VisTRA: Visual Tool-use Reasoning Analyzer for Small Object Visual Question Answering

Hiroaki Sugiyama[1]*, Ko Koga[2]*, Toshifumi Nishijima[2],
[1]NTT, Inc, [2]Toyota Motor Corporation

Abstract

This study proposes VisTRA (Visual Tool-use Reasoning Analyzer), a framework for analyzing how Visual Language Models (VLMs) utilize tools in Visual Question Answering (VQA) tasks involving small objects in high-resolution images. While tools like object detection and zoom functionality are essential for small object VQA, their potential errors necessitate careful verification of outputs. Our framework provides systematic evaluation of VLMs' tool-use capabilities through analysis of verification patterns. Using the V* bench dataset, we find that direct acceptance of tool outputs correlates with decreased VQA accuracy, while lower-performing models exhibit higher frequencies of cyclic verification loops. These findings offer insights for improving tool verification mechanisms in VLM architectures focused on small object detection tasks.

1 Introduction

[1] Visual Question Answering (VQA) is a task that requires interpreting visual content in images to generate appropriate responses to natural language questions. Recent advances in Vision-Language Models (VLMs), which integrate Large Language Models (LLMs) with vision capabilities, have improved VQA performance (Li et al., 2022; Liu et al., 2023; Dai et al., 2023; Alayrac et al., 2022).

A well-known challenge in VQA is handling high-resolution images containing small objects (Wu and Xie, 2024; Kisantal et al., 2019). The conventional approach of feeding entire images into VQA systems faces an inherent limitation: when processing images holistically, small objects that occupy only a minimal portion of the total pixels become practically indistinguishable in the global representation. This fundamental constraint of whole-image processing leads to systematic failures in capturing small objects, resulting in degraded response accuracy.

Several approaches have been proposed to address the small object challenge, including analyzing magnified portions of images (Singh et al., 2018; Carion et al., 2020), additional training with small object-focused datasets (Bosquet et al., 2023), and multi-scale learning that preserve native resolution while extracting features from numerous image sub-crops (Thapa et al., 2024; Huang et al., 2025). While these methods improve recognition accuracy, they introduce new challenges such as increased architectural complexity and limitations in detection targets.

A promising direction for addressing these limitations has emerged from recent advances in LLM capabilities. Modern LLMs demonstrate *agentic reasoning* capabilities, autonomously planning and utilizing various external tools to solve complex tasks (Wu et al., 2023b; Yao et al., 2022). This tool-use paradigm has been successfully applied to enhance LLMs' capabilities across various domains (Schick et al., 2023; Qin et al., 2025; Mialon et al., 2023), leading to its application in VQA tasks (Yang et al., 2023; Gupta and Kembhavi, 2023; Surís et al., 2023).

In small object VQA, systems utilizing detection and sliding window tools can identify objects that the core VLM might miss. However, these tool-augmented approaches face a critical challenge: VLMs often exhibit **direct acceptance** of tool outputs without proper verification (Yang et al., 2023; Hu et al., 2024), making them vulnerable to false positives and missed detections (Lu et al., 2023; Wu et al., 2023a). To address this issue, VLMs need to not only verify tool outputs but also adaptively refine their approach based on the verification results (Huang et al., 2023; Singh et al., 2023). When initial tool outputs prove unreliable, effective **re-planning**—such as adjusting detection parameters or switching to alternative

[1]* These two authors contributed equally to this work.

Proceedings of the 1st Workshop for Research on Agent Language Models (REALM 2025), pages 356–366

analysis methods—becomes crucial. While studies demonstrate improved accuracy through iterative refinement with feedback (Wang et al., 2024; Shinn et al., 2023), current VLMs struggle to consistently implement such adaptive reasoning and strategic re-planning (Madaan et al., 2023).

In this study, we propose **VisTRA** (**Vis**ual **T**ool-use **R**easoning **A**nalyzer), a systematic framework for analyzing and classifying VLMs' behavioral patterns in tool utilization, with particular focus on the relationship between VQA accuracy and two critical aspects: **direct acceptance** of tool outputs and **re-planning** capabilities. Leveraging the V* bench dataset (Wu and Xie, 2024), which features abundant small objects, we conduct a comprehensive analysis of how VLMs process external vision tool outputs and handle potential misrecognitions.

The main contributions of this paper are:

- Development of a systematic framework for quantitative analysis of VLM behavior in visual tool utilization, enabling detailed examination of decision-making patterns.
- Quantitative analysis of direct acceptance and re-planning in small object VQA, revealing their relationship with VQA accuracy.
- Public release of the framework (prompt) to facilitate reproducibility and further research in the field[2].

Our analysis reveals that direct acceptance patterns strongly correlate with decreased VQA accuracy, while lower-performing VLMs tend to exhibit ineffective re-planning through cyclic verification loops.

2 Related Work

2.1 Tool Use and Agentic Reasoning in Large Language Models

The utilization of external tools by LLMs has emerged as a key approach to expanding their capabilities beyond native text processing (Mialon et al., 2023; Qin et al., 2025). Concurrent with this development, LLMs have demonstrated enhanced *agentic reasoning*—the ability to perform autonomous multi-step reasoning processes, as exemplified by sophisticated Chain-of-Thought capabilities in models like OpenAI's o1. This combination of tool utilization and agentic reasoning enables complex problem-solving sequences (Ruan et al., 2023; Wang et al., 2024).

