VTC-BENCH: EVALUATING AGENTIC MULTIMODAL MODELS VIA COMPOSITIONAL VISUAL TOOL CHAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advancements extend Multimodal Large Language Models (MLLMs) beyond passive visual perception to actively composing external tools for complex visual tasks. Evaluating these capabilities requires examining how models select visual operations, compose them into multi-step workflows, and interpret intermediate outputs. We introduce VisualToolChain-Bench (VTC-Bench), a benchmark for evaluating multi-step visual tool composition in MLLMs. Our framework features 35 diverse visual operations and 680 curated problems structured across a nine-category functional taxonomy, each paired with reference execution trajectories that serve as diagnostic anchors for fine-grained analysis. Crucially, VTC-Bench supports a dual-paradigm evaluation protocol: a code-driven mode that tests programmatic tool orchestration, and an interface-driven mode that assesses structured tool invocation through predefined tool-sets. This design enables systematic diagnosis of where model capabilities diverge across execution paradigms. Extensive experiments on 19 leading MLLMs reveal critical limitations: models struggle with compositional visual tool use, with the best model achieving 51% accuracy. Multi-tool composition remains a fundamental bottleneck. Models concentrate their calls on a small subset of operations and make limited use of reference tools even when explicitly provided. By identifying these challenges, VTC-Bench provides a rigorous foundation for developing more capable visual agentic models.

1 INTRODUCTION

The rapid evolution (Comanici et al., 2025; Bai et al., 2025; Google, 2025; Hurst et al., 2024; OpenAI, 2025a; Team et al., 2026; GLM-5-Team et al., 2026; Zhang et al., 2025c; ByteDance Seed Team, 2026; Google, 2026; Shi et al., 2025b) of Multimodal Large Language Models (MLLMs) has led to remarkable improvements in foundational capabilities such as visual question answering. Building on this progress, recent advancements have expanded their scope by integrating external tools, transforming these models into active, agentic problem solvers. This tool-use ability empowers MLLMs to move beyond basic image understanding to execute complex visual workflows for enhanced image comprehension. By strategically leveraging specialized external visual tools, MLLMs process information more effectively and complete advanced operations. This integration significantly expands their practical skills, making them more versatile for real-world applications.

To assess these emerging agentic capabilities, several benchmarks (Guo et al., 2025a; Li et al., 2025a; Su et al., 2026) have been introduced to evaluate how effectively current MLLMs utilize visual tools. However, existing frameworks (Su et al., 2026; Guo et al., 2025a) typically rely on limited tool-sets and simple invocations, rarely testing the complex combinations required for advanced visual reasoning. Furthermore, in practical applications, visual agents must dynamically adapt to a highly diverse array of available tools, and resolving real-world tasks often demands chaining multiple distinct operations together to form a successful execution plan.

To bridge this critical gap, we introduce VisualToolChain-Bench (VTC-Bench), a comprehensive benchmark designed to rigorously evaluate the advanced tool-use proficiency of MLLMs on foundational image-based tools. As shown in Fig. 1, to emulate authentic computer vision pipelines, our framework integrates 35 distinct visual operations derived. These operations serve as the essential

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

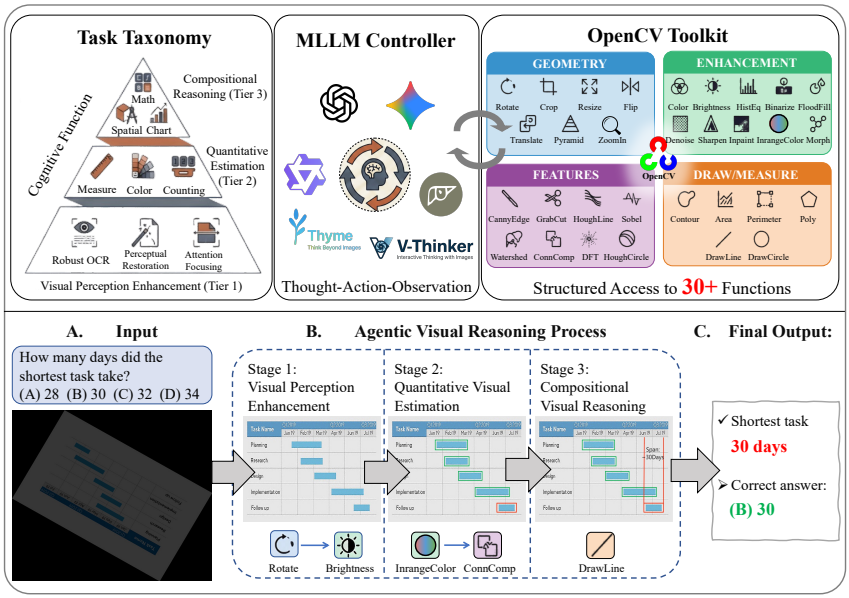


Figure 1: **Overview of VTC-Bench and the agentic reasoning workflow.** The top panel illustrates the framework architecture, featuring a functional task taxonomy and an MLLM-driven toolkit. The bottom panel exemplifies the multi-stage reasoning trajectory, evolving from visual perception enhancement to compositional reasoning.

building blocks for solving complex visual tasks. By leveraging this versatile tool-set, our benchmark naturally supports advanced tool combinations and multi-step reasoning strategies that reflect real-world challenges. To guarantee a thorough assessment of these capabilities, we constructed 680 meticulously designed problems organized into a functional taxonomy of nine tasks grouped into three tiers. Each problem includes an expert-validated reference trajectory for analyzing tool selection and composition. We evaluate task-solving performance through final-answer accuracy and diagnose tool use through execution statistics and process audits.

We evaluate 19 leading MLLMs on VTC-Bench through both code-driven and interface-driven paradigms to systematically diagnose their visual agentic capabilities. Our experiments underscore the highly challenging nature of this benchmark and reveal critical limitations in current models. The overall performance is consistently low, with the leading model Gemini-3.0-Pro achieving only 51.2%. Through dual-paradigm evaluation, we observe divergent capability profiles: closed-source models gain substantially from tool augmentation while open-source models show minimal or even negative improvements, and, counter-intuitively, interface-based invocation proves competitive with code-based execution for several proprietary models. These results highlight a pronounced disparity between the theoretical capability and practical proficiency of state-of-the-art models.

To understand the fundamental limitations of current models, we further conduct detailed diagnostic analyses. Our evaluations reveal that existing models struggle to select and compose visual operations for the task. Furthermore, multi-tool composition remains a persistent obstacle. Models concentrate their calls on a small subset of operations and make limited use of reference tools even when explicitly provided. By systematically exposing these challenges through dual-paradigm diagnostics, VTC-Bench establishes a rigorous baseline to guide the development of more capable visual agents.

2 Related Work

Visual Agentic Model Recent Multimodal Large Language Models (MLLMs) are evolving from static textual reasoning toward a dynamic visual agentic paradigm. Early tool-driven approaches (Yang et al., 2023; Zeng et al., 2023) coordinate external vision experts or fixed APIs for basic visual analysis. To enhance perception, interactive attention mechanisms, such as active zooming (Shen et al., 2025; Zhang et al., 2025a) and visual masking, are employed to refine inputs. Recent reinforcement learning

Table 1: **Comparison of VTC-Bench with representative multimodal benchmarks.** Compared to existing benchmarks, our work encompasses 35 diverse tools and supports dual interaction paradigms (Code & Interface). **RT**: Reference tool-call trajectory. **MTC**: Multi-tool composition for a single task. **FLD**: Function-level dependency, where a function requires intermediate data produced by a preceding function. ✓: supported; ✓̸: partial support; ✗: not established under these criteria.

Benchmark	Mode	#QA	#Tasks	#Tools	#Avg. Len.	RT	MTC	FLD
V* (Wu & Xie, 2024)	Static	191	2	0	–	✗	✗	✗
HRBench (Wang et al., 2025b)	Static	200	6	0	–	✗	✗	✗
GTA (Wang et al., 2024)	Interface	229	3	14	2.43	✓	✓	✗
Agent-X (Ashraf et al., 2025)	Interface	828	5	14	3.4	✓	✓	✗
VisualToolBench (Guo et al., 2025a)	Interface	1.2K	5	6	$< 5^{\ddagger}$	✓	✓	✗
AgentVista (Su et al., 2026)	Interface	209	7	4	$-^{\star}$	✓̸	✓	✗
TIR-Bench (Li et al., 2025a)	Code	1.2K	13	$1^{\dagger}$	–	✗	✓̸	✓̸
Ours	Code & Interface	680	9	35	**5.04**	✓	✓	✓

#Avg. Len.: average reference trajectory length. –: not reported; †: code interpreter. ‡ Typical length, as reported by VisualToolBench. * Actual execution lengths on VTC-Bench are comparable in scale to AgentVista (App. E.7).

methods (Zheng et al., 2025; Hong et al., 2025; Su et al., 2025a;b; Lai et al., 2025; Wang et al., 2025a; Zhou et al., 2025) optimize these strategies for specific toolset orchestration. However, the reliance on fixed collections of visual parsers fundamentally restricts generalizability. This rigid design confines models to predefined visual scenarios, preventing adaptation to unseen structures. Programmatic visual manipulation addresses these limitations by utilizing python code as a primitive tool (Gupta & Kembhavi, 2023; Surís et al., 2023). This approach enables on-demand tool construction with complex logic, including loops and conditionals. Advanced frameworks (Hu et al., 2024; Fu et al., 2025; Vinker et al., 2025; Qiao et al., 2025; Zhao et al., 2025) dynamically generate code for targeted visual editing, while Thyme (Zhang et al., 2025b) provides code testing for open-source tool-use models. Leading models, including GPT-o3 (OpenAI, 2025b), GPT-o4-mini (OpenAI, 2025b), and GPT-5.2 (OpenAI, 2025a), leverage code execution to construct task-specific tools dynamically. Consequently, as state-of-the-art models increasingly embrace this agentic tool-calling paradigm, there is a critical need for a benchmark equipped with a sufficiently diverse tool library to rigorously evaluate their complex, multi-tool compositional capabilities.

Agentic Benchmark in MLLM Benchmark Standard evaluations of multimodal large language models primarily focus on static perception and reasoning. Previous works (Lu et al., 2022; Fu et al., 2023; Lu et al., 2024; Shi et al., 2025a; Liu et al., 2024a; Yue et al., 2024; Shi et al., 2025c; Li et al., 2025b) test models using static questions and treat vision as a passive input. For visual agent evaluation, some studies (Wu & Xie, 2024; Wang et al., 2025b) introduce active visual exploration tasks. However, these early evaluations only examine basic operations like cropping and zooming. Recent research (Wang et al., 2024; Guo et al., 2025b; Li et al., 2025a; Guo et al., 2025a; Su et al., 2026; Ashraf et al., 2025) further advances this field. These methods evaluate multimodal agentic reasoning (Guo et al., 2025b), assess image processing capabilities (Li et al., 2025a), and combine multiple tools for open-ended visual tasks (Guo et al., 2025a). Despite these advancements, existing benchmarks are constrained by limited tool inventories and lack systematic requirements for compositional multi-tool reasoning, and often fail to capture the nuanced demands of practical, real-world applications. In contrast, our proposed benchmark explicitly targets tool diversity and the complexity of multi-step tool composition. We design 680 problems to evaluate visual tool selection and composition using a toolkit of 35 operations. Crucially, VTC-Bench supports a dual-paradigm evaluation protocol: a code-driven mode where agents synthesize Python programs to orchestrate tools, and an interface-driven mode where agents invoke tools through structured APIs. These two paradigms enable comparison of model performance under code-driven and interface-driven tool use. A detailed comparison between VTC-Bench and existing benchmarks is presented in Tab. 1.

3 VisualToolChain-Bench

This section introduces VisualToolChain-Bench(VTC-Bench), with an overview provided in Fig. 2. We first present the benchmark design in Sec. 3.1 by establishing a systematic task taxonomy and

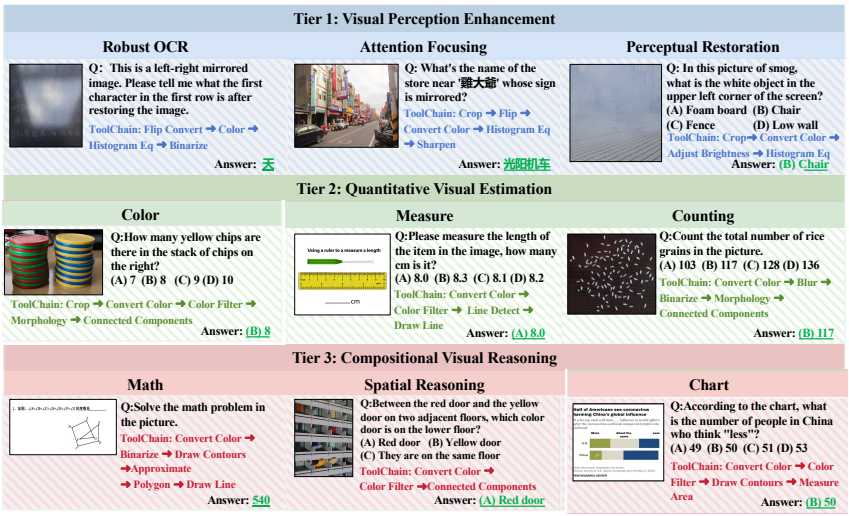


Figure 2: **Overview of VTC-Bench.** Our benchmark groups 9 VQA tasks into three functional tiers covering visual perception enhancement, quantitative visual estimation, and compositional visual reasoning. Each task provides a reference toolchain for fine-grained diagnostic analysis.

corresponding toolset. Building upon this foundation, Sec. 3.2 details the benchmark construction process, encompassing data collection, statistical analysis of the dataset, and evaluation metrics.

3.1 Benchmark Design

Tool Set To address the sparse tool diversity in existing benchmarks, we curate 35 visual operations that demand both tool-set breadth and deterministic reproducibility. We implement these operations based on OpenCV (Its, 2014), aligning our selection with a standard cognitive pipeline: initial restoration, feature distillation, and verification. The tools are organized into four functional modules: (1) Geometry for spatial transformations (e.g., rotation and image pyramids); (2) Enhancement for signal optimization (e.g., color space conversion and binarization); (3) Feature Extraction for deriving structural and semantic primitives (e.g., edge detection and watershed segmentation); and (4) Drawing for reasoning verification and attribute quantification (e.g., contour visualization and area measurement). A comprehensive taxonomy and technical definitions are detailed in App. C.1. Our toolkit exposes individual visual processing functions with input–output dependencies, such as connected-component analysis requiring binary masks generated by segmentation functions. Our audit identifies such prerequisite dependencies in 674 of 680 reference trajectories (99.12%).

