 18-yard line instead of his current position, how many opposing players would be in an offside position? Assume that the black team is on defense.



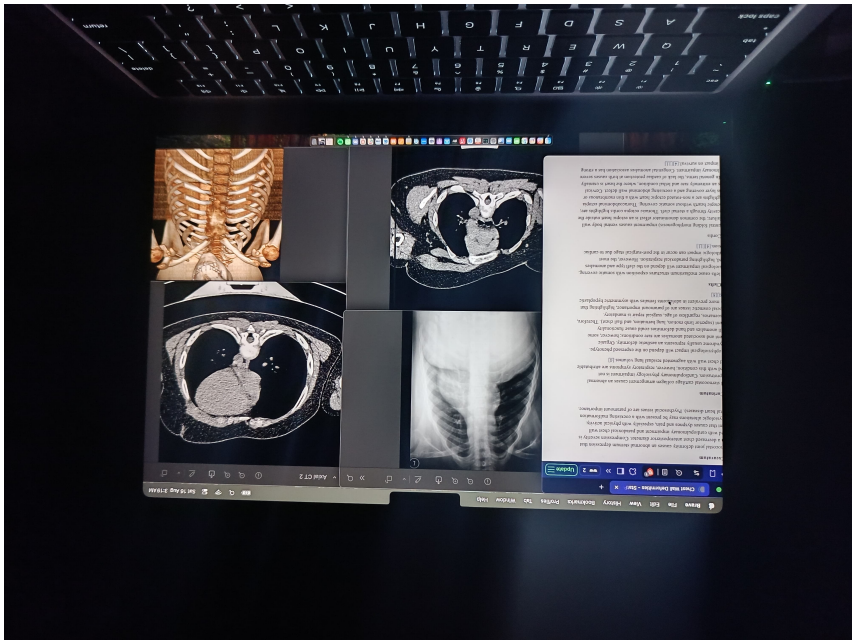
Original Image

Processed Images

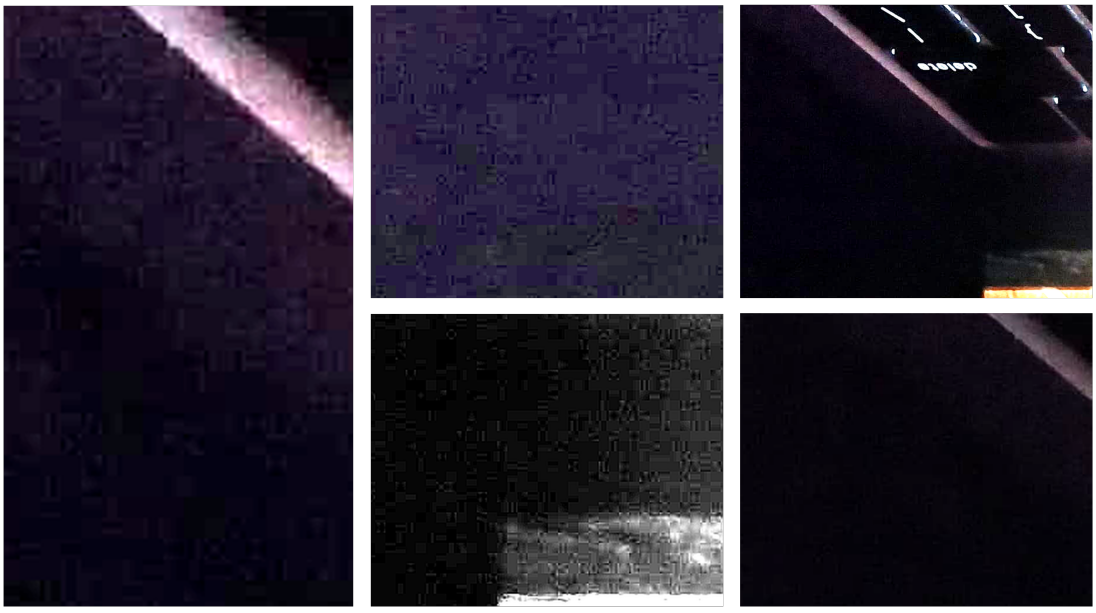


Transformed Images by GPT-5.

Prompt: A young male patient has an abnormal chest deformity. The patient is asymptomatic apart from the chest wall deformity, which has been present since birth. He has no family history of a similar condition. Various radiological imaging techniques were performed to assess the patient's chest deformity. Can you identify the abnormalities seen in the chest X-ray and the 3D volume rendering findings? Also, using the chest CT slides, identify the muscles and structures that are absent.

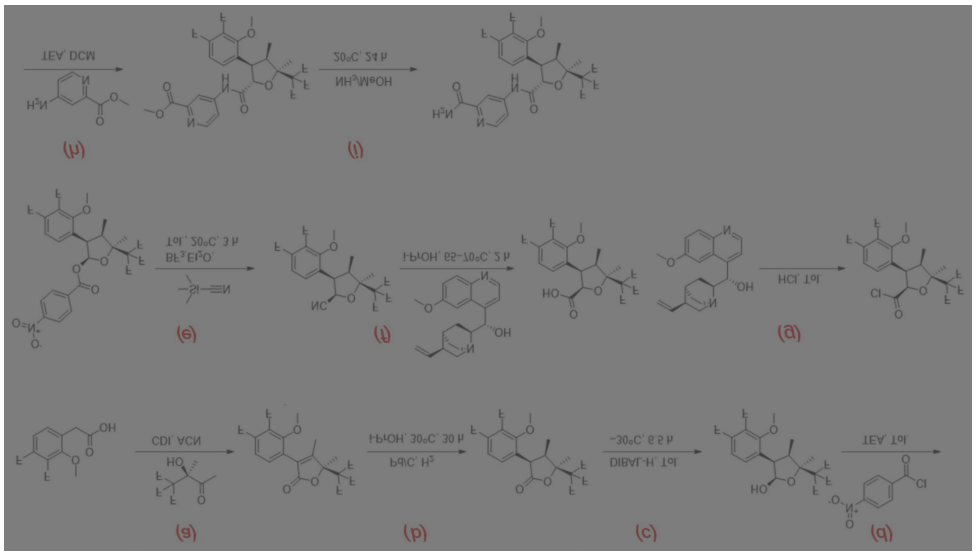


Original Image



Transformed Images by Claude-opus-4.1.

Prompt: What is the molecular formula of the final product of the reaction? Also, what is the molecular formula of the trifluorinated compound employed in the initial reaction step?



Original Image



Transformed Images by Gemini-2.5-pro.