[2]http://github.com/nttcslab/vistra/

The incorporation of external tools, while expanding LLMs' capabilities, introduces critical challenges in output verification. A significant concern is direct acceptance—the tendency of LLMs to incorporate tool outputs without proper verification (Ruan et al., 2024). This challenge becomes particularly acute when tools produce unexpected or incorrect outputs, potentially compromising the entire reasoning process.

To address these verification challenges, researchers have developed systematic approaches incorporating self-verification mechanisms. Methods such as Self-Reflection (Shinn et al., 2023) and Feedback-Driven Self-Improvement (Wang et al., 2024) enable LLMs to critically evaluate their outputs and reasoning paths. These verification mechanisms become particularly crucial in complex scenarios involving multiple tool interactions.

2.2 Tool Use Application and Challenges in VQA

Visual Question Answering (VQA) represents a domain where tool integration demonstrates particular promise (Liu et al., 2023). Current Visual Language Models (VLMs) face significant challenges in tasks involving complex visual elements, especially with numerous objects, text recognition, or small objects. Research by (Ye et al., 2025) has identified specific limitations in counting, spatial relationship reasoning, and instruction grounding tasks. These challenges have motivated the integration of specialized tools such as object detection (Ren et al., 2017), OCR (Tanaka et al., 2024), and segmentation (Kirillov et al., 2023).

The Visual Sketchpad framework (Hu et al., 2024) represents a significant advancement in addressing these challenges by extending VLMs' spatial reasoning capabilities. By implementing a step-by-step reasoning approach with multiple image processing tools, it has demonstrated improved performance in small object detection and spatial relationship understanding. However, the framework's tendency to accept tool outputs without verification highlights the need for robust cross-validation mechanisms.

2.3 Evaluation and Analysis Frameworks for Tool-Augmented Agentic Reasoning

The development of systematic evaluation frameworks for tool-augmented LLMs has produced several notable benchmarks. MINT (Wang et al.,

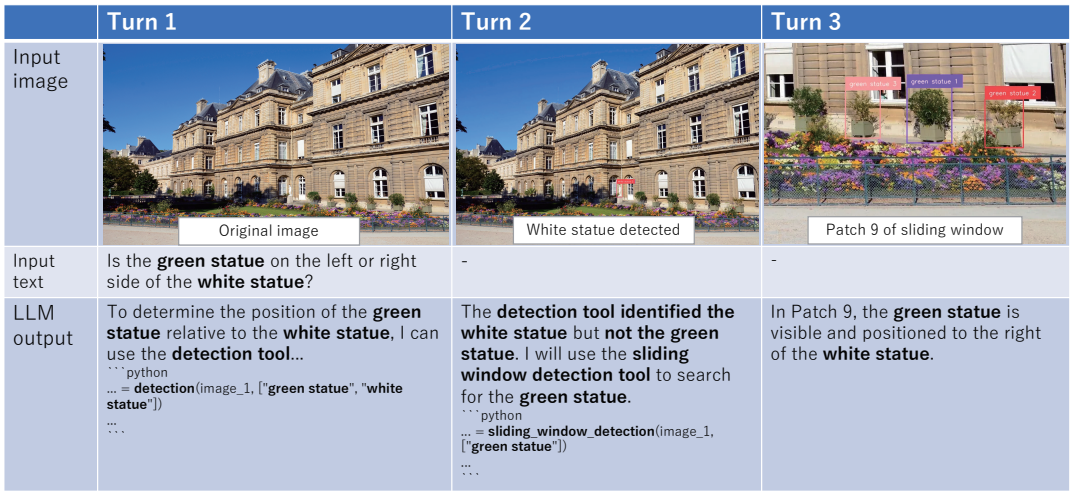


Figure 1: The inference process on the V* bench. A very small green statue is at the left end.

2024) pioneered the analysis of LLM reasoning behavior through multi-turn tool use and feedback mechanisms. This framework has demonstrated quantifiable improvements in performance. ToolComp (Nath et al., 2025) advances this evaluation approach by focusing on sequential tool use scenarios. Through its comprehensive toolkit and process monitoring system, it provides detailed insights into both final outcomes and intermediate reasoning quality.

In the visual domain, V* bench (Wu and Xie, 2024) has established new standards for evaluating high-resolution image processing capabilities. This framework specifically addresses the limitations of traditional vision-language models in processing detailed visual information, evaluating their ability to autonomously deploy and coordinate multiple visual tools. VisualAgentBench (VAB) (Liu et al., 2024) extends evaluation into diverse multimodal environments, from 3D simulations to GUI operations. While revealing promising capabilities in current Large Multimodal Models (LMMs), VAB's findings highlight persistent challenges in robustness.

A critical gap remains in both text and vision-based frameworks: the quantitative analysis of verification and replanning behaviors. Understanding these patterns is essential for developing more reliable tool-augmented systems, particularly in complex visual processing tasks requiring coordination among multiple tools.

