Scaling Agentic Reinforcement Learning for Tool-Integrated Reasoning in VLMs

Supplementary Material

A. Task and Data Information

Table 4 presents the statistics of the in-distribution (ID) and out-of-distribution (OOD) datasets; the additional details are as follows:

A.1. Information for In-Distribution Datasets

We curate a diverse suite of training tasks integrated into VISTA-Gym, spanning fifteen datasets across eight distinct reasoning domains:

- **Chart Understanding**: FigureQA [26] and ChartQA [43] demand precise quantitative extraction and logical inference over diverse data visualizations, requiring the agent to align visual features (bars, lines) with numerical values.
- **Geometric Reasoning**: Geometry3K [38], GeoQA [7], and UniGeo [8] necessitate grounding symbolic theorem application in diagrammatic parsing to solve multi-step spatial and quantitative problems.
- **Mathematical Reasoning**: MathVision [67] and MathV360 [62] evaluate the interpretation of complex mathematical diagrams and symbolic expressions, requiring the synthesis of visual perception with algebraic deduction.
- **Geospatial Reasoning**: MapQA [6] and InfographicVQA [45] challenge agents to perform rigorous spatial lookups, legend-symbol alignment, and information retrieval across dense graphical layouts.
- **Visual Scientific Reasoning**: ThinkVL [13], VizWiz [4], and ScienceQA [40] encompass a broad spectrum of tasks ranging from accessibility-focused visual description to domain-specific multimodal scientific inquiry.
- **Document Understanding**: DocVQA [44] tests layout-aware information extraction from unstructured documents, including financial reports, forms, and scanned articles.
- **Spatial Reasoning**: CLEVR [25] provides a diagnostic environment for compositional logic, assessing the agent's ability to execute complex reasoning chains regarding attribute recognition, spatial relations, and counting in 3D scenes.
- **General Visual Reasoning (Others)**: A-OKVQA [57] requires the retrieval of external world knowledge and commonsense reasoning to address open-ended visual questions.

A.2. Information for Out-of-Distribution Datasets

To assess generalization, we evaluate on six additional benchmarks featuring distinct reasoning:

Table 4. Dataset statistics in VISTA-Gym.

Dataset	Type	# of training instances	# of testing instances
FigureQA	ID	2000	–
ChartQA	ID	1500	1200
Geometry3k	ID	2000	550
GeoQA	ID	2000	500
UniGeo	ID	2000	750
MathVision	ID	2000	–
MathV360	ID	1500	–
MapQA	ID	1500	1900
InfographicVQA	ID	1000	–
ThinkVL	ID	2000	–
VizWiz	ID	200	–
ScienceQA	ID	1000	–
DocVQA	ID	970	–
CLEVR	ID	1000	–
A-OKVQA	ID	1500	–
TABMWP	OOD	–	960
AI2D	OOD	–	740
PlotQA	OOD	–	1900
Clever-math	OOD	–	900
IconQA	OOD	–	1500
Mathvista	OOD	–	1000
Total	–	**22,170**	**11,900**

- **TABMWP**: TABMWP [41] focuses on table-based math word problems requiring cell selection, row/column aggregation, and arithmetic reasoning grounded in semi-structured text, testing symbolic computation beyond visual chart parsing.
- **AI2D**: AI2D [27] is a diagram-centric science QA benchmark involving annotated figures (labels, arrows, parts); it stresses the joint interpretation of schematic layouts and textual cues via multi-hop reasoning.
- **PlotQA**: PlotQA [46] centers on scientific plots (axes, legends, bars/lines) demanding high-precision value reading and aggregation, serving as a complementary but distributionally distinct test bed to ChartQA.
- **CLEVR-Math**: CLEVR-Math [32] extends compositional reasoning in rendered 3D scenes with arithmetic operations, assessing program-like visual logic coupled with numeric computation.
- **IconQA**: IconQA [39] comprises textbook-style diagram and icon questions (often multiple-choice) that emphasize abstract visual relations and symbolic understanding, diverging from natural image statistics.
- **MathVista**: MathVista [42] is a holistic mathematical reasoning suite spanning real images, charts, and diagrams; it integrates OCR, quantitative reading, geometry, and multi-step deduction for comprehensive evaluation.