Task Design Our evaluation suite organizes 9 tasks into three functional tiers according to the cognitive function being evaluated. Tier labels distinguish visual perception enhancement, quantitative visual estimation, and compositional visual reasoning; they do not imply monotonically increasing task difficulty or reference-trajectory length:

Tier 1: Visual Perception Enhancement. This tier comprises *Robust OCR, Perceptual Restoration, and Attention Focusing*. These tasks evaluate how models use visual tools to mitigate environmental interference (e.g., haze) and rectify geometric distortions (e.g., rotation). Specifically, *Robust OCR* targets text recognition under synthetic degradation that remains human-readable; *Perceptual Restoration* focuses on scene recovery in adverse conditions such as haze or low light; and *Attention Focusing* emphasizes fine-grained analysis under geometric transformations like rotation or flipping.

Tier 2: Quantitative Visual Estimation. The tasks of *Measurement, Color, and Counting* evaluate the model's capacity to perceive and precisely quantify physical attributes. Specifically, *Measurement* requires extracting size, position, and shape; *Color* examines the precise extraction of chromatic information; and *Counting* focuses on scene analysis and the strategic invocation of specialized counting tools.

Tier 3: Compositional Visual Reasoning. The *Chart, Math, and Spatial Reasoning* tasks evaluate how models combine visual tool use with logical inference. *Chart* is a comprehensive task requiring simultaneous restoration, perception, and inference. *Math* evaluates the construction of auxiliary

Statistic	Number
Total questions	680
Multiple-choice	538 (79.12%)
Open	142 (20.88%)
Questions with image	680 (100.00%)
Average chain length	5.04
Average unique tools	4.97
Total tool calls	3,428
Min chain length	1
Median chain length	5
Max chain length	10
Average prompt length	18.52

Table 2: Statistical summary of VTC-Bench.

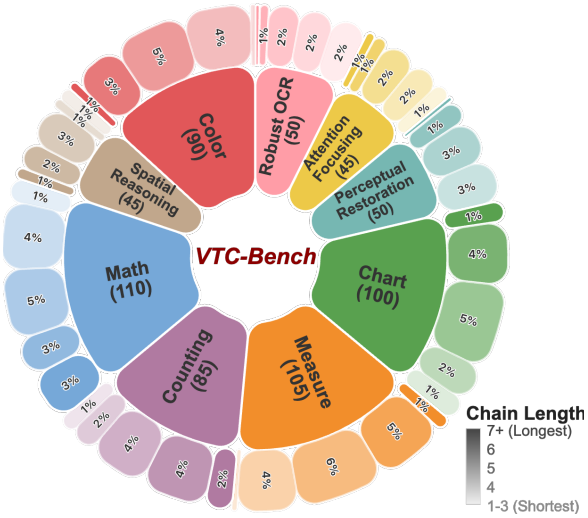


Figure 3: **Domain distribution (inner) & toolchain lengths (outer).** Lengths range from 1-3 to 7+, where darker shades denote longer steps.

geometric elements, while *Spatial Reasoning* tests the robust analysis of spatial relations under extreme conditions, such as overexposure or heavy blur.

This functional taxonomy covers complementary aspects of visual evidence recovery, attribute estimation, and compositional reasoning. A detailed overview of VTC-Bench is illustrated in Fig. 2.

3.2 Benchmark Construction

Data collection Expert researchers manually curate images from over tens of thousands of internet sources and open-source datasets, selecting samples for evaluating compositional visual tool use (Fig. 2). For each selected image, we design tool-centric instructions. Across the benchmark, 98.4% of reference trajectories contain at least three tool calls. We further introduce controlled visual perturbations (geometric distortions and radiometric noise) to evaluate model robustness under non-ideal conditions.

Verification Protocol A team of six graduate-level researchers finalizes each image, question, answer, and any task-compatible perturbation. Gemini-3.0-Pro (Google, 2026) and GPT-5.2 (OpenAI, 2025a) then generate candidate trajectories. Experts inspect the calls and intermediate images, compare the candidates, remove failed or redundant operations, and consolidate an executable reference path. At least two experts review tool applicability, data dependencies, and answer consistency; a separate expert adjudicates disagreements. Each reference is one expert-validated solution, not necessarily unique or shortest. Further construction details and task statistics appear in App. C.2.

Benchmark Statistics As summarized in Tab. 2, VTC-Bench comprises a diverse set of 680 questions with reference trajectories averaging 5.04 steps and 4.97 unique tools. To better visualize this, Fig. 3 illustrates the distribution of each task category, while App. G provides representative image and question examples for further clarity.

3.3 Evaluation Metrics

We adopt Average Pass Rate (APR) as the primary evaluation metric to represent the proportion of correctly answered questions. To analyze tool-use behavior, we define the *Effective Toolchain* as the data-flow-relevant subsequence of a model's own tool-call history. This sequence is determined through a backtracking process from the final output to the original input image.

We evaluate the models using two additional metrics: Tool Call Rate (TCR), which measures the proportion of tasks where a model invokes at least one tool; and Tool Usage Efficiency (Eff_{tool}), which assesses the precision and conciseness of the tool-calling sequences by comparing the number

of effective steps to the total predicted steps. Mathematically, TCR and Eff_{tool} are formulated as:

$$\text{TCR} = \frac{|\{i \mid L_{T,i} > 0\}|}{N}, \quad Eff_{\text{tool}} = \frac{\sum L_{e,i}}{\sum L_{T,i}} \tag{1}$$

where N represents the total number of evaluated samples. $L_{\text{T},i}$ and $L_{\text{e},i}$ denote the length of the total toolchain and the effective toolchain for the i-th sample, respectively. Additional fine-grained metrics including Mean Absolute Error (MAE) are defined and analyzed in App. A.

4 Experiments

4.1 Experiment Setup

Model Selection We evaluate 19 MLLMs across four categories, covering proprietary and open-source models with different tool-use training backgrounds. Proprietary tool-use models include GPT-o3 and GPT-o4-mini (OpenAI, 2025b), while proprietary general-purpose models include GPT-5.2 (OpenAI, 2025a), Gemini-3.0-Pro (Google, 2026), Gemini-3.0-Flash (Google, 2025), Gemini-2.5-Pro (Comanici et al., 2025), and GPT-4o (Hurst et al., 2024). Open-source tool-use models include DeepEyes (Zheng et al., 2025), Thyme (Zhang et al., 2025b), DeepEyesV2 (Hong et al., 2025), and V-Thinker (Qiao et al., 2025). For open-source general-purpose models, we evaluate Qwen3-VL (Bai et al., 2025) Instruct and Thinking variants at 8B, 30B-A3B, 32B, and 235B scales.

Evaluation Framework To measure the benefits of tool invocation, we compare direct answering baselines against tool-augmented reasoning across both interface-driven and code-driven paradigms. This comparison reveals how a model's programming proficiency influences overall orchestration success. We utilize the Qwen-Agent (Team, 2025) framework to manage both code and interface implementations for models with robust native tool-calling support. Conversely, we employ the Thyme (Zhang et al., 2025b) framework for models with limited native tool-calling proficiency, applying this approach primarily to open-source tool-use models to facilitate code generation. To facilitate this process, we provide detailed tool descriptions that explicitly outline usage instructions and parameter specifications, ensuring that even models lacking specialized OpenCV training can comprehend and invoke the available interfaces. Finally, we implement a comprehensive dual-stream evaluation protocol, integrating deterministic rule-based matching with a GPT-4o (Hurst et al., 2024) LLM-as-a-Judge paradigm. Detailed prompt templates are provided in the App. F.

4.2 Main Result

VTC-Bench challenges current MLLMs, with proprietary models showing stronger performance and tool-use gains. As shown in Tab. 3, direct-answering accuracy ranges from 22.06% to 46.47%, with most models scoring around 30%. Gemini-3.0-Flash (Google, 2025) achieves the highest baseline accuracy of 46.47%. Increasing model size alone yields limited improvements: Qwen3-VL-Instruct improves from 30.74% at 8B to 36.32% at 235B, yet even the strongest Qwen3-VL configuration remains below 40% with tool augmentation. Although the benchmark poses difficulties across all evaluated model groups, proprietary models generally achieve higher accuracy and benefit more from external tools. GPT-4o gains 9.56 percentage points through interface-based invocation, while GPT-5.2 gains 8.53 points through code execution. Gemini-3.0-Pro achieves the highest overall accuracy of 51.18% in the code-driven setting, improving by 6.77 points over its direct-answering baseline. In contrast, open-source models exhibit smaller and less consistent gains, with tool augmentation sometimes reducing accuracy. Thus, while proprietary models demonstrate stronger tool-use performance, the best overall accuracy of only 51.18% indicates that substantial challenges remain in solving tasks that involve multi-step visual tool orchestration.

Models perform better at coarse-grained tool invocation, and intrinsic perception is the prerequisite for tool use. In visual perception enhancement tasks, models exhibit strong performance, with GPT-4o boosting *Perceptual Restoration* by +14.00% via interface. These tasks require only basic, *coarse-grained* tool operations, such as rotation or mirroring, to achieve significant improvements. Conversely, quantitative visual estimation yields bifurcated outcomes, with noticeable degradation in high-precision perception tasks. Because Tier 2 rigorously tests precise tool selection and parameter configuration, it exposes models' inability to execute *fine-grained* tool manipulations. Finally, in compositional visual reasoning tasks, models achieve remarkable leaps, highlighted by a +17.78%

Table 3: **Comprehensive evaluation results. Bold and Underlined** indicates the highest value in each column. For Overall and Delta columns: Green indicates improvement and Red indicates degradation compared to base. For subcategories, Yellow indicates an improvement of more than 10% compared to base.

Model	Size	Setting	Overall	Delta	OCR	Attn.	Rest.	Chart	Meas.	Count.	Math	Spat.	Color
Category 1: Proprietary Tool-use Models													
GPT-o3	—	Base	31.91		20.00	26.67	32.00	30.00	38.10	35.29	28.18	35.56	35.56
		Code	36.62	+4.71	28.00	31.11	34.00	40.00	38.10	34.12	30.91	51.11	42.22
		Inter.	36.76	+4.85	28.00	24.44	34.00	39.00	35.24	40.00	34.55	44.44	44.44
GPT-o4-mini	—	Base	31.18		18.00	13.33	30.00	29.00	33.33	44.71	30.91	42.22	30.00
		Code	33.68	+2.50	12.00	11.11	30.00	41.00	31.43	35.29	40.00	46.67	37.78
		Inter.	32.50	+1.32	24.00	13.33	28.00	30.00	33.33	40.00	33.64	46.67	35.56
Category 2: Proprietary General-purpose Models													
GPT-4o	—	Base	24.26		18.00	15.56	22.00	25.00	25.71	25.88	22.73	26.67	30.00
		Code	31.62	+7.36	18.00	22.22	24.00	38.00	37.14	40.00	20.91	51.11	30.00
		Inter.	33.82	**+9.56**	20.00	22.22	36.00	31.00	41.90	40.00	24.55	48.89	37.78
Gemini-2.5-Pro	—	Base	38.09		48.00	26.67	28.00	44.00	40.95	49.41	30.00	40.00	32.22
		Code	36.03	-2.06	46.00	40.00	34.00	37.00	35.24	37.65	30.00	44.44	31.11
		Inter.	39.85	+1.76	60.00	44.44	40.00	38.00	44.76	41.18	26.36	35.56	40.00
GPT-5.2	—	Base	36.03		24.00	24.44	30.00	43.00	35.24	43.53	33.64	46.67	35.56
		Code	44.56	+8.53	36.00	31.11	26.00	56.00	50.48	44.71	**45.45**	51.11	42.22
		Inter.	40.74	+4.71	28.00	37.78	34.00	47.00	39.05	37.65	41.82	51.11	44.44
Gemini-3.0-Flash	—	Base	46.47		44.00	51.11	46.00	57.00	50.48	52.94	27.27	60.00	40.00
		Code	50.59	+4.12	70.00	57.78	48.00	59.00	50.48	52.94	35.45	51.11	44.44
		Inter.	50.74	+4.27	68.00	53.33	**50.00**	**61.00**	44.76	**58.82**	39.09	55.56	40.00
Gemini-3.0-Pro	—	Base	44.41		54.00	51.11	42.00	51.00	45.71	42.35	34.55	53.33	37.78
		Code	**51.18**	+6.77	**74.00**	62.22	38.00	60.00	**54.29**	36.47	41.82	55.56	**50.00**
		Inter.	51.03	+6.62	70.00	**73.33**	38.00	59.00	50.48	35.29	37.27	**71.11**	**50.00**
Category 3: Open-source Tool-use Models													
V-Thinker	7B	Base	22.06		8.00	11.11	22.00	27.00	23.81	27.06	18.18	28.89	24.44
		Code	23.82	+1.76	10.00	8.89	24.00	21.00	23.81	43.53	17.27	42.22	22.22
		Inter.	24.41	+2.35	16.00	8.89	34.00	23.00	21.90	34.12	17.27	44.44	25.56
DeepEyes-v2	7B	Base	24.85		22.00	11.11	18.00	27.00	38.10	21.18	20.00	28.89	26.67
		Code	30.00	+5.15	20.00	8.89	20.00	30.00	44.76	32.94	28.18	44.44	26.67
		Inter.	23.38	-1.47	18.00	6.67	28.00	16.00	28.57	30.59	18.18	35.56	27.78
DeepEyes	7B	Base	26.47		12.00	8.89	28.00	34.00	33.33	27.06	17.27	44.44	27.78
		Code	27.06	+0.59	22.00	8.89	28.00	30.00	32.38	36.47	17.27	40.00	25.56
		Inter.	29.26	+2.79	22.00	6.67	28.00	29.00	43.81	37.65	11.82	46.67	33.33
Thyme	7B	Base	27.35		8.00	8.89	34.00	29.00	41.90	35.29	18.18	35.56	24.44
		Code	28.82	+1.47	18.00	2.22	24.00	35.00	40.00	38.82	20.91	33.33	28.89
		Inter.	27.06	-0.29	14.00	4.44	24.00	33.00	33.33	34.12	20.00	40.00	28.89
Category 4: Open-source General-purpose Models													
Qwen3-VL-Instruct	8B	Base	30.74		32.00	11.11	28.00	27.00	31.43	38.82	31.82	35.56	33.33
		Code	28.24	-2.50	26.00	17.78	20.00	27.00	39.05	29.41	23.64	17.78	37.78
		Inter.	28.68	-2.06	30.00	17.78	34.00	20.00	30.48	42.35	24.55	26.67	31.11
Qwen3-VL-Thinking	8B	Base	30.29		20.00	11.11	24.00	33.00	40.00	35.29	29.09	35.56	28.89
		Code	26.32	-3.97	12.00	6.67	22.00	29.00	32.38	36.47	22.73	20.00	34.44
		Inter.	29.41	-0.88	20.00	15.56	22.00	34.00	32.38	37.65	24.55	28.89	35.56
Qwen3-VL-Instruct	32B	Base	32.79		28.00	8.89	26.00	37.00	31.43	48.24	27.27	44.44	34.44
		Code	30.59	-2.20	28.00	15.56	20.00	32.00	31.43	34.12	33.64	31.11	35.56
		Inter.	33.09	+0.30	34.00	13.33	16.00	21.00	37.14	51.76	29.09	51.11	38.89
Qwen3-VL-Thinking	32B	Base	32.94		20.00	4.44	28.00	32.00	40.95	43.53	34.55	42.22	32.22
		Code	33.09	+0.15	22.00	8.89	26.00	41.00	35.24	43.53	31.82	33.33	35.56
		Inter.	33.97	+1.03	30.00	17.78	28.00	32.00	35.24	44.71	31.82	40.00	37.78
Qwen3-VL-Instruct	30B-A3B	Base	33.09		42.00	15.56	22.00	23.00	41.90	41.18	32.73	42.22	32.22
		Code	31.03	-2.06	42.00	22.22	32.00	31.00	31.43	36.47	23.64	37.78	28.89
		Inter.	31.91	-1.18	40.00	24.44	28.00	25.00	37.14	40.00	28.18	44.44	25.56
Qwen3-VL-Thinking	30B-A3B	Base	29.56		16.00	8.89	22.00	31.00	30.48	44.71	24.55	46.67	32.22
		Code	28.24	-1.32	18.00	13.33	22.00	35.00	36.19	23.53	27.27	40.00	27.78
		Inter.	33.24	+3.68	28.00	15.56	24.00	29.00	39.05	43.53	30.00	48.89	34.44
Qwen3-VL-Instruct	235B	Base	36.32		42.00	11.11	28.00	33.00	40.95	48.24	35.45	35.56	38.89
		Code	33.68	-2.64	36.00	15.56	28.00	26.00	40.00	42.35	30.00	46.67	35.56
		Inter.	34.85	-1.47	40.00	22.22	20.00	32.00	38.10	48.24	27.27	37.78	41.11
Qwen3-VL-Thinking	235B	Base	35.15		34.00	8.89	24.00	38.00	39.05	45.88	32.73	37.78	38.89
		Code	34.41	-0.74	36.00	22.22	26.00	37.00	36.19	35.29	34.55	37.78	36.67
		Inter.	38.09	+2.94	40.00	24.44	38.00	35.00	32.38	52.94	34.55	48.89	38.89