3 VisTRA: Visual Tool-use Reasoning Analyzer

This section introduces and details the Visual Tool-use Reasoning Analyzer (VisTRA), our proposed framework for systematically analyzing how Visual Language Models (VLMs) utilize external tools and respond to tool errors during agentic reasoning. Focusing on VQA tasks involving small objects, we analyze how VLMs employ tools such as Detection and SlidingWindow, and how they handle tool misrecognition (through direct acceptance, verification, or re-planning). Our study utilizes the SKETCHPAD framework proposed in VisualSketchpad (Hu et al., 2024) and evaluates it on V* bench (Wu and Xie, 2024), a benchmark for small object VQA. The VisTRA framework records and categorizes each step of a VLM's reasoning process, providing detailed analysis of reasoning transitions.

The following subsections detail our three-stage methodology: first, we collect development and validation data through V* bench; second, we establish a systematic tagging system and transition patterns; and finally, we automate the annotation process using LLMs.

Through this framework, we aim to provide quantitative insights into how VLMs utilize tools and handle potential errors in their reasoning processes, particularly in challenging scenarios involving small object recognition in high-resolution images.

3.1 Case Analysis and Data Collection on V* bench

To develop and validate our framework, we analyze reasoning processes in V* bench tasks using the SKETCHPAD framework. Figure 1 presents a representative case study demonstrating typical reasoning patterns and potential failure modes.

The illustrated task requires determining the

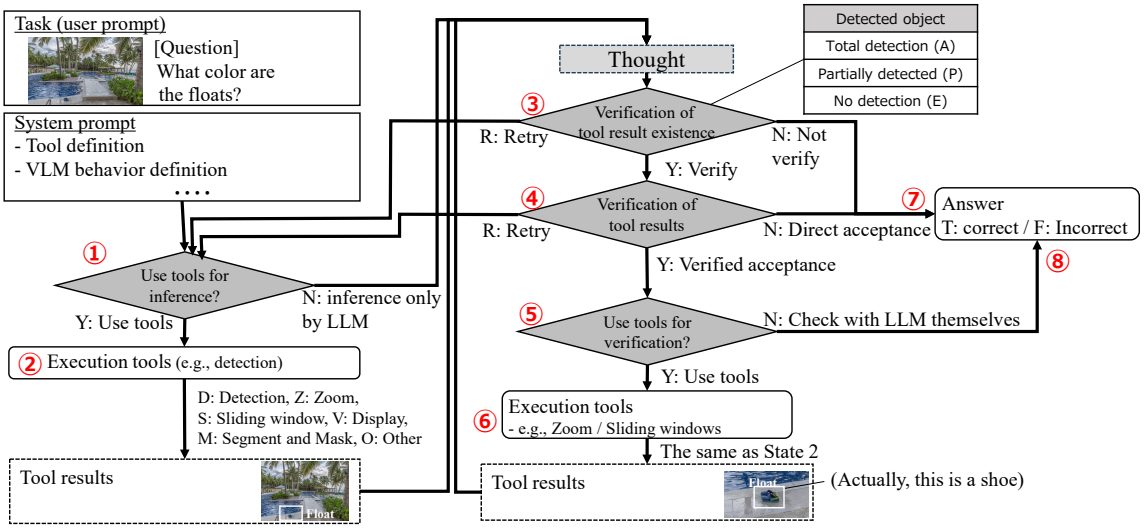


Figure 2: Developed flowchart. Each state in the framework is denoted by a number, and possible actions from that state are represented by action tags (e.g., Y, N). For example, verifying tool output is expressed as `4:Y` (Verified acceptance), while bypassing verification is expressed as `4:N` (Direct acceptance).

spatial relationship between two statues in a high-resolution image. The target objects—a green statue and a white statue—appear at a scale that makes them difficult to identify through VLM alone, with the green statue located at the left edge and the white statue in the center of the image.

The reasoning process proceeds through three turns (Figure 1). In Turn 1, the LLM employs the detection tool to locate both the green and white statues in the input image. Based on this detection result which reveals only the white statue in the center, in Turn 2, the LLM initiates a sliding window detection to search for the missing green statue. With the new detection results where the tool misidentifies a bush as the green statue in patch 9, in Turn 3, the LLM accepts this detection without verification. This leads to an incorrect conclusion that "the green statue is to the right of the white statue," when the actual green statue is on the left.

For our experimental evaluation, we collected data using two LLM variants: base gpt-4o and fine-tuned ft-gpt-4o. The ft-gpt-4o model was trained for one epoch on 10 V* bench examples where direct acceptance errors had been manually corrected. Using these models, we created two datasets: a reference dataset of 20 V* bench examples and a validation dataset of 15 examples. Each example was processed by both models, yielding 40 reasoning sequences for reference and 30 for validation, enabling comprehensive analysis of tool-use and verification patterns.