B. Toolset Information

In VISTA-Gym, we enable access to 26 tools from four categories detailed as follows:

Perception. The perception layer provides low-level visual signals that supply structural cues for higher-level reasoning, including:

- **Detection**: `GroundingDINO` is an open-set object detection model that unifies visual and textual modalities, allowing for the detection of arbitrary objects described by natural language. It performs open-set, text-conditioned object detection, localizing arbitrary entities referenced by natural-language queries.
- **Segmentation**: `SAM` (Segment Anything Model) is a foundational model for promptable image segmentation that can zero-shot segment any object based on flexible inputs like points, boxes, or text. It offers promptable, zero-shot segmentation, isolating regions given points, boxes, or textual prompts.
- **OCR**: `EasyOCR` is a versatile and ready-to-use optical character recognition tool supporting over 80 languages and various writing scripts for robust text extraction from images. It conducts multilingual optical character recognition, extracting text from images across diverse scripts and layouts.

Chart Understanding. The chart layer specializes in plot- and table-centric inference. Chart understanding tool collections like *ChartMoE* use a Mixture-of-Experts connector to enhance visual–text alignment, supporting both structured data extraction and high-level analytical reasoning for chart question answering, including:

- **Chart To Table.** The *ChartToTable* tool subset converts chart images into structured tabular data suitable for downstream numerical reasoning. `UniChart` and `DePlot` recover underlying data points by detecting axes, legends, and visual marks, while `ChartMoE.to_table` and `ChartMoE.extract_data` leverage the ChartMoE backbone to extract series-wise values and metadata (*e.g.*, categories, units) with improved robustness across chart styles.
- **Chart To SVG.** The subset of *ChartToSVG* tools reconstructs vectorized representations of charts. `OpenCV` provides low-level image processing to segment graphical primitives, and `ChartDet` localizes key chart components (axes, legends, bars, lines, markers); `ChartOCR` recognizes textual elements (titles, axis labels, tick labels, legends) and anchors them to detected regions, enabling SVG-style reconstruction.
- **Chart To SCG.** The subset of *ChartToSCG* tools maps charts into structured scene graphs (SCG). `ChartDet` and `ChartOCR` are combined to instantiate nodes for visual and textual elements and edges for relations such as series membership, axis association, and legend–color bindings, yielding a graph representation amenable to symbolic reasoning.
- **Captioning Color.** The subset of *CaptioningColor* generates natural-language descriptions of charts with explicit grounding in visual encodings. `ChartAssistant` produces concise summaries of chart type, trends, and salient extrema, whereas `ChartVLM` provides more detailed captions that explicitly reference colors, legends, and value ranges.
- **QA Analysis.** The *QAAnalysis* supports chart question answering and intermediate analysis. `ChartMoE.answer` directly predicts answers to chart-related questions, while `ChartMoE.analyze` performs intermediate computations (*e.g.*, differences, ratios, aggregations) over extracted data, and `ChartMoE.describe` generates explanatory textual analyses that connect the visual structure, quantitative reasoning steps, and final answer.

Diagram Formalization. This layer converts visual diagrams into symbolic structures amenable to automated reasoning. *DiagramFormalizer* parses geometric and schematic diagrams into ConsCDL and ImgCDL-style formal languages, enabling symbolic representations for downstream deductive geometric reasoning.

- **CDL.** The *CDL* tools convert geometric diagrams and problem statements into formal constraint languages suitable for symbolic solvers. `image_cdl` maps raw diagram images to ImgCDL-style primitives (points, lines, circles, incidences), whereas `text_cdl` parses textual problem descriptions into ConsCDL-style constraints and goals. `construction_cdl` recovers the underlying construction sequence of a diagram, and `goal_cdl` extracts explicit target predicates (*e.g.*, lengths, angles, congruence relations), enabling theorem-driven reasoning in downstream geometry solvers.
- **Symbolic Parsing.** The *SymbolicParsing* tools build fully symbolic problem representations by combining diagram parsing with formal language construction. `Inter-GPS` parses both the problem text and the associated diagram into a unified formal language and then applies theorem-based symbolic reasoning step by step to solve geometry problems. `DiagramFormalizer` provides complementary parsing endpoints that expose ImgCDL/ConsCDL representations directly, supplying structured inputs for interpretable geometry problem solvers such as Inter-GPS.

