This CVPR Findings paper is the Open Access version, provided by the Computer Vision Foundation. Except for this watermark, it is identical to the accepted version; the final published version of the proceedings is available on IEEE Xplore.

Visual Reasoning through Tool-supervised Reinforcement Learning

Qihua Dong[1,2*] Gozde Sahin[2] Pei Wang[2] Zhaowei Cai[2]
Robik Shrestha[2] Hao Yang[2] Davide Modolo[2]
[1]Northeastern University [2]Amazon AGI

Abstract

In this paper, we investigate the problem of how to effectively master tool-use to solve complex visual reasoning tasks for Multimodal Large Language Models. To achieve that, we propose a novel **Tool-s***upervised* **R***einforcement* **L***earning (ToolsRL) framework, with direct tool supervision for more effective tool-use learning. We focus on a series of simple, native, and interpretable visual tools, including zoom-in, rotate, flip, and draw point/line, whose tool supervision is easy to collect. A reinforcement learning curriculum is developed, where the first stage is solely optimized by a set of well motivated tool-specific rewards, and the second stage is trained with the accuracy targeted rewards while allowing calling tools. In this way, tool calling capability is mastered before using tools to complete visual reasoning tasks, avoiding the potential optimization conflict among those heterogeneous tasks. Our experiments have shown that the tool-supervised curriculum training is efficient and ToolsRL can achieve strong tool-use capabilities for complex visual reasoning tasks.*

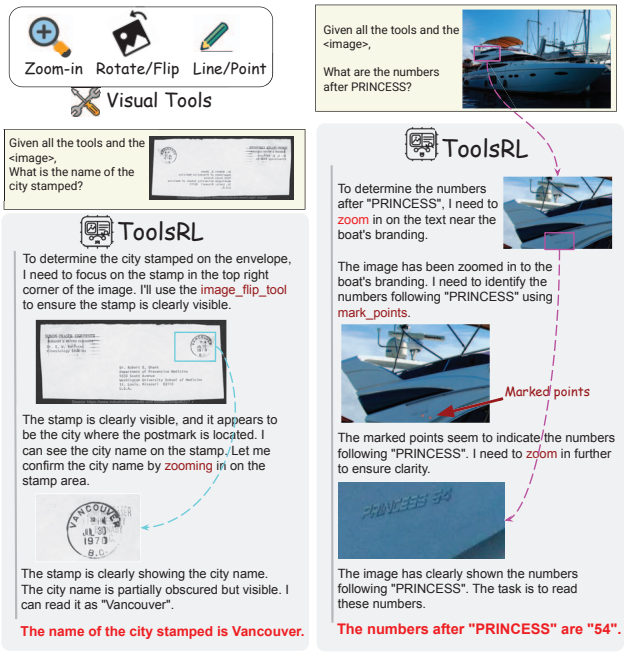

Figure 1. **Visual reasoning with ToolsRL.** Illustrative examples of tool-supervised RL integrating various visual tools into coherent multi-step reasoning chains for different tasks.

1. Introduction

Recent advances in Multimodal Large Language Models (MLLMs) have demonstrated substantial progress in text-only reasoning (thinking-with-text) [1, 3, 10, 17]. However, the capabilities of these models in visual reasoning (thinking-with-images) remain comparatively less explored. Specifically, text-only reasoning proves inadequate for complex visual analysis tasks, such as interpreting rotated text or localizing small objects within cluttered scenes. A promising direction for enhancing MLLM visual reasoning involves integrating visual augmentation tools (e.g., zoom-in, rotation, drawing functions). These tools can generate intermediate visual evidence to support the reasoning process [2, 4, 9, 18, 22, 27, 33, 34]. While proprietary models (e.g., OpenAI-o3 [18]) have shown success, effective and autonomous tool-use capability—specifically determining the optimal invocation strategies (how, when, and why)—remains a significant, unsolved challenge for open-source MLLMs and the broader research community.