3.2 Iterative Development of VLM's Tool-use Analysis Framework

Using the reference dataset from Section 3.1, we constructed a flowchart (Figure 2) that captures VLM tool-use behaviors through a tag-based representation and transition patterns. Our development process began with designing a preliminary flowchart focused on detecting direct acceptance and re-planning behaviors. Since LLMs often combine input interpretation and action decisions within their reasoning steps, we established a fine-grained annotation scheme that operates at a sub-step level. Two expert annotators independently labeled the actions in each VLM reasoning step, followed by three cycles of discussion and consensus building:

Cycle 1 The first cycle revealed significant discrepancies in annotation practices, with agreement rates of 13/20 for gpt-4o samples and 7/20 for ft-gpt-4o samples. Analysis of these differences highlighted several key issues, particularly in state 3 annotations involving object detection outcomes. Through discussion, we established clear definitions for object detection outcomes in state 3: no detection (E), partial detection (P), and complete detection (A). We also standardized tool notation for state 2 and 6: **D: Detection, Z: Zoom, S: Sliding window and detection, V: Display, M: Segmentation and mask, O: Other**.

Cycle 2 The second cycle showed substantial improvement in agreement rates, reaching 20/20

for gpt-4o samples and 19/20 for ft-gpt-4o samples. During this phase, we finalized several critical aspects of the flowchart. We clarify that our framework analyses exclusively how the LLM interprets tool outputs, explicitly avoiding any human assessment of tool accuracy. We introduced a dedicated verification state (state 3) to check the presence/absence of tool outputs before proceeding to the acceptance phase (state 4). Importantly, the determination of Y/N in this verification state is based on whether the LLM acknowledges the presence of tool outputs, rather than the actual outputs themselves.

Cycle 3 The final cycle achieved complete inter-annotator agreement across all samples. This level of consensus allowed us to finalize our annotation scheme. Our notation represents each flowchart step and transition as `<state:tag>` in Figure 2, where each behavior pattern is encoded through a sequence of state-tag pairs.

For example, the reasoning process illustrated in Figure 1 is represented as:

`1:Y,2:D,3:PR,1:Y,2:S,3:AY,4:N,7:F`

This sequence traces the reasoning path: initial detection (`1:Y,2:D`), partial recognition triggering re-planning (`3:PR`), sliding window application (`1:Y,2:S`), object recognition (`3:AY`), direct answer without verification (`4:N`), and ultimately an incorrect response (`7:F`).

Table 1 provides formal definitions of VLM behaviors and illustrates how these behaviors are encoded in our tag-based representation system. For instance, the example sequence above exhibits direct acceptance behavior, as it reaches state 7 without verified acceptance (3:AY,4:Y). This notation system allows us to systematically identify and quantify such behavioral patterns across different VLMs while maintaining the interpretability of individual decision steps in their tool utilization process.

Table 1: VLM behaviors categories in tool usage. `:X` represents any action tag corresponding to each state number.

Behavior	Definition
Verified acceptance	Verification performed after all targets are found (`3:AY,4:Y`)
Direct acceptance	Reaches `7:X` without any verified acceptance (`3:AY,4:Y`)
Cyclic loop	Contains repeated use of the same tool (`2:X`)
Re-planning	Uses different tools each time (all `2:X` differ)

3.3 Prompt Engineering for Automated Annotation

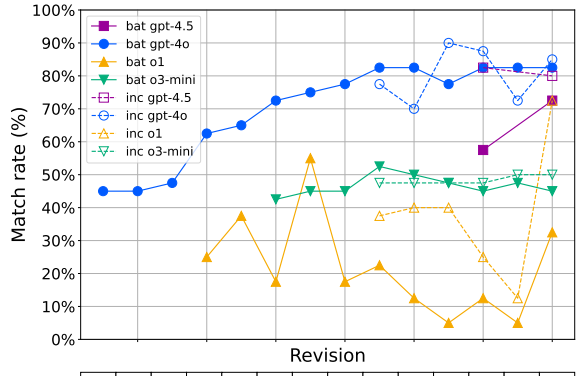
While manual annotation using VisTRA provides detailed insights into VLM behaviors, its time-intensive nature (approximately three hours per 20 samples) necessitates automation for scaling analysis across multiple models. This section describes our systematic approach to automating the VisTRA annotation process through prompt engineering.

Our automation strategy involved iterative refinement of prompts corresponding to VisTRA tag definitions, validated against our reference dataset. The development progressed through multiple phases, initially using only gpt-4o (rev.1-3) before expanding to evaluate generalizability with additional models: o1 (rev.4), o3-mini (rev.6), and gpt-4.5-preview (rev.12). While we primarily focused on batch annotation, where tags were assigned to the entire reasoning process simultaneously, we introduced step-by-step annotation from rev.9 onward.