Math Solvers. The solver layer performs domain-specific high-level reasoning on top of structured perceptual inputs:

- **Multimodal**. `G-LLaVA` is a domain-specific model that integrates a CLIP-based vision encoder with DeepSeekMath-RL to bridge the gap between visual perception and complex mathematical reasoning. It executes geometry-aware multimodal reasoning by aligning figure interpretation with textual reasoning through specialized instruction tuning.
- **Math**. `MultiMath` is a specialized multimodal model for

geometric problem solving, optimized through geometric cross-modal alignment and instruction tuning to accurately interpret figures and relationships. It bridges visual perception and symbolic mathematics by combining a CLIP-based vision encoder with DeepSeekMath-RL, tackling complex multimodal math problems end-to-end.

C. Baseline Details

We include additional details of baseline tool-integrated or reasoning-augmented VLMs as follows:

- **VTool-R1** [73] integrates multimodal tool invocation directly into the VLM reasoning loop via reinforcement learning. By optimizing for feedback signals during problem-solving, it surpasses standard prompting and supervised fine-tuning baselines in complex visual reasoning tasks.
- **Perception-R1** [83] utilizes RL to learn active perception policies that optimize how VLMs attend to and interpret visual inputs. This approach moves beyond passive image encoding, fostering decision-oriented visual understanding through reward-driven refinement.
- **R1-VL** [85] employs GRPO to enhance step-by-step visual reasoning. By incentivizing effective intermediate reasoning steps rather than solely focusing on outcomes, it aligns visual perception with logical inference throughout the generated trajectory.
- **R1-OneVision** [81] establishes a framework for converting visual inputs into structured, cross-modal formal representations. By leveraging these symbolic intermediates during reinforcement learning, the model achieves generalized reasoning capabilities across heterogeneous task types.
- **LLaVA-OneVision** [1] proposes a unified training paradigm that consolidates diverse visual tasks under a consistent interface. This design enables a single general-purpose VLM to generalize across wide-ranging visual domains without requiring task-specific adaptation.

D. Prompt Templates

We include the prompt template in Figure 8.

E. Additional Related Works

RL for VLM Reasoning. Improving the reasoning capabilities for VLMs has been an active research front. [76] synthesize reasoning chains via distilling reasoning knowledge from teacher models. Inspired by the success of R1-style training [18], several works attempt to leverage RL [22, 31, 34, 37, 60] to improve VLM reasoning capabilities on visual tasks, including general visual understanding and mathematical reasoning.

System Prompt:

You are an advanced AI agent capable of complex reasoning and tool usage. You must strictly adhere to the following protocol for every interaction:
1. ALWAYS call the appropriate tool first;
2. NEVER provide answers without tool results;
3. Call appropriate tools based on the task;
{tool_descriptions}
4. Reasoning Before Action: before selecting a tool, you must analyze the user's request and determine the necessary steps. Output your internal monologue and logic inside `<think>` and `</think>` tags;
5. Tool Execution: If a tool is required, generate the tool call immediately after your reasoning. Use the following standard JSON format wrapped in XML tags: `<tool_call>{"tool": "<tool_name>", "task": "<task_name>"}</tool_call>`
6. Reasoning After Action: Once you receive the output from a tool, you must analyze the results to determine if further actions are needed or if the task is complete. Output this analysis inside `<think>` and `</think>` tags;
7. Final Output: When you have formulated your conclusion, you must wrap your final answer in `<answer>` and `</answer>` tags.