Previous research efforts to instill tool-use capabilities primarily leveraged Supervised Fine-Tuning (SFT) on expert tool-use trajectories [9, 22, 25, 27, 33]. However, this approach faces significant scalability challenges [6, 32] due to the substantial manual effort required to construct high-quality expert trajectories (typically generated via prompting stronger reasoning models). Furthermore, recent findings indicate that SFT trajectories require rigorous curation to prevent overfitting and maintain generalization capacity [6, 32]. A more scalable alternative is Reinforcement Learning (RL) methods, such as GRPO [19], which enable models to explore and acquire tool-use strategies without relying on expert SFT trajectory data [2, 8, 28, 34]. However,

*Work done during an internship at Amazon.

current RL-based methods are constrained by conventional reward design. Specifically, their reward functions often rely solely on the final task outcome [28] or provide only generic encouragement for any tool invocation [2, 8, 34], lacking explicit guidance on the optimal timing and execution of tool usage. Consequently, such simple rewards lead to inefficient tool-use training. Models often exhibit infrequent tool invocation (typically fewer than one per episode) and struggle to establish the coherent, multi-step tool-use chains necessary for complex visual reasoning.

To overcome the inherent limitations of both SFT-based and existing RL-based tool-use training pipelines, we introduce **Tool**-**s**upervised **R**einforcement **L**earning (ToolsRL), a novel framework that integrates standard task accuracy rewards with direct *tool supervision* during the RL training process. This tool supervision provides explicit feedback on tool invocation, directly addressing the critical lack of targeted guidance observed in prior RL-based methods. Specifically, we focus on a set of simple, native, and interpretable visual tools, including *zoom-in*, *rotate*, *flip*, *draw line*, and *draw point*, whose ground-truth supervision is easy to collect. For example, the bounding box of the object of interest is utilized as the supervision signal for *zoom-in*, and the underlying rotation degree of the image is for *rotate*. To use these tool supervision signals during RL training, we have designed a suite of novel, well-motivated, and tool-specific reward functions, which encourage right and effective tool invocation.

In general, all rewards are supposed to be optimized jointly during RL training. However, we observed that optimizing both tool-supervised and task accuracy rewards together in a single stage is ineffective, as models frequently defaulted to text-only reasoning. This failure to establish a critical link between tool manipulation and successful task completion motivates our two-stage training curriculum. First, the *Tool Supervision Stage* focuses solely on mastering tool manipulation, with the proposed tool-supervised rewards. Second, the *Task Accuracy Stage* is optimizing the task accuracy rewards only, but allowing calling tools to generate intermediate visual evidences for complex visual reasoning tasks. This curriculum design avoids the potential optimization conflict of the heterogeneous tool and accuracy rewards together, and enables the model to master different skills stage by stage.

In our experiments, ToolsRL has shown strong empirical performance across various tasks demanding visual reasoning capability, e.g. rotated document analysis, high-resolution image understanding, and chart comprehension, as visualized in Figure 1. Our contributions are:

- We propose Tool-supervised Reinforcement Learning (ToolsRL), a simple yet effective two-stage curriculum that enables the model to master tool-use for complex visual reasoning tasks.
- We design well-motivated tool-supervised reward functions for a series of visual tools, which only need a small amount of easily accessible tool annotations, eliminating the need for expensive expert trajectories.
- We provide comprehensive empirical validation across diverse tool-use tasks, demonstrating that ToolsRL yields stable training dynamics, strong accuracy, and enhanced generalization.

2. Related Work

RL for Multimodal Reasoning Without Explicit Tools. Recent works apply reinforcement learning to multimodal language models using accuracy and format rewards [5, 16, 23, 30, 31], often initialized with multimodal chain-of-thought (CoT) or grounded rationale. These methods improve performance on tasks such as math, document, and chart understanding. Representative approaches include Vision-R1 [5], which uses CoT cold start with GRPO; Reason-RFT and R1-ShareVL [23, 31], which stabilize and diversify trajectories under RL; Point-RFT [16], which learns visually grounded rationales before RL; and Look-Back [30], which re-focuses during reasoning without callable tools. While effective for single-round text outputs, these methods cannot leverage intermediate visual manipulations for multi-step visual reasoning.