Key improvements across revisions included:

- **Rev.4**: Restructured the evaluation format to use labeled OBSERVATION, THOUGHT, and ACTION statements as evidence for state transitions, replacing narrative descriptions
- **Rev.5-6**: Refined reasoning step transitions by enforcing single-THOUGHT evaluation per step and implementing clear conditions for terminal state transitions
- **Rev.12-14**: Explicitly defined state-transition constraints between state 3 and 4, and implemented explicit enumeration of target objects to standardize the distinction between partial and complete detection annotations at state 3

Figure 3 shows the progression of automated versus manual annotation agreement rates across revisions. Comparative analysis revealed that while gpt-4.5-preview (purple; square) achieved comparable performance to gpt-4o (blue; circle), both o1 (orange; upper triangle) and o3-mini (green; lower triangle) showed lower accuracy, often due to their tendency to over-analyze and deviate from our defined standards.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14
bat gpt-4.5												0.57		0.72
bat gpt-4o	0.45	0.45	0.47	0.62	0.65	0.72	0.75	0.7	0.85	0.82	0.78	0.82	0.82	0.82
bat o1				0.25	0.38	0.17	0.55	0.17	0.23	0.12	0.05	0.12	0.05	0.33
bat o3-mini						0.42	0.45	0.45	0.53	0.5	0.47	0.45	0.47	0.45
inc gpt-4.5												0.82		0.8
inc gpt-4o									0.78	0.7	0.9	0.88	0.72	0.85
inc o1									0.38	0.4	0.4	0.25	0.12	0.72
inc o3-mini									0.47	0.47	0.47	0.47	0.5	0.5

Figure 3: Improvement through prompt updates

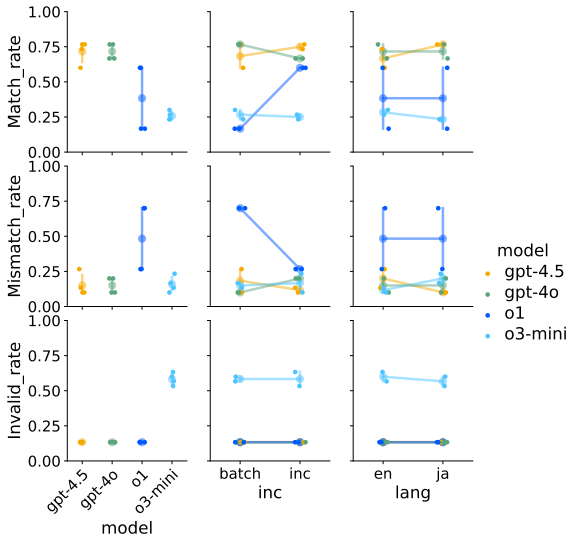


Figure 4: VisTRA's Automated Annotation Accuracy on validation dataset

3.4 Validation of VisTRA's Automated Annotation Accuracy

To validate the reliability of our automated annotation framework, we conducted comprehensive experiments using a validation set distinct from our reference dataset. This validation set comprises 30 reasoning sequences derived from 15 new V* bench examples, each processed by both gpt-4o and ft-gpt-4o models. We evaluated the framework's performance across multiple dimensions: model selection, annotation approaches (batch vs. incremental), and language settings.

Figure 4 presents the match rates between automated and manual annotations under various conditions. Each subplot shows three metrics—match rate, mismatch rate, and invalid rate—across different experimental conditions. The results reveal several key findings:

First, regarding model performance, gpt-4o and gpt-4.5-preview achieved higher match rates compared to reasoning-specialized models (o1 and o3-mini). Despite their focus on reasoning tasks, o3-mini showed notably high invalid rates, while o1 produced a high proportion of mismatches, particularly in batch processing.

Second, when comparing annotation approaches, batch processing proved more effective for gpt-4o, likely due to our focused optimization of batch prompts during development. Conversely, gpt-4.5-preview showed better performance with incremental processing, suggesting its enhanced ability to maintain consistency across multiple reasoning steps.

Finally, the framework demonstrated consistent performance across both English and Japanese inputs, suggesting its potential applicability across different languages without significant degradation in accuracy.

Based on these validation results, we selected gpt-4o with batch processing as our primary configuration for automated annotation. This choice enables reliable, large-scale analysis of VLM's tool-using reasoning patterns across the entire V* bench dataset.

4 Analysis of VLM Reasoning Processes Using VisTRA

4.1 Experimental Setup

We analyze the tool-use behaviors of multiple VLMs on the V* bench dataset using VisTRA, with a focus on quantifying their reasoning patterns during visual question-answering tasks. Our evaluation includes four VLMs: gpt-4o, gpt-4o-mini, gpt-4.5-preview, and a fine-tuned variant of gpt-4o (ft-gpt-4o). We assess these models on 238 V* bench examples, excluding those used in reference and verification datasets. For each example, we record the models' reasoning processes using the SKETCHPAD framework's tools and analyze three key behavioral patterns: direct acceptance (reaching final states without verification), re-planning (switching to different tools), and cyclic loops (repeated use of the same tool) shown in Table 1.