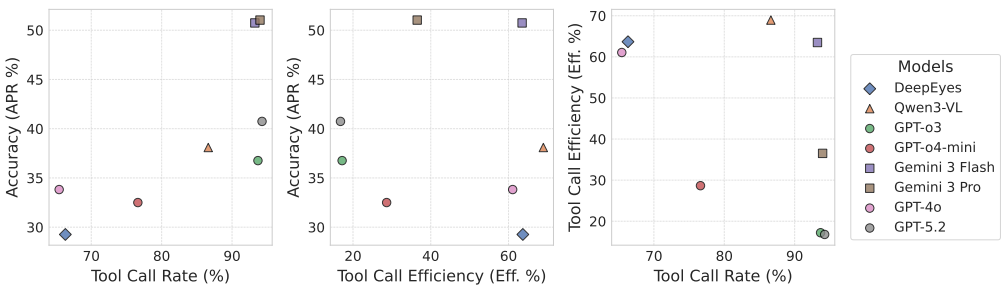


Figure 4: **Relationships among Tool Call Rate, Tool Call Efficiency, and APR.** The panels illustrate APR vs. Tool Call Rate (left), APR vs. Tool Call Efficiency (middle), and Tool Call Rate vs. Tool Call Efficiency (right).

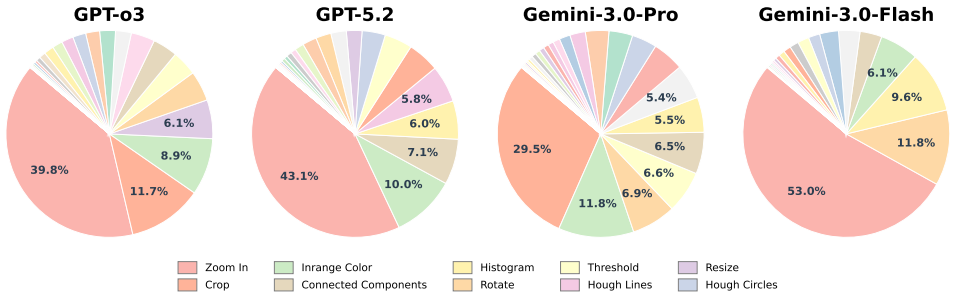


Figure 5: **Distribution of tool utilization across different models.** The usage patterns remain largely consistent across models, with the most frequently invoked tools including *Zoom In, Inrange Color, Rotate, Histogram, and Connected Components*.

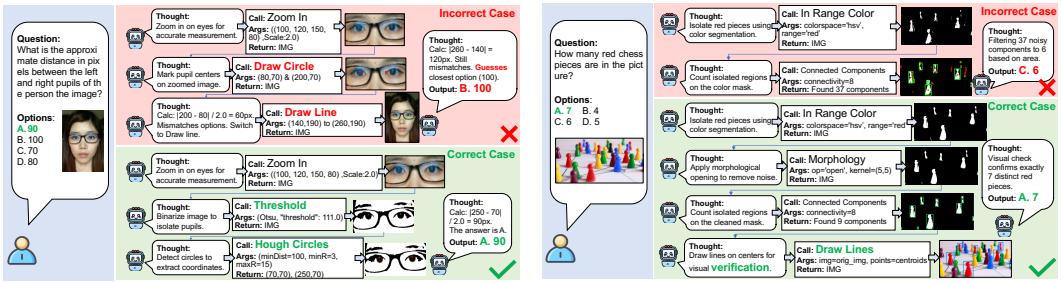
improvement in Gemini-3.0-Pro's *Spatial Reasoning* score. This resurgence stems from the comprehensive nature of Tier 3, which simultaneously challenges intrinsic perception and tool proficiency. Notably, models with stronger foundational perception excel here, underscoring that robust intrinsic perception is an indispensable prerequisite for advanced tool calling.

Dual-paradigm evaluation reveals divergent capability profiles. Contrary to the common assumption that code execution is inherently superior, interface-based invocation proves comparable overall, with three of five proprietary general-purpose models performing better via interfaces. Further inspection of the Delta columns reveals that specialized tool-use models (Category 1) consistently benefit from code-driven execution due to their code-centric training, while general-purpose models often achieve larger gains through interfaces, which abstract away syntactic complexity and allow models to focus on high-level planning. The two paradigms offer complementary views of tool use: code-driven execution evaluates programmatic orchestration, while interface-driven invocation evaluates tool selection and composition through structured API calls.

4.3 Analysis Experiments

Analysis of Tool Utilization and Efficiency. As shown in Fig. 4, TCR is positively associated with APR: models invoking tools on a larger fraction of tasks generally achieve higher answer accuracy. Efficiency nevertheless varies substantially among models with similar TCR, distinguishing the prevalence of tool use from the fraction of calls retained in the effective toolchain. However, the TCR–APR association may also reflect differences in underlying model capability. To examine gains within each model, we pair Direct and Interface outputs by question ID and measure the fraction of Direct failures corrected after at least one actual tool call. Across eight models, recovery rates range from 14.40% to 33.07%, showing that tool-using runs recover a subset of failures from direct answering. Full results and efficiency statistics appear in App. E.1 and App. Tab. 4, respectively.

Tool-use concentration. We design the tasks to cover diverse visual operations and their combinations. However, Fig. 5 shows that representative Gemini and GPT models concentrate their calls on a small subset of the available tools. Even when supplied with a reference tool sequence, models use only



(a) Tool selection and execution errors.

(b) Over-reliance on unverified tool outputs.

Figure 6: **Qualitative analysis of failure modes in tool utilization.** (a) The agent mistakenly invokes two tools based on unfounded intuition, leading to a speculative answer. (b) The agent blindly adopts preliminary tool outputs without visual validation, bypassing essential verification steps.

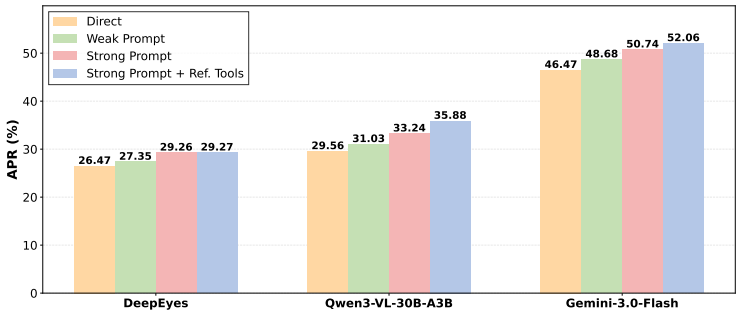


Figure 7: **Impact of system prompts and prompt information density on model APR.** We evaluate the performance of three distinct models, under four prompting configurations: Direct, Weak Prompt, Strong Prompt, and Strong Prompt + Ref. Tools.

20.74%–47.81% of its tool types on average (App. E.5). These results highlight a gap between the tool diversity incorporated into task design and the range of tools models use during execution.

Tool composition and execution depth. Figure 8a relates effective-chain length to APR, while Figs. 8b and 10 compare effective and reference trajectory lengths. These descriptive patterns do not make a reference path optimal or a shorter path invalid. Length, final-answer correctness, and process validity must be interpreted jointly; App. E.2 and App. E.4 provide process and failure audits, complementing the MAE and efficiency statistics in App. A.

Case Study The examples in Fig. 6 illustrate two failure modes: selecting inappropriate tools or parameters, and accepting intermediate outputs without verification. The broader audits of 240 processes and 144 incorrect trajectories are reported in App. E.2–E.4.

Prompt Ablation As shown in Fig. 7, we evaluate Gemini-3.0-Flash (Google, 2025), Qwen3-VL-30B-A3B (Bai et al., 2025), and DeepEyes (Zheng et al., 2025) with direct answering, weak prompting, strong prompting, and strong prompting with reference tools. The last condition supplies ordered operation names while retaining all 35 tools. Models use the guidance only partially: just 1.48%–18.69% of trajectories use all supplied tool types (App. E.5). Limited gains therefore cannot isolate execution difficulty from incomplete adoption of the guidance.

5 Conclusion

We introduced **VTC-Bench**, a benchmark for evaluating compositional visual tool planning in MLLMs. Through 35 visual operations, 680 curated problems, and a dual-paradigm evaluation protocol, our benchmark enables systematic diagnosis of both code-driven and interface-driven tool-use capabilities. Experiments on 19 MLLMs reveal difficulties in visual tool selection and multi-step composition. Our dual-paradigm analysis further uncovers divergent capability profiles across model categories. These findings motivate better planning over interdependent visual operations. Extending to broader tool ecosystems with external APIs remains a future direction.

AI USE STATEMENT

We use generative AI models to support the construction and evaluation of VTC-Bench. Specifically, Gemini-3.0-Pro and GPT-5.2 generate candidate tool-use trajectories, which human experts review, refine, and validate to establish executable reference trajectories. Human annotators construct the benchmark questions and answers. We also use GPT-4o as an automated judge alongside deterministic rule-based matching, as described in the paper. An AI assistant helps draft and edit the manuscript, prepare figures from recorded experimental results and model outputs, and check consistency between the implementation and the text. The authors retain responsibility for all research decisions, the design of the benchmark and experiments, the verification and interpretation of results, and the conclusions and final content of the manuscript.

REPRODUCIBILITY STATEMENT

Sections 3.3 and 4.1 describe the evaluation metrics, model selection, and evaluation framework. Appendix B specifies the inference settings, generation hyperparameters, and interaction limit. Appendices C.1 and C.2 document the tool set, data sources, and annotation and validation procedures. Appendix F provides the system and user prompts, along with instructions for the code interpreter. Appendix E details the protocols and results for the additional validation and process analyses.

REFERENCES

Tajamul Ashraf, Amal Saqib, Hanan Ghani, Muhra AlMahri, Yuhao Li, Noor Ahsan, Umair Nawaz, Jean Lahoud, Hisham Cholakkal, Mubarak Shah, et al. Agent-x: Evaluating deep multimodal reasoning in vision-centric agentic tasks. *arXiv preprint arXiv:2505.24876*, 2025.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

ByteDance Seed Team. Seed2.0 model card: Towards intelligence frontier for real-world complexity, February 2026. URL https://github.com/ByteDance-Seed/Seed2.0. Model Card.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, et al. Mme: A comprehensive evaluation benchmark for multimodal large language models. *arXiv preprint arXiv:2306.13394*, 2023.

Xingyu Fu, Minqian Liu, Zhengyuan Yang, John Corring, Yijuan Lu, Jianwei Yang, Dan Roth, Dinei A. F. Florêncio, and Cha Zhang. ReFocus: Visual editing as a chain of thought for structured image understanding. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 17783–17805. PMLR, 2025. URL https://proceedings.mlr.press/v267/fu25d.html.