User Prompt:

Please answer the following {task_type} question:
Question: {Question}
Please provide a complete step-by-step solution to this problem. Your reasoning should:
1. Analyze the problem systematically
2. Check if the tool execution and answer are correct
3. If there are errors, explain what went wrong and provide the correct reasoning
4. Provide the final answer
Use natural expressions like 'let me think' or 'hmm' when helpful, but keep it concise. It's encouraged to use self-reflection or verification especially in the verifying tool output in the reasoning process.
Provide your detailed reasoning between `<think>` and `</think>` tags, then give your final answer between `<answer>` and `</answer>` tags.
Output format: {output_format}

Figure 8. Prompt template.

F. Additional Method Details of VISTA-Gym

Interface Details. VISTA-Gym follows the Gymnasium API with core methods `reset` and `step`. On `reset`, the environment samples a task $x \in \mathcal{I}$ (question + image) and

Table 5. Effect of reasoning and tool-use results without RL training. *w/ Tools* refers to directly augmenting VLMs with tools significantly degrades accuracy; *w/ Reasoning* refers to CoT reasoning without tools. *w/ T&R* refers to a different setting compared to TIR, only supplying *tool-selection prior knowledge* and interleaving reasoning with tool execution improve performance.

Baselines (↓)	**In-Distribution**						**Out-of-Distribution**							**All**
Datasets (→)	ChartQA	Geometry3K	GeoQA	UniGeo	MapQA	avg.	TABMWP	AI2D	PlotQA	CLEVR-Math	IconQA	MathVista	avg.	avg.
GPT-5	85.92	94.84	90.91	49.34	60.95	76.39	99.90	86.10	72.90	26.00	87.20	80.20	75.38	75.84
w/ Tools	58.60	74.04	65.00	28.78	31.20	51.52	69.83	65.07	46.75	18.96	60.53	42.60	50.62	51.03
w/ Reasoning	85.68	91.01	86.27	51.25	59.48	74.74	99.81	83.03	84.73	25.50	86.27	80.90	76.71	75.81
w/ T&R	94.00	93.84	91.56	51.69	57.35	77.69	98.92	84.62	79.45	27.79	91.07	57.80	73.28	75.28
InternVL3-8B	77.32	47.09	44.87	33.71	34.70	47.54	95.32	74.54	63.05	28.95	73.27	59.60	65.79	57.49
w/ Tools	25.44	17.06	27.27	26.90	16.22	22.58	58.15	51.74	58.31	18.01	25.86	22.70	39.13	31.61
w/ Reasoning	81.28	47.09	45.07	35.81	31.10	48.07	94.38	76.51	65.85	24.15	75.53	60.70	66.19	57.95
w/ T&R	68.56	35.64	55.90	27.06	26.45	42.72	92.34	50.45	38.70	28.18	80.07	29.10	53.14	48.40

returns the initial observation. At time step t, given the interaction history c_{t-1}, the environment provides the current observation o_t, and the agent produces a thought–action pair (g_t, a_t). The environment executes a_t, transitions according to the POMDP formalization in Section 3, and yields the next observation o_{t+1}, enabling multi-turn tool-integrated interaction.

G. Additional Method Details of VISTA-R1

Image Additional Design. Integrating the InternVL3 [88] family into our reinforcement learning framework requires a bespoke adaptation of the visual processing pipeline, as its composite architecture differs substantially from monolithic models such as Qwen2.5-VL [3]. Instead of performing early tensor conversion, we preserve raw `<image>` placeholders in the prompt and rely on InternVL's native processor to expand them into 256 `<IMG_CONTEXT>` tokens and inject the corresponding visual embeddings during the forward pass. To ensure stability in distributed training, we customize Ray and Fully Sharded Data Parallel (FSDP) to wrap only the language decoder layers, leaving the vision encoder unsharded for memory efficiency, and we adjust attention masks and position IDs in VLLM to accommodate the extended visual token sequence. This minimally invasive interface adaptation preserves InternVL's native visual processing while enabling robust, efficient multimodal GRPO training.