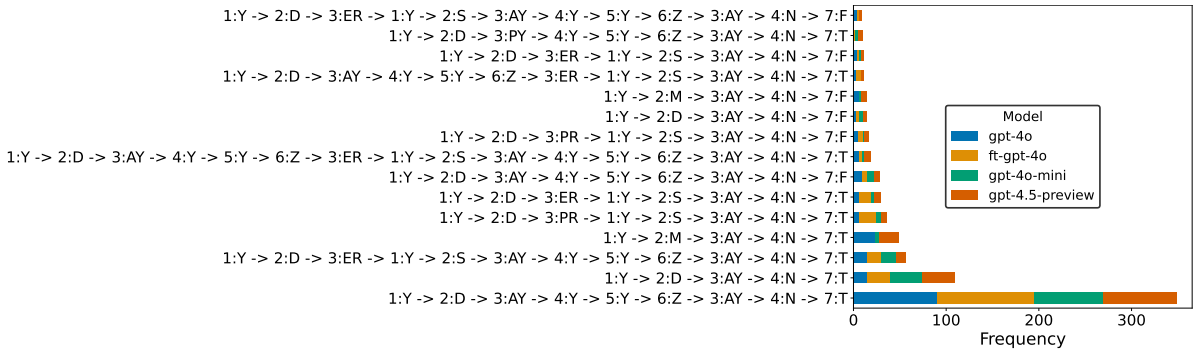


Figure 5: Distribution of common patterns in automated reasoning step annotation

4.2 Analysis of Reasoning Patterns and Performance

Table 2: VQA accuracy and rates of direct acceptance, re-planning, and cyclic loop occurrence for each VLM

Model	VQA Accuracy	Direct Acceptance	Re-planning	Cyclic Loop
gpt-4o	80.1	38.5	27.4	2.21
ft-gpt-4o	85.3	35.3	39.8	5.88
gpt-4o-mini	74.4	37.4	32.8	10.1
gpt-4.5	85.3	50.0	22.3	0.42

Table 2 summarizes the performance metrics of the four evaluated models. Gpt-4.5-preview and fine-tuned gpt-4o achieve the highest VQA accuracy (both 85.3%), followed by the base gpt-4o (80.1%) and gpt-4o-mini (74.4%). The models show considerable variation in their behavioral patterns: gpt-4.5-preview exhibits the highest direct acceptance rate (45.0%) and lowest re-planning rate (22.3%), while fine-tuned gpt-4o shows a lower direct acceptance rate (32.8%) but higher re-planning rate (39.8%). Notably, gpt-4o-mini demonstrates the highest loop rate (10.08%), substantially exceeding other models.

Contrary to our initial hypothesis, we found no clear correlation between overall VQA accuracy and behavioral metrics (direct acceptance rate and re-planning rate) across different model scales. For instance, gpt-4.5-preview achieves the highest accuracy while showing the highest direct acceptance rate and the lowest re-planning rate. Similarly, despite its relatively low direct acceptance rate and moderate re-planning rate, gpt-4o-mini exhibits the lowest accuracy. These results suggest that model capabilities play a more fundamental role in determining VQA performance than these behavioral patterns. In the following sections, we conduct a detailed analysis to understand how these behavioral patterns influence performance within models of similar capabilities.

4.2.1 Direct Acceptance and Verification

Table 3: VQA accuracy by direct / verified acceptance

Model	Direct (%)	Verified (%)
gpt-4o	70.1	86.3
ft-gpt-4o	78.6	89.6
gpt-4o-mini	65.2	79.9
gpt-4.5	80.7	89.9

Analysis of direct acceptance versus verified responses reveals a consistent pattern across all models: direct acceptance of tool outputs leads to lower accuracy compared to cases involving verification steps (Table 3). However, the magnitude of this accuracy drop varies significantly among models. Gpt-4o-mini shows a substantial decline from 79.9% to 65.2% (14.7 percentage points), while gpt-4.5-preview exhibits a more modest decrease from 89.9% to 80.7% (9.2 percentage points). The fine-tuned gpt-4o also demonstrates improved resilience against accuracy degradation, with the gap between verified and direct acceptance (89.6% vs 78.6%) being smaller than its base model (86.3% vs 70.1%). These findings suggest that while verification generally improves performance, more sophisticated models like gpt-4.5-preview and the fine-tuned gpt-4o can better maintain accuracy even when directly accepting tool outputs.

4.2.2 Impact of Re-planning and Verification

Table 4: Correct answer rate (# correct answers / # attempts) for each retry type by model

Model	Straight	Cyclic loop	Re-planning
gpt-4o	85.1 (137/161)	20.0 (1/5)	71.7 (43/60)
ft-gpt-4o	92.4 (133/144)	35.7 (5/14)	82.5 (66/80)
gpt-4o-mini	82.9 (131/158)	41.7 (10/24)	64.3 (36/56)
gpt-4.5	89.9 (161/179)	0.00 (0/1)	72.4 (42/58)

Table 4 shows how model performance varies across different reasoning paths. All models perform best with straight reasoning paths, ranging from 82.9% (gpt-4o-mini) to 92.4% (ft-gpt-4o). During re-planning attempts, fine-tuned gpt-4o maintains relatively high accuracy (82.5%), while other models show noticeable degradation (64.3-72.4%).

The occurrence and handling of cyclic loops reveal characteristic behaviors of each model. Gpt-4.5-preview effectively avoids cyclic patterns with only one occurrence, demonstrating its capability to maintain efficient reasoning. In contrast, fine-tuned gpt-4o shows an increased tendency for cyclic behaviors (14 cases) compared to its base model (5 cases), suggesting that supervised fine-tuning may lead to over-fixation on certain patterns. Gpt-4o-mini exhibits both the highest number of cyclic loops (24 cases) and the lowest re-planning success rate (64.3%), indicating fundamental limitations in its reasoning capabilities that prevent it from either avoiding loops or effectively changing its approach when necessary.