GLM-5-Team, :, Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, Chenzheng Zhu, Congfeng Yin, Cunxiang Wang, Gengzheng Pan, Hao Zeng, Haoke Zhang, Haoran Wang, Huilong Chen, Jiajie Zhang, Jian Jiao, Jiaqi Guo, Jingsen Wang, Jingzhao Du, Jinzhu Wu, Kedong Wang, Lei Li, Lin Fan, Lucen Zhong, Mingdao Liu, Mingming Zhao, Pengfan Du, Qian Dong, Rui Lu, Shuang-Li, Shulin Cao, Song Liu, Ting Jiang, Xiaodong Chen, Xiaohan Zhang, Xuancheng Huang, Xuezhen Dong, Yabo Xu, Yao Wei, Yifan An, Yilin Niu, Yitong Zhu, Yuanhao Wen, Yukuo Cen, Yushi Bai, Zhongpei Qiao, Zihan Wang, Zikang Wang, Zilin Zhu, Ziqiang Liu, Zixuan Li, Bojie Wang, Bosi Wen, Can Huang, Changpeng Cai, Chao Yu, Chen Li, Chengwei Hu, Chenhui Zhang, Dan Zhang, Daoyan Lin, Dayong Yang, Di Wang, Ding Ai, Erle Zhu, Fangzhou Yi, Feiyu Chen, Guohong Wen, Hailong Sun, Haisha Zhao, Haiyi Hu, Hanchen Zhang, Hanrui Liu, Hanyu Zhang, Hao Peng, Hao Tai, Haobo Zhang, He Liu, Hongwei Wang, Hongxi Yan, Hongyu Ge, Huan Liu, Huanpeng Chu,

Jia'ni Zhao, Jiachen Wang, Jiajing Zhao, Jiamin Ren, Jiapeng Wang, Jiaxin Zhang, Jiayi Gui, Jiayue Zhao, Jijie Li, Jing An, Jing Li, Jingwei Yuan, Jinhua Du, Jinxin Liu, Junkai Zhi, Junwen Duan, Kaiyue Zhou, Kangjian Wei, Ke Wang, Keyun Luo, Laiqiang Zhang, Leigang Sha, Liang Xu, Lindong Wu, Lintao Ding, Lu Chen, Minghao Li, Nianyi Lin, Pan Ta, Qiang Zou, Rongjun Song, Ruiqi Yang, Shangqing Tu, Shangtong Yang, Shaoxiang Wu, Shengyan Zhang, Shijie Li, Shuang Li, Shuyi Fan, Wei Qin, Wei Tian, Weining Zhang, Wenbo Yu, Wenjie Liang, Xiang Kuang, Xiangmeng Cheng, Xiangyang Li, Xiaoquan Yan, Xiaowei Hu, Xiaoying Ling, Xing Fan, Xingye Xia, Xinyuan Zhang, Xinze Zhang, Xirui Pan, Xu Zou, Xunkai Zhang, Yadi Liu, Yandong Wu, Yanfu Li, Yidong Wang, Yifan Zhu, Yijun Tan, Yilin Zhou, Yiming Pan, Ying Zhang, Yinpei Su, Yipeng Geng, Yong Yan, Yonglin Tan, Yuean Bi, Yuhan Shen, Yuhao Yang, Yujiang Li, Yunan Liu, Yunqing Wang, Yuntao Li, Yurong Wu, Yutao Zhang, Yuxi Duan, Yuxuan Zhang, Zezhen Liu, Zhengtao Jiang, Zhenhe Yan, Zheyu Zhang, Zhixiang Wei, Zhuo Chen, Zhuoer Feng, Zijun Yao, Ziwei Chai, Ziyuan Wang, Zuzhou Zhang, Bin Xu, Minlie Huang, Hongning Wang, Juanzi Li, Yuxiao Dong, and Jie Tang. Glm-5: from vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763.

Google. Gemini 3 Flash: Frontier Intelligence Built for Speed. https://blog.google/technology/ai/google-gemini-3-flash/, December 2025. Published December 10, 2025. Accessed: 2026-01-29.

Google. Gemini: The Most Capable and General AI Model. https://gemini.google.com/, 2026. Accessed: 2026-01-29.

Xingang Guo, Utkarsh Tyagi, Advait Gosai, Paula Vergara, Jayeon Park, Ernesto Gabriel Hernández Montoya, Chen Bo Calvin Zhang, Bin Hu, Yunzhong He, Bing Liu, et al. Beyond seeing: Evaluating multimodal llms on tool-enabled image perception, transformation, and reasoning. *arXiv preprint arXiv:2510.12712*, 2025a.

Yifu Guo, Zishan Xu, Zhiyuan Yao, Yuquan Lu, Jiaye Lin, Sen Hu, Zhenheng Tang, Huacan Wang, and Ronghao Chen. Octopus: Agentic multimodal reasoning with six-capability orchestration. *arXiv preprint arXiv:2511.15351*, 2025b.

Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14953–14962, 2023.

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

Yushi Hu, Weijia Shi, Xingyu Fu, Dan Roth, Mari Ostendorf, Luke Zettlemoyer, Noah A Smith, and Ranjay Krishna. Visual sketchpad: Sketching as a visual chain of thought for multimodal language models. *Advances in Neural Information Processing Systems*, 37:139348–139379, 2024.

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

The OpenCV Reference Manual. Itseez, 2.4.9.0 edition, April 2014.

Xin Lai, Junyi Li, Wei Li, Tao Liu, Tianjian Li, and Hengshuang Zhao. Mini-o3: Scaling up reasoning patterns and interaction turns for visual search. *arXiv preprint arXiv:2509.07969*, 2025.

Ming Li, Jike Zhong, Shitian Zhao, Haoquan Zhang, Shaoheng Lin, Yuxiang Lai, Chen Wei, Konstantinos Psounis, and Kaipeng Zhang. Tir-bench: A comprehensive benchmark for agentic thinking-with-images reasoning. *arXiv preprint arXiv:2511.01833*, 2025a.

Yuying Li, Siyi Qian, Hao Liang, Leqi Zheng, Ruichuan An, Yongzhen Guo, and Wentao Zhang. Capgeo: A caption-assisted approach to geometric reasoning. *arXiv preprint arXiv:2510.09302*, 2025b.

Yijun Liang, Ming Li, Chenrui Fan, Ziyue Li, Dang Nguyen, Kwesi Cobbina, Shweta Bhardwaj, Jiuhai Chen, Fuxiao Liu, and Tianyi Zhou. ColorBench: Can VLMs see and understand the colorful world? a comprehensive benchmark for color perception, reasoning, and robustness. In *Advances*

in Neural Information Processing Systems, volume 38, 2025. doi: 10.52202/085713-4803. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/d34497330b1fd6530f7afd86d0df9f76-Abstract-Datasets_and_Benchmarks_Track.html.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. Mmbench: Is your multi-modal model an all-around player? In *European conference on computer vision*, pp. 216–233. Springer, 2024a.

Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xu-Cheng Yin, Cheng-Lin Liu, Lianwen Jin, and Xiang Bai. Ocrbench: on the hidden mystery of ocr in large multimodal models. *Science China Information Sciences*, 67(12):220102, 2024b.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in neural information processing systems*, 35:2507–2521, 2022.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. MathVista: Evaluating mathematical reasoning of foundation models in visual contexts. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://dblp.org/rec/conf/iclr/LuBX0LH0CG024.

Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the association for computational linguistics: ACL 2022*, pp. 2263–2279, 2022.

OpenAI. Introducing GPT-5.2. https://openai.com/index/introducing-gpt-5-2/, December 2025a. Published December 11, 2025. Accessed: 2026-01-29.

OpenAI. Introducing OpenAI o3 and o4-mini. https://openai.com/index/introducing-o3-and-o4-mini/, April 2025b. Published April 16, 2025. Accessed: 2026-01-29.

Runqi Qiao, Qiuna Tan, Minghan Yang, Guanting Dong, Peiqing Yang, Shiqiang Lang, Enhui Wan, Xiaowan Wang, Yida Xu, Lan Yang, et al. V-thinker: Interactive thinking with images. *arXiv preprint arXiv:2511.04460*, 2025.

Haozhan Shen, Kangjia Zhao, Tiancheng Zhao, Ruochen Xu, Zilun Zhang, Mingwei Zhu, and Jianwei Yin. Zoomeye: Enhancing multimodal llms with human-like zooming capabilities through tree-based image exploration. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 6602–6618, 2025.

Yang Shi, Yuhao Dong, Yue Ding, Yuran Wang, Xuanyu Zhu, Sheng Zhou, Wenting Liu, Haochen Tian, Rundong Wang, Huanqian Wang, Zuyan Liu, Bohan Zeng, Ruizhe Chen, Qixun Wang, Zhuoran Zhang, Xinlong Chen, Chengzhuo Tong, Bozhou Li, Chaoyou Fu, Qiang Liu, Haotian Wang, Wenjing Yang, Yuanxing Zhang, Pengfei Wan, Yi-Fan Zhang, and Ziwei Liu. Realunify: Do unified models truly benefit from unification? a comprehensive benchmark, 2025a. URL https://arxiv.org/abs/2509.24897.

Yang Shi, Jiaheng Liu, Yushuo Guan, Zhenhua Wu, Yuanxing Zhang, Zihao Wang, Weihong Lin, Jingyun Hua, Zekun Wang, Xinlong Chen, et al. Mavors: Multi-granularity video representation for multimodal large language model. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 10994–11003, 2025b.

Yang Shi, Huanqian Wang, Wulin Xie, Huanyao Zhang, Lijie Zhao, Yi-Fan Zhang, Xinfeng Li, Chaoyou Fu, Zhuoer Wen, Wenting Liu, Zhuoran Zhang, Xinlong Chen, Bohan Zeng, Sihan Yang, Yushuo Guan, Zhang Zhang, Liang Wang, Haoxuan Li, Zhouchen Lin, Yuanxing Zhang, Pengfei Wan, Haotian Wang, and Wenjing Yang. Mme-videoocr: Evaluating ocr-based capabilities of multimodal llms in video scenarios, 2025c. URL https://arxiv.org/abs/2505.21333.

Alex Su, Haozhe Wang, Weiming Ren, Fangzhen Lin, and Wenhu Chen. Pixel Reasoner: Incentivizing pixel space reasoning via curiosity-driven reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 38, 2025a. doi: 10.52202/085713-0280. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/0c38f54740062529aa4117a04b583f3c-Abstract-Conference.html.

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

Zhaochen Su, Jincheng Gao, Hangyu Guo, Zhenhua Liu, Lueyang Zhang, Xinyu Geng, Shijue Huang, Peng Xia, Guanyu Jiang, Cheng Wang, et al. Agentvista: Evaluating multimodal agents in ultra-challenging realistic visual scenarios. *arXiv preprint arXiv:2602.23166*, 2026.

Dídac Surís, Sachit Menon, and Carl Vondrick. Vipergpt: Visual inference via python execution for reasoning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 11888–11898, 2023.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, S. H. Cai, Yuan Cao, Y. Charles, H. S. Che, Cheng Chen, Guanduo Chen, Huarong Chen, Jia Chen, Jiahao Chen, Jianlong Chen, Jun Chen, Kefan Chen, Liang Chen, Ruijue Chen, Xinhao Chen, Yanru Chen, Yanxu Chen, Yicun Chen, Yimin Chen, Yingjiang Chen, Yuankun Chen, Yujie Chen, Yutian Chen, Zhirong Chen, Ziwei Chen, Dazhi Cheng, Minghan Chu, Jialei Cui, Jiaqi Deng, Muxi Diao, Hao Ding, Mengfan Dong, Mengnan Dong, Yuxin Dong, Yuhao Dong, Angang Du, Chenzhuang Du, Dikang Du, Lingxiao Du, Yulun Du, Yu Fan, Shengjun Fang, Qiulin Feng, Yichen Feng, Garimugai Fu, Kelin Fu, Hongcheng Gao, Tong Gao, Yuyao Ge, Shangyi Geng, Chengyang Gong, Xiaochen Gong, Zhuoma Gongque, Qizheng Gu, Xinran Gu, Yicheng Gu, Longyu Guan, Yuanying Guo, Xiaoru Hao, Weiran He, Wenyang He, Yunjia He, Chao Hong, Hao Hu, Jiaxi Hu, Yangyang Hu, Zhenxing Hu, Ke Huang, Ruiyuan Huang, Weixiao Huang, Zhiqi Huang, Tao Jiang, Zhejun Jiang, Xinyi Jin, Yu Jing, Guokun Lai, Aidi Li, C. Li, Cheng Li, Fang Li, Guanghe Li, Guanyu Li, Haitao Li, Haoyang Li, Jia Li, Jingwei Li, Junxiong Li, Lincan Li, Mo Li, Weihong Li, Wentao Li, Xinhang Li, Xinhao Li, Yang Li, Yanhao Li, Yiwei Li, Yuxiao Li, Zhaowei Li, Zheming Li, Weilong Liao, Jiawei Lin, Xiaohan Lin, Zhishan Lin, Zichao Lin, Cheng Liu, Chenyu Liu, Hongzhang Liu, Liang Liu, Shaowei Liu, Shudong Liu, Shuran Liu, Tianwei Liu, Tianyu Liu, Weizhou Liu, Xiangyan Liu, Yangyang Liu, Yanming Liu, Yibo Liu, Yuanxin Liu, Yue Liu, Zhengying Liu, Zhongnuo Liu, Enzhe Lu, Haoyu Lu, Zhiyuan Lu, Junyu Luo, Tongxu Luo, Yashuo Luo, Long Ma, Yingwei Ma, Shaoguang Mao, Yuan Mei, Xin Men, Fanqing Meng, Zhiyong Meng, Yibo Miao, Minqing Ni, Kun Ouyang, Siyuan Pan, Bo Pang, Yuchao Qian, Ruoyu Qin, Zeyu Qin, Jiezhong Qiu, Bowen Qu, Zeyu Shang, Youbo Shao, Tianxiao Shen, Zhennan Shen, Juanfeng Shi, Lidong Shi, Shengyuan Shi, Feifan Song, Pengwei Song, Tianhui Song, Xiaoxi Song, Hongjin Su, Jianlin Su, Zhaochen Su, Lin Sui, Jinsong Sun, Junyao Sun, Tongyu Sun, Flood Sung, Yunpeng Tai, Chuning Tang, Heyi Tang, Xiaojuan Tang, Zhengyang Tang, Jiawen Tao, Shiyuan Teng, Chaoran Tian, Pengfei Tian, Ao Wang, Bowen Wang, Chensi Wang, Chuang Wang, Congcong Wang, Dingkun Wang, Dinglu Wang, Dongliang Wang, Feng Wang, Hailong Wang, Haiming Wang, Hengzhi Wang, Huaqing Wang, Hui Wang, Jiahao Wang, Jinhong Wang, Jiuzheng Wang, Kaixin Wang, Linian Wang, Qibin Wang, Shengjie Wang, Shuyi Wang, Si Wang, Wei Wang, Xiaochen Wang, Xinyuan Wang, Yao Wang, Yejie Wang, Yipu Wang, Yiqin Wang, Yucheng Wang, Yuzhi Wang, Zhaoji Wang, Zhaowei Wang, Zhengtao Wang, Zhexu Wang, Zihan Wang, Zizhe Wang, Chu Wei, Ming Wei, Chuan Wen, Zichen Wen, Chengjie Wu, Haoning Wu, Junyan Wu, Rucong Wu, Wenhao Wu, Yuefeng Wu, Yuhao Wu, Yuxin Wu, Zijian Wu, Chenjun Xiao, Jin Xie, Xiaotong Xie, Yuchong Xie, Yifei Xin, Bowei Xing, Boyu Xu, Jianfan Xu, Jing Xu, Jinjing Xu, L. H. Xu, Lin Xu, Suting Xu, Weixin Xu, Xinbo Xu, Xinran Xu, Yangchuan Xu, Yichang Xu, Yuemeng Xu, Zelai Xu, Ziyao Xu, Junjie Yan, Yuzi Yan, Guangyao Yang, Hao Yang, Junwei Yang, Kai Yang, Ningyuan Yang, Ruihan Yang, Xiaofei Yang, Xinlong Yang, Ying Yang, Yi Yang, Yi Yang, Zhen Yang, Zhilin Yang, Zonghan Yang, Haotian Yao, Dan Ye, Wenjie Ye, Zhuorui Ye, Bohong Yin, Chengzhen Yu, Longhui Yu, Tao Yu, Tianxiang Yu, Enming Yuan, Mengjie Yuan, Xiaokun Yuan, Yang Yue, Weihao Zeng, Dunyuan Zha, Haobing Zhan, Dehao Zhang, Hao Zhang, Jin Zhang, Puqi Zhang, Qiao Zhang, Rui Zhang, Xiaobin Zhang, Y. Zhang, Yadong Zhang, Yangkun Zhang, Yichi Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yushun Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Chenguang Zhao, Feifan Zhao, Jinxiang Zhao, Shuai Zhao, Xiangyu Zhao, Yikai Zhao, Zijia Zhao, Huabin Zheng,