Visual Tool-Use in Multimodal Language Models To overcome the limitation that arises from lacking explicit tool guidance, a complementary line of work exposes callable visual tools (e.g., zoom-in, draw) to multimodal models [2, 4, 9, 22, 27, 33, 34], with training recipes ranging from training-free prompting [4] to SFT-only [25], SFT-then-RL [22, 27, 33], and RL-only [2, 34] approaches. In **Training-free**, for example, Visual Sketchpad treats drawing as an action interface, improving localization and counting without finetuning, but performance is limited and relies on strong base models [4]. **SFT-only** models are fine-tuned with supervised trajectories of expert tool usage, learning explicit tool invocation patterns but without reinforcement feedback. Simple o3 is a typical example, which interleaves executable operations with a curated tools dataset [25]. However, this type of methods relies on manually curated expert trajectories that are expensive to obtain and task-specific, limiting scalability. In **SFT-then-RL**, models first learn tool-use behavior through supervised fine-tuning and are then refined with reinforcement learning to balance tool effectiveness and final task accuracy. It is a dominant tool-use thread and there are many works on it. *OpenThinkIMG* standardizes tool APIs and mixes answer-quality rewards with intermediate tool-output signals [22]; *Chain-of-Focus* learns adaptive zoom strategies [33]; *Mini-o3* scales to longer multi-turn visual search [9]; and *ViLaSR* uses drawing primitives in a three-stage pipeline with an RL

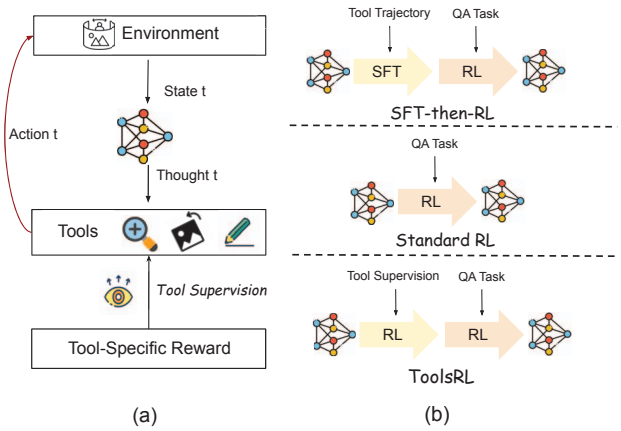

Figure 2. **Overview of Tool-supervised Reinforcement Learning (ToolsRL).** (a) *ToolsRL* includes tool-specific rewards that supervises tool usage. (b) Unlike SFT-then-RL and standard RL, *ToolsRL* injects tool supervision before training on QA tasks.

phase [27]. The same as SFT-only approaches, they all require substantial data effort to construct supervised demonstrations before reinforcement learning can begin. **RL-only** models learn tool-use strategies entirely through reinforcement learning from reward signals, without any supervised demonstrations, promoting scalability and generation. For instance, DeepEyes [34] learns tool policies end-to-end, and RRVF [2] uses render–execute–judge feedback. Simultaneously acquiring fine-grained tool control and optimizing task objectives from sparse rewards remains highly challenging. Our work directly addresses this challenge by incorporating tool supervision during RL.

3. Method

Figure 2 provides an overview of the proposed ToolsRL framework and its two-stage training curriculum.

3.1. Problem Formulation

We cast visual tool use as a finite-horizon sequential decision process. At each turn t, the agent observes the state s_t, consisting of the input question, the current image, and the trajectory so far, and selects an action a_t. The action space includes (1) calling a visual tool with a specific argument, which produces a new image and advances to turn $t{+}1$, or (2) outputting a final answer to terminate the episode. At each turn, the agent may apply tools to any image in the trajectory history. Each turn permits at most one tool call and therefore yields at most one new image.

The goal is to learn a policy $\pi_\theta(a_t \mid s_t)$ that maximizes expected return:

$$\max_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=1}^{T} r(s_t, a_t)\right], \quad a_T = \texttt{<answer>} \quad (1)$$

where $r(s_t, a_t)$ is the per-step reward at turn t, $\tau = (s_1, a_1, \ldots, s_T, a_T)$ is a trajectory, and T is stopping time.