4.3 Detailed Analysis of Tool Usage Patterns

Dominant Reasoning Patterns Figure 5 reveals the most frequent tool usage sequences across models. The predominant pattern follows a systematic verification approach: Using detection tool (2:D) → All the objects are detected (3:AY,4:Y) → Zoom-based verification (6:Z) → Answer (7:T). This sequence demonstrates how models typically employ multiple tools to verify their initial findings before reaching conclusions.

Impact of Fine-tuning on Tool Preferences Fine-tuning led to notable changes in tool usage patterns. A striking example is the disappearance of the "Segment and mask → (direct acceptance) → Answer" sequence, which was the second most frequent pattern in base gpt-4o but absent in fine-tuned versions. Despite using only 10 training examples, fine-tuning significantly reduced the overall usage of segmentation masks, indicating a substantial shift in tool preferences.

5 Discussion

Our analysis using VisTRA revealed two critical insights about Visual Language Models' reasoning capabilities. First, while more capable models like gpt-4.5-preview can effectively utilize direct reasoning paths, strategic re-planning remains valuable for improving performance in models with similar architectural capabilities. This suggests that both inherent model capacity and reasoning strategy contribute to overall VQA performance, with the relative importance of each factor varying by model scale.

Our analysis also revealed how training influences tool usage patterns. Fine-tuning can dramatically reshape a model's approach to tool selection and verification, even with minimal training data. We observed the emergence of a hierarchical tool usage pattern, where certain tools serve as primary diagnostic instruments while others function as fallback options. This suggests that models develop tool-specific confidence levels that guide their verification strategies.

These findings point to several important directions for future research. First, we need more sophisticated frameworks for distinguishing between genuine direct acceptance and internal verification processes, particularly in high-capability models. Second, the relationship between model scale and optimal verification strategy deserves further investigation - there may be scale-dependent patterns that could inform more efficient training approaches. Finally, understanding how models develop tool-specific confidence levels could lead to more effective training strategies for improving tool use in visual reasoning tasks.

6 Conclusion

This work introduces VisTRA, a systematic framework for analyzing VLMs' tool utilization patterns in small object VQA tasks. Through automated reasoning step annotation, we quantitatively demonstrated that direct acceptance of tool outputs correlates with decreased VQA accuracy across all models, while lower-performing VLMs exhibit ineffective re-planning through cyclic verification loops. Our analysis revealed that more capable models like gpt-4.5-preview show reduced accuracy degradation with direct acceptance, and that fine-tuning can significantly reshape tool preferences and verification patterns, even with minimal training data. These findings provide insights into the relationship between model capacity and reasoning strategies in visual tool utilization.

Limitations

VisTRA's current implementation is specifically designed for analyzing behavior within the SKETCHPAD framework on V* bench. While the framework could potentially be adapted to similar VQA tasks and benchmarks, such applications remain untested. Although SKETCHPAD represents a common approach to visual tool-based reasoning, adapting the framework to other contexts may present challenges, particularly in prompt engineering. The automation and optimization of prompting strategies remain important areas for future work. Regarding direct acceptance, our study deliberately focused on tool usage patterns rather than judgment accuracy. While this approach allowed us to examine behavioral tendencies in isolation, it does not account for the relationship between tool output accuracy and model behavior. Given that the impact of direct acceptance likely varies significantly based on tool output accuracy, future work should investigate the interaction between output correctness and model behavior, particularly for improving VLM performance.

Acknowledgments

References

Jean Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. 2022. Flamingo: a Visual Language Model for Few-Shot Learning. In *Advances in Neural Information Processing Systems*, volume 35.

Brais Bosquet, Daniel Cores, Lorenzo Seidenari, Víctor M. Brea, Manuel Mucientes, and Alberto Del Bimbo. 2023. A full data augmentation pipeline for small object detection based on generative adversarial networks. *Pattern Recognition*, 133.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-End Object Detection with Transformers. pages 213–229.

Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. 2023. InstructBLIP: towards general-purpose vision-language models with instruction tuning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Tanmay Gupta and Aniruddha Kembhavi. 2023. Visual Programming: Compositional visual reasoning without training. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2023-June.

Yushi Hu, Weijia Shi, Xingyu Fu, Dan Roth, Mari Ostendorf, Luke Zettlemoyer, Noah A Smith, and Ranjay Krishna. 2024. Visual Sketchpad: Sketching as a Visual Chain of Thought for Multimodal Language Models. In *Proceedings of Advances in Neural Information Processing Systems*.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. 2023. Inner Monologue: Embodied Reasoning through Planning with Language Models. In *Proceedings of Machine Learning Research*, volume 205.

Zile Huang, Chong Zhang, Mingyu Jin, Fangyu Wu, Chengzhi Liu, and Xiaobo Jin. 2025. Better Sampling, Towards Better End-to-End Small Object Detection. pages 319–335.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan Yen Lo, Piotr Dollár, and Ross Girshick. 2023. Segment Anything. In *Proceedings of the IEEE International Conference on Computer Vision*.