Ruihan Zheng, Shaojie Zheng, Tengyang Zheng, Junfeng Zhong, Longguang Zhong, Weiming Zhong, M. Zhou, Runjie Zhou, Xinyu Zhou, Zaida Zhou, Jinguo Zhu, Liya Zhu, Xinhao Zhu, Yuxuan Zhu, Zhen Zhu, Jingze Zhuang, Weiyu Zhuang, Ying Zou, and Xinxing Zu. Kimi k2.5: Visual agentic intelligence, 2026. URL https://arxiv.org/abs/2602.02276.

Qwen Team. Qwen-agent. https://github.com/QwenLM/Qwen-Agent, 2025.

Yael Vinker, Tamar Rott Shaham, Kristine Zheng, Alex Zhao, Judith E. Fan, and Antonio Torralba. Sketchagent: Language-driven sequential sketch generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 23355–23368, 2025.

Jize Wang, Zerun Ma, Yining Li, Songyang Zhang, Cailian Chen, Kai Chen, and Xinyi Le. Gta: a benchmark for general tool agents. *Advances in Neural Information Processing Systems*, 37: 75749–75790, 2024.

Qixun Wang, Yang Shi, Yifei Wang, Yuanxing Zhang, Pengfei Wan, Kun Gai, Xianghua Ying, and Yisen Wang. Monet: Reasoning in latent visual space beyond images and language, 2025a. URL https://arxiv.org/abs/2511.21395.

Wenbin Wang, Liang Ding, Minyan Zeng, Xiabin Zhou, Li Shen, Yong Luo, Wei Yu, and Dacheng Tao. Divide, conquer and combine: A training-free framework for high-resolution image perception in multimodal large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 7907–7915, 2025b.

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

Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. Mm-react: Prompting chatgpt for multimodal reasoning and action. *arXiv preprint arXiv:2303.11381*, 2023.

Xiang Yue, Yuansheng Ni, Tianyu Zheng, Kai Zhang, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. 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*, pp. 9556–9567, 2024.

Andy Zeng, Maria Attarian, Brian Ichter, Krzysztof Marcin Choromanski, Adrian Wong, Stefan Welker, Federico Tombari, Aveek Purohit, Michael S. Ryoo, Vikas Sindhwani, Johnny Lee, Vincent Vanhoucke, and Pete Florence. Socratic Models: Composing zero-shot multimodal reasoning with language. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://dblp.org/rec/conf/iclr/ZengAICWWTPRSLV23.

Xintong Zhang, Zhi Gao, Bofei Zhang, Pengxiang Li, Xiaowen Zhang, Yang Liu, Tao Yuan, Yuwei Wu, Yunde Jia, Song-Chun Zhu, et al. Adaptive chain-of-focus reasoning via dynamic visual search and zooming for efficient vlms. *arXiv preprint arXiv:2505.15436*, 2025a.

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

YiFan Zhang, Yang Shi, Weichen Yu, Qingsong Wen, Xue Wang, Wenjing Yang, Zhang Zhang, Liang Wang, and Rong Jin. Debiasing multimodal large language models via penalization of language priors. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 4232–4241, 2025c.

Shitian Zhao, Haoquan Zhang, Shaoheng Lin, Ming Li, Qilong Wu, Kaipeng Zhang, and Chen Wei. Pyvision: Agentic vision with dynamic tooling. *arXiv preprint arXiv:2507.07998*, 2025.

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

Zetong Zhou, Dongping Chen, Zixian Ma, Zhihan Hu, Mingyang Fu, Sinan Wang, Yao Wan, Zhou Zhao, and Ranjay Krishna. Reinforced visual perception with tools. *arXiv preprint arXiv:2509.01656*, 2025.

A Detailed Tool Invocation Analysis

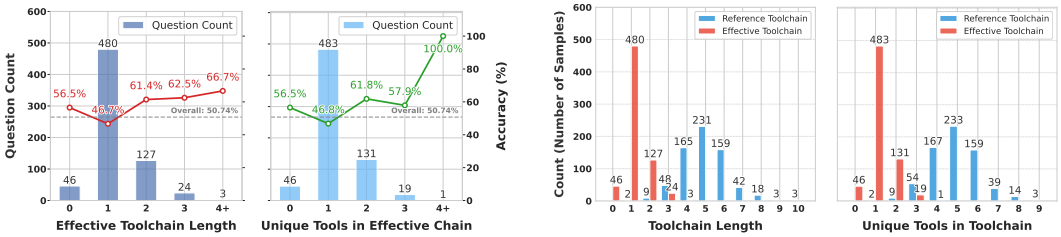
To describe differences in trajectory length, we define the Mean Absolute Error (MAE) as:

$$\text{MAE} = \frac{1}{N} \sum |L_{G,i} - L_{T,i}| \tag{2}$$

where $L_{\text{G},i}$ denotes the reference length and $L_{\text{T},i}$ the model's total toolchain length. MAE is a descriptive length difference, not a measure of correctness or route validity. For example, a correct two-call solution to a question with a five-call reference contributes three to MAE while receiving full APR credit. In Tab. 4, GPT-5.2 has attempted-call MAE 9.57 and efficiency 16.78%; GPT-o3 and Gemini-3.0-Pro have efficiencies of 17.20% and 36.51%. Efficiency measures the retained data-flow-relevant fraction of the model's own calls. These diagnostics should be interpreted with accuracy and the process audits rather than treating reference agreement as optimality.

Table 4: **Statistical analysis of tool invocation performance.** Beyond the Tool-calling Rate (TCR) and Average Pass Rate (APR), we provide a fine-grained evaluation categorized by total attempted tool calls (All Calls) and the effective toolchain (Effective Calls). For both categories, we detail the average number of calls (Avg. T.), the number of unique tools used (Uniq. T.), and the Mean Absolute Error (MAE) from the reference trajectory length. The Tool Usage Efficiency (Eff_{tool}, denoted as Eff.) represents the ratio of effective steps to total attempted steps.

Model	TCR (%)	APR (%)	All Calls			Effective Calls			Eff. (%)
			Avg. T.	Uniq. T.	MAE	Avg. T.	Uniq. T.	MAE	
DeepEyes	66.32	29.26	1.90	1.04	4.18	1.21	0.90	4.16	63.68
Qwen3-VL-235B	86.62	38.09	2.19	1.77	3.17	1.51	1.47	3.56	68.95
GPT-5.2	94.26	40.74	13.23	4.08	9.57	2.22	1.97	2.97	16.78
GPT-o3	93.68	36.76	9.77	3.47	6.65	1.68	1.51	3.42	17.20
GPT-o4-mini	76.62	32.50	3.63	1.77	3.63	1.04	1.02	4.01	28.65
Gemini-3.0-Pro	93.97	51.03	5.04	2.72	3.75	1.84	1.76	3.24	36.51
GPT-4o	65.44	33.82	2.26	1.81	4.00	1.38	1.26	3.97	61.06
Gemini-3.0-Flash	93.24	50.74	1.89	1.38	3.35	1.20	1.19	3.84	63.49



(a) **Sample distribution vs. Accuracy.** (b) **Distributional Discrepancy.**

Figure 8: **Comprehensive analysis for Gemini-3.0-Flash (Interface). (a)** Sample counts and APR grouped by effective-chain length (left) and unique tools in the effective chain (right); overall APR is 50.74%. **(b)** Reference and effective-chain distributions, using the same two measures. Effective calls are the retained data-flow-relevant steps defined in Sec. 3.3.

B Implementation Details

We conduct all evaluations in a zero-shot setting. Open-source models are deployed on NVIDIA H100 GPUs using the `vLLM` framework, while proprietary models are accessed via their official APIs. The detailed generation hyperparameters are set as follows:

- **Open-source Instruct Models:** temperature $= 0.7$, top-p $= 0.8$, top-k $= 20$, repetition penalty $= 1.0$, presence penalty $= 1.5$, max tokens $= 16{,}384$, and seed $= 3407$.

Table 5: Detailed taxonomy and definitions of the 35 OpenCV-based tools utilized in our benchmark. The tools are categorized into four logical groups: Geometry, Enhancement, Feature Extraction, and Drawing. API names have the prefix `opencv_`; the listed suffixes also identify the corresponding operations in the Code prompt.

Tool Name	API suffix	Description
		GEOMETRY
Resize	`resize`	Adjusts the image dimensions to a specified size.
Rotate	`rotate`	Rotates the image by a specified angle.
Translate	`translate`	Shifts the image location along the x or y axis.
Flip	`flip`	Mirrors the image horizontally, vertically, or both.
Crop	`crop`	Extracts a specific rectangular region of interest (ROI).
Zoom In	`zoom_in`	Scales up a specific region of the image for detail inspection.
Pyramid	`pyramid`	Performs pyramid upsampling or downsampling.
		ENHANCEMENT
Gray	`colorspace_gray`	Converts the image to grayscale.
HSV	`colorspace_hsv`	Converts the image to HSV color space.
LAB	`colorspace_lab`	Converts the image to LAB color space.
In-Range Color	`inrange_color`	Filters pixels within a specific color range (segmentation mask).
Blur	`blur`	Smooths image noise using Gaussian, Median, or Bilateral filters.
Denoise	`denoise`	Applies Non-local Means Denoising to smooth while preserving edges.
Threshold	`threshold`	Applies binary, Otsu, or adaptive thresholding techniques.
Morphology	`morphology`	Performs morphological operations (Erosion, Dilation, Open, Close).
Histogram	`histogram`	Enhances contrast using Histogram Equalization or CLAHE.
Adjust Brightness	`convertscaleabs`	Adjusts brightness using absolute scaling (ConvertScaleAbs).
Inpaint	`inpaint`	Restores damaged or missing areas within an image.
		FEATURE EXTRACTION
Canny Edge	`canny`	Identifies edges using the Canny edge detector.
Compute Gradients	`gradients`	Calculates gradients using Sobel or Laplacian operators.
Watershed	`watershed`	Segments touching objects based on topographical distance.
GrabCut	`grabcut`	Extracts foreground objects interactively using graph cuts.
Flood Fill	`floodfill`	Fills a connected component starting from a seed point.
Conn. Components	`connected_components_with_stats`	Computes connected components and their statistics (bounding box, area).
Keypoint Features	`features`	Detects feature points using algorithms like Harris, SIFT, or ORB.
Hough Lines	`hough_lines`	Detects straight lines using the Hough Line Transform.
Hough Circles	`hough_circles`	Detects circular objects using the Hough Circle Transform.
Template Match	`template_match`	Locates the area in the image that matches a template.
DFT	`dft`	Converts the image to the frequency domain (Discrete Fourier Transform).
		DRAWING
Draw Contours	`contours`	Detects and retrieves contours from the binary image.
Approx Poly	`approx_poly`	Approximates a contour shape to a polygon with fewer vertices.
Draw Line	`draw_line`	Annotates the image with straight lines.
Draw Circle	`draw_circle`	Annotates the image with circles.
Contour Area	`contour_area`	Calculates the area of a specific contour.
Arc Length	`arc_length`	Computes the perimeter or curve length of a contour.

- **Open-source Thinking Models:** temperature $= 0.6$, top-p $= 0.95$, top-k $= 20$, repetition penalty $= 1.0$, presence penalty $= 0$, max tokens $= 40{,}960$, and seed $= 1234$.
- **Proprietary Models:** We strictly enable the High Reasoning mode for all assessments and set the max new tokens to $65{,}536$.

In Qwen-Agent, we set the maximum number of rounds to 20.

C Detailed Tool Set And Data Collection

C.1 Tool Set

Table 5 lists all 35 tools and their API names. We count each separately exposed tool API once; parameter options within an API do not count as additional tools.

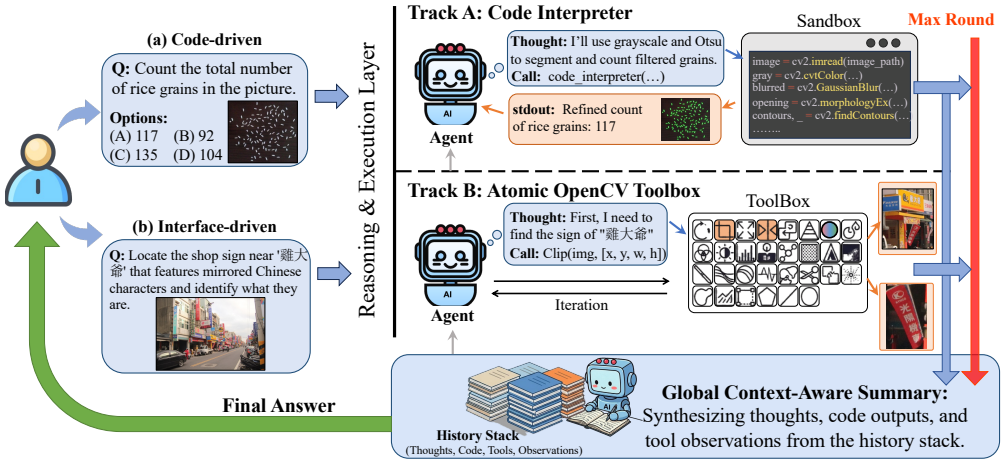


Figure 9: **Overview of workflows for two tool-use protocols of LLM agents during evaluation.** The agent iteratively interacts with tools within the Max Round constraint, ultimately synthesizing tool execution results to generate the final answer.