H. Additional Implementation Details

H.1. Detailed Error Types for Error Analysis

We define the detailed error types as follows:

- **E1: Invocation schema violation (wrong function-call structure).** The model produces an invalid tool call that violates the prescribed schema, such as missing required fields, extraneous keys, incorrect nesting, or non-JSON-conformant structures that prevent execution.
- **E2: Invalid argument name (wrong argument key).** The tool call structure is syntactically valid, but one or more argument *names* do not match the tool specification (*e.g.*, using `"x_axis"` instead of `"x_label"`), causing the call to be rejected by schema validation.
- **E3: Invalid argument value (wrong argument format).** Argument names are correct, but the *value types or formats* are invalid, such as providing strings where numbers are required, out-of-range values, or malformed lists that violate the tool's input constraints.
- **E4: Incorrect argument value (wrong argument content).** The tool call is syntactically valid and passes type checks, but the *semantic content* of one or more arguments is wrong (*e.g.*, selecting the wrong region, axis, or series for analysis), leading the tool to operate on an incorrect target.
- **E5: Invalid output from tool execution (wrong answer format).** The tool executes but returns an output that does not conform to the expected format for downstream use, such as missing required fields, malformed JSON, or values that cannot be parsed into the canonical answer representation.
- **E6: Incorrect reasoning from tool execution.** The tool output is valid and informative, but the model fails to map it to the correct final answer, due to faulty logical deductions, misinterpretation of intermediate results, or inconsistent multi-step reasoning.

H.2. Additional Training Setup Details

We configure the training environment with a maximum of three turns per interaction, balancing exploration depth and computational cost. The agent retains full access to the conversation history and tool interaction logs across turns, and up to 24 trajectories are processed concurrently to maximize throughput. Agent-side code execution is implemented in Python 3.10. Training uses 8 NVIDIA GPUs with the FSDP2 (Fully Sharded Data Parallel) strategy for efficient distributed optimization. We set parameters to temperature 0.7. The pipeline integrates a tool-router service for real-time tool execution and feedback, and employs asynchronous rollouts with a batch size of 128 to maintain high GPU utilization. We consider four different backbones for VISTA-R1 with varying sizes, including InternVL3-2B/8B/14B and Qwen2.5-VL-7B, in the main experiment.

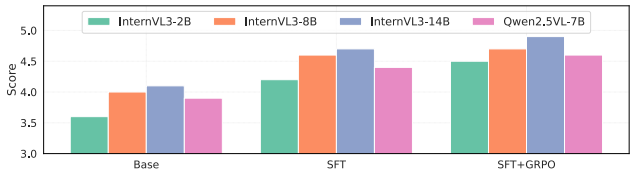

Figure 9. Human study on trajectory quality.

We perform ablation experiments and additional analysis with InternVL3-8B as the VLM. For SFT, we train for one epoch with a batch size of 128, a learning rate of 2×10^{-6}, and a context length of 16,384 tokens. For RL, we use a micro-batch size of 8 per GPU, a mini-batch size of 128, and $G = 8$ rollouts per update. We set the regularization coefficient $\beta = 10^{-3}$, cap the maximum response length at 26,780 tokens, and optimize with a learning rate of 5×10^{-7} for $E = 300$ steps.

I. Additional Experimental Results

I.1. Effect of Tool and Reasoning in Vanilla VLMs

Table 5 summarizes the effect of tools and reasoning on vanilla VLMs for open- and closed-source models. Directly augmenting VLMs with tools significantly degrades accuracy, and intrinsic chain-of-thought alone yields only modest gains on complex VQA. Supplying *tool-selection priors* improves performance; however, gains are strongly task-dependent for commercial VLMs, and smaller open-source VLMs remain particularly challenged. These observations underscore that it is both important and non-trivial to endow VLMs with robust, generalizable tool-use capabilities that adapt across diverse visual reasoning scenarios.

I.2. Human Study for Trajectory Quality

As TIR trajectories are a key component of VISTA-Gym, we evaluate the quality of the generated interleaved tool-use and reasoning steps. Figure 9 reports average human ratings (1–5 scale) over 40 randomly sampled solution trajectory per dataset. We observe that both training stages significantly improve TIR across tasks and model sizes.