Mate Kisantal, Zbigniew Wojna, Jakub Murawski, Jacek Naruniec, and Kyunghyun Cho. 2019. Augmentation for small object detection.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. 2022. BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation. In *Proceedings of Machine Learning Research*, volume 162.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual Instruction Tuning. In *Proceedings of Advances in Neural Information Processing Systems*, volume 36.

Xiao Liu, Tianjie Zhang, Yu Gu, Iat Long Iong, Yifan Xu, Xixuan Song, Shudan Zhang, Hanyu Lai, Xinyi Liu, Hanlin Zhao, Jiadai Sun, Xinyue Yang, Yu Yang, Zehan Qi, Shuntian Yao, Xueqiao Sun, Siyi Cheng, Qinkai Zheng, Hao Yu, Hanchen Zhang, Wenyi Hong, Ming Ding, Lihang Pan, Xiaotao Gu, Aohan Zeng, Zhengxiao Du, Chan Hee Song, Yu Su, Yuxiao Dong, and Jie Tang. 2024. VisualAgentBench: Towards Large Multimodal Models as Visual Foundation Agents.

Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. 2023. Chameleon: Plug-and-Play Compositional Reasoning with Large Language Models.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-Refine: Iterative Refinement with Self-Feedback. In *Proceedings of Advances in Neural Information Processing Systems*.

Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ram Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, Edouard Grave, Yann LeCun, and Thomas Scialom. 2023. Augmented Language Models: a Survey.

Vaskar Nath, Pranav Raja, Claire Yoon, and Sean Hendryx. 2025. ToolComp: A Multi-Tool Reasoning & Process Supervision Benchmark.

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Xuanhe Zhou, Yufei Huang, Chaojun Xiao, Chi Han, Yi Ren Fung, Yusheng Su, Huadong Wang, Cheng Qian, Runchu Tian, Kunlun Zhu, Shihao Liang, Xingyu Shen, Bokai Xu, Zhen Zhang, Yining Ye, Bowen Li, Ziwei Tang, Jing Yi, Yuzhang Zhu, Zhenning Dai, Lan Yan, Xin Cong, Yaxi Lu, Weilin Zhao, Yuxiang Huang, Junxi Yan, Xu Han, Xian Sun, Dahai Li, Jason Phang, Cheng Yang, Tongshuang Wu, Heng Ji, Guoliang Li, Zhiyuan Liu, and Maosong Sun. 2025. Tool Learning with Foundation Models. *ACM Computing Surveys*, 57(4):1–40.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2017. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6).

Jingqing Ruan, Yihong Chen, Bin Zhang, Zhiwei Xu, Tianpeng Bao, Guoqing Du, Shiwei Shi, Hangyu Mao, Ziyue Li, Xingyu Zeng, and Rui Zhao. 2023. TPTU: Large Language Model-based AI Agents for Task Planning and Tool Usage. In *Proceedings of Advances in Neural Information Processing Systems*.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. 2024. Identifying the Risks of LM Agents with an LM-Emulated Sandbox. In *Proceedings of International Conference on Learning Representations*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language Models Can Teach Themselves to Use Tools.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: an autonomous agent with dynamic memory and self-reflection. *Advances in Neural Information Processing Systems*, 36.

Bharat Singh, Mahyar Najibi, and Larry S. Davis. 2018. Sniper: Efficient multi-scale training. In *Advances in Neural Information Processing Systems*, volume 2018-December.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. 2023. ProgPrompt: Generating Situated Robot Task Plans using Large Language Models. In *Proceedings - IEEE International Conference on Robotics and Automation*, volume 2023-May.

Dídac Surís, Sachit Menon, and Carl Vondrick. 2023. ViperGPT: Visual Inference via Python Execution for Reasoning. In *Proceedings of the IEEE International Conference on Computer Vision*.

Ryota Tanaka, Taichi Iki, Kyosuke Nishida, Kuniko Saito, and Jun Suzuki. 2024. InstructDoc: A Dataset for Zero-Shot Generalization of Visual Document Understanding with Instructions. In *Proceedings of AAAI Conference on Artificial Intelligence*.

Rahul Thapa, Kezhen Chen, Ian Covert, Rahul Chalamala, Ben Athiwaratkun, Shuaiwen Leon Song, and James Zou. 2024. Dragonfly: Multi-Resolution Zoom-In Encoding Enhances Vision-Language Models.

Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. 2024. MINT: Evaluating LLMs in Multi-turn Interaction with Tools and Language Feedback. In *Proceedings of International Conference on Learning Representations*.

Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. 2023a. Visual ChatGPT: Talking, Drawing and Editing with Visual Foundation Models.

Penghao Wu and Saining Xie. 2024. V*: Guided Visual Search as a Core Mechanism in Multimodal LLMs. In *Proceedings of 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13084–13094. IEEE.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. 2023b. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation.

Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. 2023. MM-REACT: Prompting ChatGPT for Multimodal Reasoning and Action.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. ReAct: Synergizing Reasoning and Acting in Language Models. In *Proceedings of International Conference on Learning Representations*.

Zhoutong Ye, Mingze Sun, Huan-ang Gao, Chun Yu, and Yuanchun Shi. 2025. MOAT: Evaluating LMMs for Capability Integration and Instruction Grounding.