C.2 Data Collection

We describe the data sources for the nine tasks below. Human annotators construct the questions and answers; reference trajectories are generated by models and then compared, refined, and validated by experts, as described in Sec. 3.2. Before annotation, experts determine whether a perturbation preserves the task semantics and answer validity. Eligible transformations are sampled from a task-compatible pool including rotation, mirroring, blur, noise, and brightness adjustment; unsuitable samples remain unperturbed. Human annotation takes approximately 5–10 minutes per instance. Experts inspect model-produced calls and intermediate images, remove failed or redundant operations, and, where needed, include intermediate operations required for a complete executable reference. Every retained operation must serve a functional role rather than merely increase length. At least two experts cross-review each candidate, with a separate expert adjudicating disagreements.

Attention Focusing: Sourced from HRBench (Wang et al., 2025b) and web-crawled images. We evaluate the agent's robustness against Geometric Perturbations using task-compatible transformations (e.g., rotation, reflection). These transformations support analysis of tool-assisted spatial normalization.

Chart: Derived from ChartQA (Masry et al., 2022) and targeted web searches. We erase original reference labels to evaluate how agents use auxiliary construction and precision sampling to interpret chart elements.

Color: Based on ColorBench (Liang et al., 2025) and internet data. After applying blur and illumination perturbations, we evaluate color-proportion estimation with tools for chromatic-space manipulation (HSV/LAB).

Counting: Adapted from Kaggle and web sources. We target occluded and densely packed objects to evaluate tool-assisted counting strategies such as segmentation and morphological processing.

Math: A specialized set of web-collected and manually synthesized samples focusing on STEM-oriented Geometric Reasoning. These problems evaluate how models use auxiliary geometric constructions, including drawn lines, in polygonal reasoning.

Measurement: Sourced from high-fidelity web images with manual calibration. It bridges the gap between vision and Metrology, featuring scenarios like Industrial Component Calibration that require sub-pixel precision through tool-assisted physical dimension estimation.

Perceptual Restoration: Leveraging Kaggle open-source data, this task evaluates the agent's capacity to act as a "pre-processor," effectively neutralizing atmospheric haze and photon noise to recover latent semantic information from degraded scenes.

Table 6: **Overview of 9 distinct tasks in our benchmark.** Questions and answers are human-annotated. The benchmark contains 538 multiple-choice and 142 open-ended questions overall (Tab. 2); this table reports sources and sample counts.

Task Category	Data Source	QA Construction	Samples
Attention Focusing	HRBench (Wang et al., 2025b) & Web	Human-annotated	45
Chart	ChartQA (Masry et al., 2022) & Web	Human-annotated	100
Color	ColorBench (Liang et al., 2025) & Web	Human-annotated	90
Counting	Kaggle & Web	Human-annotated	85
Math	Web & Manual Construction	Human-annotated	110
Measurement	Web	Human-annotated	105
Perceptual Restoration	Kaggle & Web	Human-annotated	50
Robust OCR	OCRBench (Liu et al., 2024b) & Kaggle	Human-annotated	50
Spatial Reasoning	Kaggle & Web	Human-annotated	45

Robust OCR: An integration of OCRBench (Liu et al., 2024b) and Kaggle competition data. These samples contain compound degradations. We evaluate how models combine image enhancement and text recognition to recover the text.

Spatial Reasoning: Derived from Kaggle and web sources. This task focuses on the quantification of relative topologies, requiring models to transform qualitative visual cues into precise spatial coordinates to resolve complex positional queries.

D More On Analysis Results

D.1 Tool-Use Analysis Results

Figure 10 compares Gemini-3.0-Flash effective-chain lengths with reference lengths by task. Many effective chains contain one or two calls, while references frequently contain four to six; Color and Measurement show particularly concentrated short effective chains. A shorter effective chain alone does not establish premature termination: a model may use a few operations to expose evidence and complete the answer through intrinsic reasoning. Because references are neither unique nor guaranteed shortest, these distributions describe differences between retained-chain and reference lengths. Diagnosing a failure requires checking final-answer correctness and intermediate evidence, as in App. E.2 and App. E.4.

D.2 Evaluation Metrics

We define the Tool Call Rate (TCR) as Eq 1:

$$\text{TCR} = \frac{|\{i \mid L_{T,i} > 0\}|}{N}, \tag{3}$$

where N denotes the total number of samples and $L_{T,i}$ represents the toolchain length for the i-th sample.

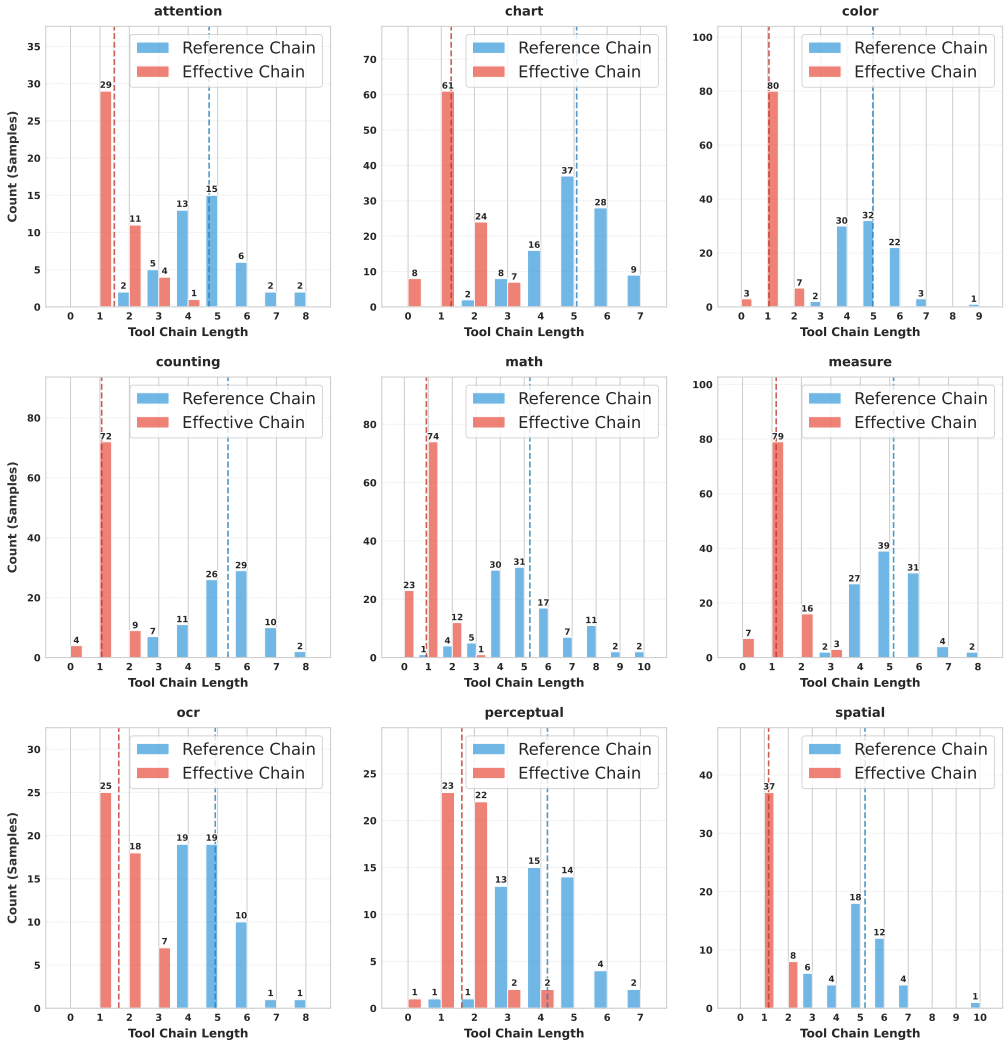


Figure 10: Reference and effective-chain length distributions by task for Gemini-3.0-Flash (Interface). Model lengths count retained data-flow-relevant calls.

E Additional Validation and Process Analyses

E.1 Matched Tool-Assisted Recovery

Cross-model correlations between Tool Call Rate (TCR) and Average Pass Rate (APR) can reflect differences in underlying model capability. To examine changes within each model, we pair its Direct and Interface outputs by question ID. A tool-assisted recovery is a Direct-incorrect answer that becomes correct under Interface after at least one actual tool call. For model m, let $D_{m,i}$ and $I_{m,i}$ indicate final-answer correctness under Direct and Interface, respectively, and let $L_{m,i}$ be the number of actual Interface tool calls. We define

$$R_m = \sum_{i=1}^{N} \mathbf{1}[D_{m,i} = 0,\ I_{m,i} = 1,\ L_{m,i} > 0], \qquad \text{Recovery}_m = \frac{R_m}{\sum_{i=1}^{N} \mathbf{1}[D_{m,i} = 0]}. \quad (4)$$

The denominator contains all Direct failures for that model, including cases not recovered under Interface. Zero-call Interface answers are excluded from the recovery numerator even if they correct a Direct error.

Table 7: **Matched tool-assisted recovery on all 680 questions.** Recovered cases require a correct Interface answer and at least one actual tool call. Recovery rates use each model's Direct-incorrect count as the denominator; TCR is measured across all questions.

Model	TCR (%)	Direct incorrect	Recovered	Recovery (%)
DeepEyes	66.32	500	72	14.40
Qwen3-VL-235B-Thinking	86.62	441	98	22.22
GPT-5.2	94.26	435	127	29.20
GPT-o3	93.68	463	108	23.33
GPT-o4-mini	76.62	468	76	16.24
Gemini-3.0-Pro	93.97	378	125	33.07
GPT-4o	65.44	515	92	17.86
Gemini-3.0-Flash	93.24	364	104	28.57

Recovery rates range from 14.40% to 33.07% across the eight models (Tab. 7). GPT-5.2 recovers 127 of 435 Direct failures, and Gemini-3.0-Pro recovers 125 of 378. Across these models, recovery-rate rankings correlate with TCR rankings (Spearman $\rho = 0.881$, exact two-sided permutation $p = 0.00724$) and Interface APR rankings ($\rho = 0.952$, $p = 0.00114$). These rank statistics are computed over the eight model-level values, with all $8!$ permutations enumerated. They describe associations rather than a causal effect of additional calls.

The matched results identify successful tool-using runs on questions that the same model failed under Direct. They do not imply that every operation was necessary, that redundant calls help, or that a single reference path must be reproduced. Recovery measures one direction of change; it is not the net accuracy gain, since Direct-correct answers may also become incorrect under Interface. The overall Direct-to-Interface APR differences in Tab. 3 summarize net changes across all questions. Without a controlled intervention holding other aspects of generation fixed, neither recovery nor cross-model correlation isolates a causal effect of tool use.

E.2 Final-Answer Correctness and Process Validity

We manually audit 240 trajectories with non-zero tool use to distinguish final-answer correctness from the validity of the observed process. A trajectory is *valid* when its planning, execution and parameters, and interpretation of intermediate outputs form a coherent path with sufficient evidence. It is *partially valid* when it makes useful progress but contains a local error or incomplete evidence, and *invalid* when a key failure leaves the answer unsupported. Different tools, orders, and shorter routes are allowed; exact matching to the reference is not required.

Table 8: **Process-validity audit of 240 non-zero-tool trajectories.** Percentages are calculated within each final-answer group. Process labels are diagnostic and do not change APR credit.

Final answer	Audited	Valid	Partially valid	Invalid
Correct	96	30 (31.25%)	50 (52.08%)	16 (16.67%)
Incorrect	144	0 (0.00%)	36 (25.00%)	108 (75.00%)

Among answer-correct trajectories, 80/96 (83.33%) are valid or partially valid, while 16/96 (16.67%) remain unsupported under the audit criteria. Among answer-incorrect trajectories, 108/144 (75.00%) are invalid, while the remaining 36 show useful but incomplete or locally erroneous progress. Thus, answer correctness alone does not establish process validity, and process inspection provides complementary diagnostic information. Every correct final answer still receives full APR credit. These results describe the audited subset, not an estimate of process-validity prevalence across the entire benchmark.

E.3 Observable Support for Correct Multiple-Choice Answers

We audit the observable support for 3,311 multiple-choice responses that receive correct-answer scores from eight models under Direct and Interface. GLM-5V-Turbo provides initial screening, and GPT-6 Astra independently reviews a stratified sample of 24 responses. We then manually inspect every trajectory to determine the final labels. The audit uses original images, saved textual reasoning, and tool traces; historical intermediate tool images are unavailable.

We label a response *unsupported* when a substantive contradiction, incorrect calculation or observation, or unjustified inference breaks the link between its visible evidence and final answer. Answer-only responses, plan-only responses, and cases with insufficient evidence for assessment receive the label *unassessable*. Following rule checks and human review, the audit identifies 298 responses (9.0%) as unsupported and 415 (12.5%) as unassessable. These labels concern observable support and do not establish whether a correct answer resulted from guessing. APR continues to credit every correct final answer.

E.4 Primary Failure Attribution

For the 144 incorrect non-zero-tool trajectories, we manually verify the primary failure using a sample balanced across eight models and task categories. GLM-5.1 supplies an initial label, after which we inspect every trajectory, intermediate result, and final response. Table 9 reports one primary category per trajectory; its percentages characterize this balanced audit sample rather than corpus-wide failure prevalence.

Tool selection accounts for 24.31% of primary failures. Intermediate-result interpretation, premature termination, and parameterization together account for 56.25%. The audited failures therefore involve not only choosing an operation but also interpreting its output and completing a sufficiently supported solution. This breakdown complements the qualitative examples in Fig. 6 and does not infer failure solely from disagreement with a reference trajectory.

Table 9: **Primary failures in 144 incorrect non-zero-tool trajectories.** Percentages may not sum to exactly 100% because of rounding.

Primary failure	Share (%)
Intermediate-result interpretation	28.47
Tool selection	24.31
Premature termination	20.14
Visual perception	13.89
Parameterization	7.64
Answer extraction	2.78
Redundant trial-and-error	2.78

E.5 USE OF SUPPLIED REFERENCE TOOLS

Strong Prompt + Ref. Tools (Fig. 7) supplies the ordered operation names from one reference path while retaining all 35 tools. This guidance reduces selection and ordering burden; parameterization, execution, interpretation, and verification remain with the model. It does not ablate tool availability.

We map successful API calls to canonical tool names. *Used* $\geq$*1* is the percentage of trajectories using at least one supplied reference-tool type; *Mean coverage* averages the fraction of distinct supplied types used per trajectory; *Used all* is the percentage using every supplied type. These metrics measure type overlap, not reproduction of call order, repetition, or parameters.

Table 10: **Reference-tool guidance and observed tool use.** Strong APR uses the original strong prompt; +Ref. APR additionally supplies the reference operation sequence. All numerical entries are percentages.

Model	Strong APR	+Ref. APR	Used $\geq$1	Mean coverage	Used all
DeepEyes	29.26	29.27	70.00	36.70	12.50
Qwen3-VL-30B-A3B	33.24	35.88	79.08	47.81	18.69
Gemini-3.0-Flash	50.74	52.06	64.15	20.74	1.48

Models use at least one supplied type in 64.15%–79.08% of trajectories, but all types in only 1.48%–18.69%. Limited gains may therefore reflect incomplete adoption of guidance as well as parameter or interpretation errors, rather than execution failure after a fully followed plan. Alternative routes remain valid; App. E.4 provides complementary failure attribution.

E.6 APPLICATION-LEVEL GIMP PILOT

We conduct a pilot in GIMP, an independent image-editing environment, to examine whether model rankings on VTC-Bench are consistent with an application-oriented setting. The pilot contains 20 previously unseen tasks covering document/photo restoration, product/region editing, privacy-preserving local manipulation, and diagram annotation. We expose 19 GIMP operations as structured tools through its Python-Fu/PDB interface, covering geometry, color and tonal adjustment, selection and local editing, annotation, and export.

Models receive natural-language goals without operation coordinates, freely compose the tools, and receive intermediate images after each operation. We evaluate final outputs using mean structural similarity (SSIM), without requiring a match to an expert tool trajectory. Table 11 compares these results with VTC-Bench Code APR.

Table 11: **Application-level pilot on 20 GIMP tasks.** The first four rows use the same model in both environments. The last row pairs Gemini-3.0-Pro on VTC-Bench with its successor Gemini-3.1-Pro on GIMP.

Model	VTC-Bench Code APR (%)	GIMP mean SSIM
GPT-o4-mini	33.68	0.8233
Qwen3-VL-235B-Thinking	34.41	0.8536
GPT-5.2	44.56	0.8621
Gemini-3.0-Flash	50.59	0.8822
Gemini-3.0-Pro / Gemini-3.1-Pro	51.18	0.9002

Gemini-3.0-Pro was unavailable during the pilot, so its GIMP result uses Gemini-3.1-Pro. Including this successor pair, the five-row rankings match exactly (Spearman $\rho = 1.00$, exact two-sided $p = 0.017$). The four same-version rows also preserve their ordering, but the reported test uses all five rows. The small task and model samples, version substitution, and use of image similarity rather than task success limit this result to preliminary cross-environment rank consistency. It does not establish general predictive validity across GIMP, Photoshop, or CAD workflows.

E.7 Execution Length across Benchmarks

Table 12 compares mean execution lengths for three models on AgentVista (Su et al., 2026) and VTC-Bench. We use the reported AgentVista results and our Interface results, with interaction budgets of 30 and 20 rounds, respectively.

Table 12: **Mean model execution length on AgentVista and VTC-Bench.**

Model	AgentVista (turns)	Ours (calls)
GPT-5.2	13.85	13.23
o4-mini	1.89	3.63
Gemini-3-Pro	6.67	5.04

E.8 Reference Tool-Call Distribution

The 680 reference trajectories contain 3,428 annotated tool calls. Figure 11 shows their distribution, counting repeated calls within each trajectory. The five most frequent annotation labels account for 51.95% of calls, and the ten most frequent account for 77.51%.

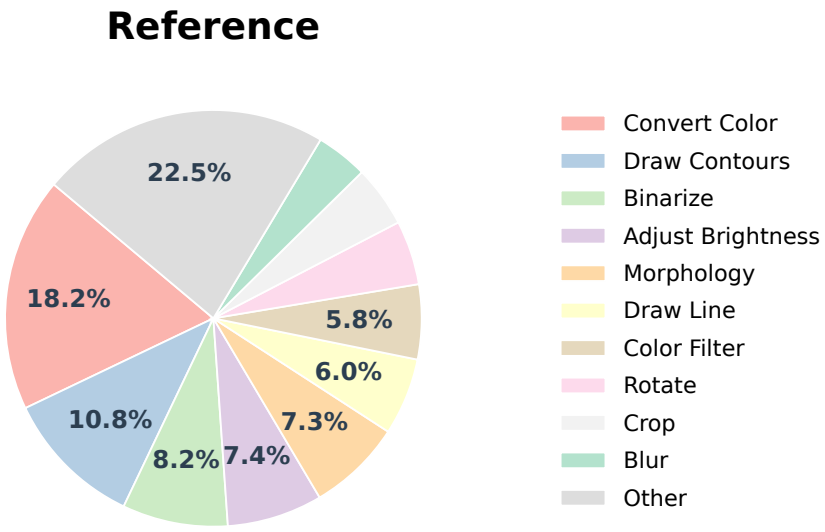


Figure 11: **Tool-call distribution in reference trajectories.** The ten most frequent annotation labels are shown individually; the remaining 17 are grouped as Other. Convert Color groups color-space conversions.

F DETAILED PROMPTS

System Prompt

System Prompt (Strong)

Your role is that of a helpful assistant specialized in solving real-world visual problems using image processing tools. Answer questions about images by combining your visual understanding with the precision of available tools.
Please follow this structured thinking process and show your work.
Start an iterative loop for each question:
- **First, look closely:** Begin with a detailed description of the image in the context of the user's real-life query. List what is immediately visible, and explicitly identify what specific visual details need to be clarified, adjusted, or measured using tools.
- **Next, apply tools:** Select and invoke just one appropriate function to process the image.
- **Then, review the findings:** Carefully analyze the tool's output (e.g., the processed image, detected features, or status) and decide on your next action (e.g., did the rotation fix the view? do you need further adjustments?).
Continue this loop until you have sufficient information.
To finish, bring everything together in a clear, synthesized answer that fully responds to the user's question.

System Prompt (Weak)

Your role is that of a helpful assistant specialized in solving real-world visual problems using image processing tools.

User Prompt

User Prompt without GT Toolchains

<image>
{question}
User Image Path: {image_path}
User Image Size: {image_size}
Output Format (strict adherence required):
<think>Your detailed reasoning process, should go here.</think>
<answer>Your final answer to the user's question goes here.</answer>

User Prompt with GT Toolchains

<image>
{question}
User Image Path: {image_path}
User Image Size: {image_size}
Here is the **Reference Trajectory. You can refer to this trajectory to use tools to solve problems : {reference_trajectory}**
Output Format (strict adherence required):
<think>Your detailed reasoning process, should go here.</think>
<answer>Your final answer to the user's question goes here.</answer>

Code Interpreter Description

Executes Python code to handle images based strictly on the allowed capabilities listed below.

CRITICAL INSTRUCTIONS FOR MODEL:

1. RESTRICTED SCOPE: You are strictly permitted to implement only the specific image handling operations defined in the Allowed Capabilities Reference section below. Do not generate code for operations outside this list.
2. NO PRE-DEFINED FUNCTIONS: The list below defines behaviors, NOT callable functions.
   - INCORRECT: Calling colorspace_gray(image) or resize(image, param). These functions DO NOT exist.
   - CORRECT: You must write the raw Python code using the cv2 library to implement the logic described. For example, to achieve colorspace_gray, you must write cv2.cvtColor(image, cv2.COLOR_BGR2GRAY).
3. PARAMETER ADHERENCE: When writing the code, you must strictly follow the parameter logic described in the reference list (e.g., input ranges, default values, and calculation logic).

Allowed Capabilities Reference (Implement these using raw cv2 code)

1. COLOR SPACE CONVERSION TOOLS

colorspace_gray
Description: Converts the image to grayscale color space. Useful for enhancing contrast or isolating intensity information.
Parameters:

- image (string, required): The input image identifier
- param (object, optional): Empty parameter object

colorspace_hsv
Description: Converts the image to HSV (Hue, Saturation, Value) color space. Useful for color-based segmentation or isolating specific color channels.
Parameters:

- image (string, required): The input image identifier
- param (object, optional): Empty parameter object

colorspace_lab
Description: Converts the image to LAB color space. Useful for perceptual color operations or isolating specific color channels.
Parameters:

- image (string, required): The input image identifier
- param (object, optional): Empty parameter object

2. GEOMETRIC TRANSFORMATION TOOLS

resize
Description: Resizes the image to specified dimensions or by a preset scale (half or double).
Parameters:

- image (string, required): The input image identifier
- param (object, required): width (integer), height (integer), preset (string, optional)

rotate
Description: Rotates the image by specified angle in degrees (clockwise).
Parameters:

- image (string, required): The input image identifier
- param (object, required): angle (number, required)

translate
Description: Shifts the image by a specified number of pixels in the specified direction.
Parameters:

- image (string, required): The input image identifier
- param (object, required): direction (string, optional), distance (integer, optional)

flip
Description: Flips the image horizontally or vertically.
Parameters:

- image (string, required): The input image identifier
- param (object, required): direction (string, optional)

crop
Description: Crops a rectangular region from the image.
Parameters:

- image (string, required): The input image identifier
- param (object, optional): x (integer), y (integer), width (integer), height (integer)

zoom_in
Description: Zooms into a specific region of the image by cropping and optionally resizing.
Parameters:

- image (string, required): The input image identifier
- param (object, optional): x, y, width, height, scale, target_width, target_height

3. FILTERING AND SMOOTHING TOOLS

blur
Description: Applies blurring to reduce noise or smooth the image. Supports average, gaussian, median, and bilateral.
Parameters:

- image (string, required): The input image identifier
- param (object, required): method (string), ksize (integer), sigma_x, sigma_y, d, sigma_color, sigma_space

denoise
Description: Applies fast non-local means denoising.
Parameters:

- image (string, required): The input image identifier
- param (object, optional): mode, h, h_color, template_window, search_window, channels

4. THRESHOLDING AND BINARIZATION TOOLS

threshold
Description: Applies thresholding to create a binary image.
Parameters:

- image (string, required): The input image identifier
- param (object, required): mode, invert, color_mode, threshold_value, adaptive_block_size, adaptive_constant, channels

inrange_color
Description: Creates a mask for pixels within a specified color range.
Parameters:

- image (string, required): The input image identifier
- param (object, optional): colorspace, lower, upper, output_format

5. MORPHOLOGICAL OPERATIONS

morphology
Description: Applies morphological operations (erode, dilate, open, close).
Parameters:

- image (string, required): The input image identifier
- param (object, optional): op, kernel_size, iterations, kernel_shape

6. EDGE DETECTION TOOLS

gradients
Description: Computes gradient images using Sobel or Laplacian operator.
Parameters:

- image (string, required): The input image identifier
- param (object, required): mode (sobel_x, sobel_y, laplacian)

canny
Description: Detects edges using Canny edge detector.
Parameters:

- image (string, required): The input image identifier
- param (object, optional): preset, threshold_low, threshold_high, color_mode, channels

convertscaleabs
Description: Converts image to absolute value and scales it to 0-255 range.
Parameters:

- image (string, required): The input image identifier
- param (object, optional): alpha, beta

7. CONTOUR DETECTION AND ANALYSIS TOOLS

contours
Description: Finds contours using Canny edges and returns bounding boxes/areas.
Parameters:

- image (string, required): The input image identifier
- param (object, optional): mode, canny_low, canny_high, rank, max_contours

contour_area
Description: Calculates contour areas using cv2.contourArea.
Parameters:

- image (string, required): The input image identifier
- param (object, optional): mode, canny_low, canny_high, rank, max_contours, color, thickness

arc_length
Description: Computes contour perimeters using cv2.arcLength.
Parameters:

- image (string, required): The input image identifier
- param (object, optional): mode, canny_low, canny_high, rank, max_contours, color, thickness

approx_poly
Description: Approximates contours with fewer points using cv2.approxPolyDP.
Parameters:

- image (string, required): The input image identifier
- param (object, optional): epsilon_ratio, mode, canny_low, canny_high, rank, max_contours, color, thickness

connected_components_with_stats
Description: Finds and analyzes connected components in a binary image.
Parameters:

- image (string, required): The input image identifier
- param (object, optional): threshold_mode, threshold_value, connectivity, max_components

8. SHAPE DETECTION TOOLS

hough_lines
Description: Detects line segments in the image using Hough transform.
Parameters:

- image (string, required): The input image identifier
- param (object, optional): threshold, minLineLength, maxLineGap, canny_low, canny_high, max_lines

hough_circles
Description: Detects circles in the image using Hough transform.
Parameters:

- image (string, required): The input image identifier
- param (object, optional): dp, minDist, param1, param2, minRadius, maxRadius, max_circles

9. HISTOGRAM AND CONTRAST ENHANCEMENT

histogram
Description: Applies histogram equalization or CLAHE to enhance image contrast.
Parameters:

- image (string, required): The input image identifier
- param (object, optional): mode, color_mode, channels, clip_limit, tile_grid_size

10. IMAGE SEGMENTATION TOOLS

watershed
Description: Applies watershed segmentation to separate overlapping objects.
Parameters:

- image (string, required): The input image identifier
- param (object, optional): max_regions

grabcut
Description: Performs foreground/background segmentation using GrabCut algorithm.
Parameters:

- image (string, required): The input image identifier
- param (object, required): preset (center, tight, loose)

floodfill
Description: Performs flood fill operation starting from a seed point.
Parameters:

- image (string, required): The input image identifier
- param (object, optional): x, y, loDiff, upDiff, newVal, flags

11. ADVANCED OPERATIONS

pyramid
Description: Applies image pyramid operations (upsample or downsample).
Parameters:

- image (string, required): The input image identifier
- param (object, required): mode (pyr_up, pyr_down)

dft
Description: Computes and visualizes the Discrete Fourier Transform magnitude spectrum.
Parameters:

- image (string, required): The input image identifier
- param (object, optional): Empty parameter object

template_match
Description: Matches a template image within the source image.
Parameters:

- image (string, required): The input image identifier
- param (object, optional): template_path, method

features
Description: Detects and draws keypoints using various feature detection methods.
Parameters:

- image (string, required): The input image identifier
- param (object, optional): method (harris, sift, orb, etc.), max_points

inpaint
Description: Inpaints (fills) regions in the image using automatically generated mask.
Parameters:

- image (string, required): The input image identifier
- param (object, optional): preset, canny_low, canny_high, threshold_value, radius, method

12. DRAWING TOOLS

draw_line
Description: Draws a line segment between two points.
Parameters:

- image (string, required): The input image identifier
- param (object, required): x1, y1, x2, y2, color, thickness

draw_circle
Description: Draws a circle on the image at specified center coordinates.
Parameters:

- image (string, required): The input image identifier
- param (object, optional): x, y, radius, color, thickness

G DETAILED TASK EXAMPLE

			
Question: Please find the slogan board on the low wall near the girl wearing light blue clothes in this image, read the text on it, and answer what the last 5 letters are.	**Question:** Please find the orange book on the third shelf from the top on the right side of the bookshelf opposite the cabinet labeled 'Luisterpunt' in the picture. Read the text on it and answer what the book is titled.	**Question:** Please find the glider in the sky in this image, read the text on it, and answer what the text content is.	**Question:** Please find the text in blue font on the sail in this image, read the text on it, and answer what the text says.
Reference Tool Trajectory: Rotate, Crop, Convert Color, Binarize	**Reference Tool Trajectory:** Crop, Resize, Convert Color, Adjust Brightness, Sharpen	**Reference Tool Trajectory:** Crop, Resize, Flip	**Reference Tool Trajectory:** Crop, Flip, Rotate, Convert Color, Color Filter, Convert Color, Binarize
Answer: ataud	**Answer:** IN BRUSSEL	**Answer:** Windy	**Answer:** ebasto

Figure 12: Supplementary examples of Attention Focusing tasks.

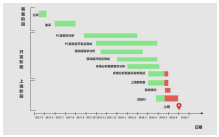


Question:
Please calculate the proportion of Greece among European countries with coal production exceeding 5 million tons in 2020.

Options:
(A)2.0 (B)1.5
(C)2.5 (D)3.0

Reference Tool Trajectory:
Convert Color, Color Filter, Morphology, Draw Contours, Measure Area

Answer: (A)2.0



Question:
Please find the longest task cycle.

Options:
(A)5 (B)6
(C)5.5 (D)6.5

Reference Tool Trajectory:
Convert Color, Color Filter, Morphology, Draw Contours, Measure Area

Answer: (B)6

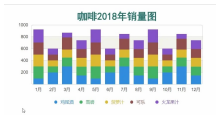


Question:
Please calculate the percentage of Mumbai relative to New Delhi?

Options:
(A)83 (B)82
(C)84 (D)85

Reference Tool Trajectory:
Rotate, Adjust Brightness, Convert Color, Binarize, Draw Contours, Measure Area

Answer: (C)84



Question:
What is the total sales volume of pineapple juice in January and November?

Options:
(A)360 (B)340
(C)330 (D)350

Reference Tool Trajectory:
Convert Color, Color Filter, Draw Contours, Measure Area, Draw Line

Answer: (C)330

Figure 13: Supplementary examples of Chart tasks.

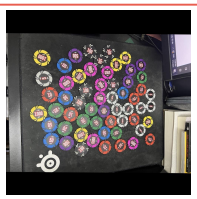


Question:
How many yellow chips are there in the stack of chips on the right?

Options:
(A)8 (B)10
(C)7 (D)9

Reference Tool Trajectory:
Rotate, Convert Color, Color Filter, Morphology, Draw Contours

Answer: (A)8



Question:
How many white chips are in the picture?

Options:
(A)5 (B)7
(C)9 (D)11

Reference Tool Trajectory:
Rotate, Crop, Convert Color, Color Filter, Morphology, Draw Contours

Answer: (C)9

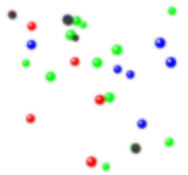


Question:
Excluding the black background, what is the proportion of blue in the image?
A B C D

Options:
(A) 58%-60% (B) 60%-62%
(C) 54%-56% (D) 56%-58%

Reference Tool Trajectory:
Blur, Convert Color, Color Filter, Morphology, Draw Contours, Measure Area

Answer: (C) 54%-56%



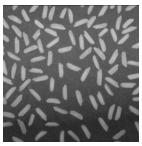
Question:
Please use the HSV color space to count beans of different colors. How many blue beans are there in the picture?

Options:
(A)9 (B)6
(C)8 (D)7

Reference Tool Trajectory:
Convert Color, Color Filter, Morphology, Draw Contours

Answer: (D)7

Figure 14: Supplementary examples of Color tasks.



Question:
Count the total number of rice grains in the picture.

Options:
(A)110 (B)100
(C)95 (D)105

Reference Tool Trajectory:
Convert Color, Blur, Binarize, Morphology, Watershed, Draw Contours

Answer:(B)100



Question:
Please count the total number of players on the field in the image. It is recommended to use color filtering or edge detection to assist in counting.

Options:
(A)17 (B)14
(C)15 (D)16

Reference Tool Trajectory:
Convert Color, Blur, Edge Detect, Binarize, Morphology, Draw Contours

Answer: (D)16



Question:
Please count the number of white cotton swab heads inside the circular container in the image. The cotton swabs are densely arranged in the container, presenting a dense white dotted texture. Please carefully identify each individual cotton swab head and count them.

Options:
(A) 385 (B) 415
(C) 400 (D) 360

Reference Tool Trajectory:
Convert Color, Blur, Binarize, Morphology, Watershed, Draw Contours

Answer: (A) 385



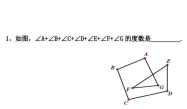
Question:
Please count the number of birds flying in the sky in the image. The image contains a large number of scattered black bird silhouettes. Please carefully identify all targets and count them.

Options:
(A)153 (B)189
(C)177 (D)165

Reference Tool Trajectory:
Convert Color, Binarize, Morphology, Draw Contours

Answer: (C)177

Figure 15: Supplementary examples of Counting tasks.



Question:
Solve the math problem in the diagram.

Reference Tool Trajectory:
Convert Color, Draw Contours, Edge Detect, Draw Line

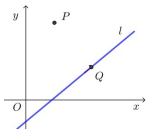
Answer: 540



Question:
Please calculate the number of edges of the shape in the image.

Reference Tool Trajectory:
Convert Color, Binarize, Edge Detect, Line Detect

Answer: 54

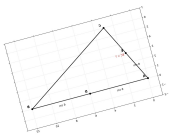


Question:
What is the distance from point P to line l?

Options:
(A) 135 (B) 125
(C) 140 (D) 130

Reference Tool Trajectory:
Convert Color, Edge Detect, Line Detect, Draw Line

Answer: (D) 130



Question:
As shown in the figure, in △ABC, D and E are on AB and AC respectively, and DE∥BC. Find the length of EC (rounded to two decimal places).

Options:
(A) 3.43 (B) 4.03
(C) 3.63 (D) 3.83

Reference Tool Trajectory:
Rotate, Crop, Convert Color, Binarize, Line Detect, Draw Line

Answer: (A) 3.43

Figure 16: Supplementary examples of Math tasks.



Question:
What is the inner diameter in pixels of the largest nut in the picture?

Options:
(A) 160 (B) 170
(C) 180 (D) 190

Reference Tool Trajectory:
Convert Color, Binarize, Morphology, Draw Contours, Measure Area, Draw Circle

Answer: (B) 170

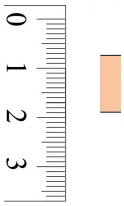


Question:
What is the length in pixels of the black screw in the upper right corner of the table?

Options:
(A) 115 (B) 145
(C) 125 (D) 135

Reference Tool Trajectory:
Crop, Convert Color, Binarize, Morphology, Draw Contours, Approximate Polygon

Answer: (C) 125

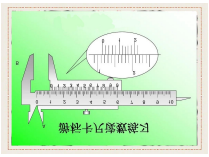


Question:
Measure the length of the wooden block in the figure. How many cm is it?

Options:
(A) 1.25 (B) 1.05
(C) 1.15 (D) 1.35

Reference Tool Trajectory:
Rotate, Draw Contours, Draw Line, Draw Line

Answer: (C) 1.15



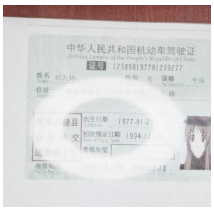
Question:
Measure the length shown by the vernier caliper in the figure. How many mm is it?

Options:
(A) 11.14 (B) 11.25
(C) 11.35 (D) 11.05

Reference Tool Trajectory:
Rotate, Flip, Convert Color, Binarize, Line Detect, Draw Line

Answer: (A) 11.14

Figure 17: Supplementary examples of Measure tasks.



Question:
When does the validity period of the certificate start and end?

Reference Tool Trajectory:
Adjust Brightness, Convert Color, Histogram Eq, Sharpen, Crop, Binarize

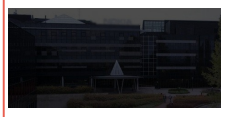
Answer:
1994-12-18 to 2000-12-18



Question:
What is the license plate number shown in the picture?

Reference Tool Trajectory:
Crop, Resize, Convert Color, Adjust Brightness, Binarize, Morphology

Answer: MH02DN6980



Question:
Which company's building is this?

Reference Tool Trajectory:
Adjust Brightness, Convert Color, Histogram Eq, Sharpen, Binarize

Answer: NOKIA

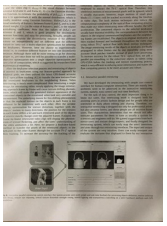


Question:
Who is the preparer of this form?

Reference Tool Trajectory:
Rotate, Adjust Brightness, Convert Color, Binarize

Answer: 任璐

Figure 18: Supplementary examples of OCR tasks.



Question:
This is a photo that is partially obscured by shadows. Please tell me what the third-to-last word in the 13th row is.

Reference Tool Trajectory:
Convert Color, Histogram Eq, Sharpen, Binarize, Crop

Answer: the

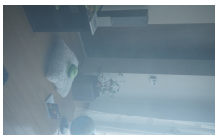


Question:
This is a photo that is covered with haze. Please tell me how many cars are in the image.

Options:
(A) 3 (B) 4
(C) 5 (D) 6

Reference Tool Trajectory:
Histogram Eq, Convert Color, Crop

Answer: (D) 6

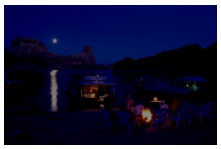


Question:
On the globe, which continent is facing us?

Options:
(A) Eurasia (B) North America
(C) Africa (D) Antarctica

Reference Tool Trajectory:
Rotate, Histogram Eq, Adjust Brightness, Crop

Answer: (A) Eurasia



Question:
How many people are sitting on the chair in the picture?

Options:
(A) 3 (B) 2
(C) 4 (D) 1

Reference Tool Trajectory:
Adjust Brightness, Convert Color, Crop , Edge Detect, Draw Contours

Answer: (C) 4

Figure 19: Supplementary examples of Perceptual Restoration tasks.



Question:
Is the floor in the picture north-south or east-west oriented?

Options:
(A) It is uncertain
(B) East-west (C) North-south

Reference Tool Trajectory:
Rotate, Convert Color, Line Detect

Answer: (C) North-south



Question:
Between the red door and the yellow door on two adjacent floors, which color door is on the lower floor?

Options:
(A) They are on the same floor
(B) Red door (C) Yellow door

Reference Tool Trajectory:
Flip, Convert Color, Color Filter, Morphology, Draw Contours

Answer: (C) Yellow door

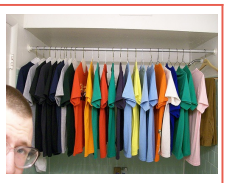


Question:
Using the vertical line at the center of the image as a reference, how many blue lights are on the left side?

Options:
(A) 2 (B) 4
(C) 3 (D) 1

Reference Tool Trajectory:
Crop, Blur, Convert Color, Color Filter, Morphology, Draw Contours

Answer: (C) 3



Question:
Counting from right to left, what is the color of the clothes to the right of the third green clothing item?

Options:
(A) Orange-yellow (B) Black
(C) Green (D) Red

Reference Tool Trajectory:
Blur, Convert Color, Edge Detect, Morphology, Draw Contours

Answer: (B) Black

Figure 20: Supplementary examples of Spatial Reasoning tasks.

H DETAILED MODEL REASONING PROCESS

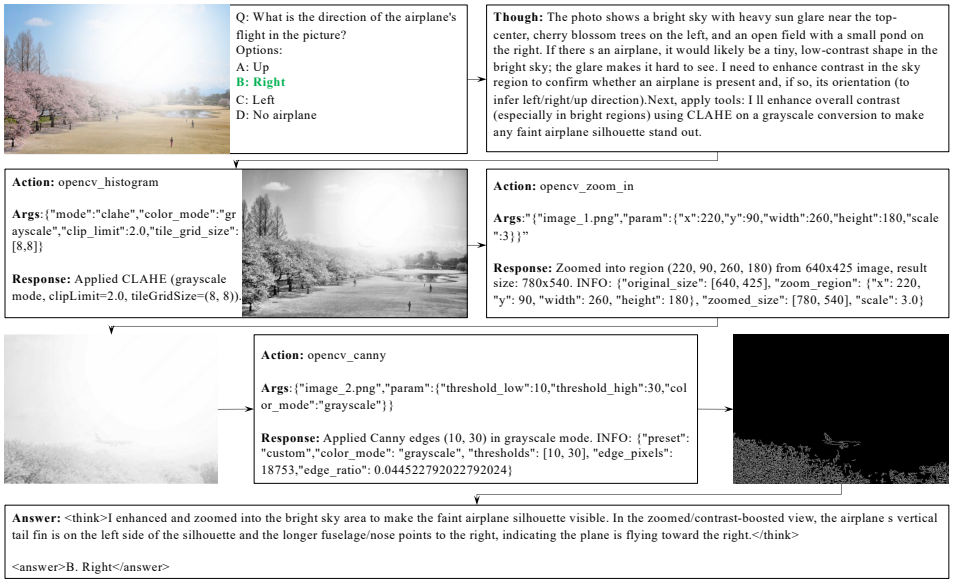


Figure 21: A case study of the reasoning process for Perceptual Restoration tasks.



Figure 22: A case study of the reasoning process for Measure tasks.