3.2. Tool Supervision for RL

3.2.1. Tool Suite and Tasks

We use three core tools that cover the essential visual operations our framework requires: **Zoom-in** (crop to a bounding box and resize), **Rotate/Flip** (90°/180°/270° rotations or horizontal/vertical flips), and **Draw** (overlay horizontal/vertical lines or points on the image). Together, they span region selection, orientation correction, and coordinate-based annotation. Note that we focus on native tool-calling in this work, and do not consider calling external tools (e.g. standalone segmentation models).

Unlike tool-use SFT, which imitates whole trajectories including all textual reasoning and tool uses, tool supervision is more flexible and scalable. We supervise each task type with ground-truth specific to that task. Given a base visual question-answer dataset, we prepare ground-truth supervision for each of our tasks as follows:

- **Zoom-in tasks:** We use the ground-truth bounding boxes of objects or regions from the question to define the target crop area, and record the corresponding zoom-in operation as the ground-truth tool use. These bounding boxes are obtained from datasets that provide object-level annotations, and serve as supervision for the tool's spatial localization behavior.
- **Rotate/Flip tasks:** We augment images with random rotations and flips, recording the inverse transformation as ground-truth.
- **Draw tasks:** We synthesize chart-style questions and define ground-truth tool use as drawing a horizontal/vertical line to read a point's x/y value, or placing points to support counting or marking (read-value and compare-and-count tasks).

3.2.2. Tool-supervised Reward Design

Building on the ground-truth tool supervision we highlight for each task above, we design rewards to evaluate the tools invoked by the model at each step. We adopt a per-state view: at state s_t with current image I_t, the per-state reward $R_{\text{task}}(s_t, \mathcal{G}^{\text{task}})$ is computed from all tools applied at s_t using the ground-truth set $\mathcal{G}^{\text{task}}$ for that sample.

Zoom-in: Modified F1 reward. For zoom-in, the model must localize visual elements by predicting bounding boxes in tool calls, which lets it crop and resize to target regions in the image. We define a pixel-level modified F1-style overlap metric, ModF1, to evaluate zoom-in tool calls. True positives (TP), false positives (FP), and false negatives (FN) are computed from the intersection-over-union (IoU) between the predicted box mask b and the ground-truth box mask g at the pixel level:

$$\mathrm{ModF1}\big(b, g\big) \;=\; \frac{2\,\mathrm{TP}}{2\,\mathrm{TP} + w_{\mathrm{fp}}\,\mathrm{FP} + w_{\mathrm{fn}}\,\mathrm{FN}}\,, \qquad (2)$$

where g is a ground-truth bounding box and b is the zoom-in box in the tool call. Factors w_{fp} and w_{fn} are the weighting coefficients we apply for FP and FN. Because zoom-in is not a strict grounding task, spurious zooms (FP) are far less harmful than missing the target region (FN). This is why we introduce the coefficients w_{fp} and w_{fn}, emphasizing recall over precision in our reward design. In our final framework, we use w_{fp}=0.1 and w_{fn}=1.0 to reflect this asymmetry.

We compute the per-state reward by matching the predicted zoom-in box b at state s_t to the best ground-truth box in $\mathcal{G}^{\text{zoom-in}}$:

$$R_{\text{zoom-in}}(s_t, \mathcal{G}^{\text{zoom-in}}) \;=\; \max_{g_i \in \mathcal{G}^{\text{zoom-in}}} \mathrm{ModF1}\big(b, g_i\big). \qquad (3)$$

Rotate/Flip: Orientation reward. For rotate/flip tasks, $\mathcal{G}^{\text{rotflip}}$ is the canonical orientation for the original input image; equivalently, it defines the target orientation o^* for I_t. Since we evaluate *only* calls on the current image I_t, we use a binary per-state reward:

$$R_{\text{rotflip}}(s_t, \mathcal{G}^{\text{rotflip}}) \;=\; \mathbb{1}[\mathrm{o}(I_t) = o^*] \in \{0, 1\}, \qquad (4)$$

where $o(I_t)$ is the orientation for the current image, and o^* is the target orientation.

Draw: Unified coordinate-based reward. For tasks requiring precise spatial reasoning, the model draws lines or points at specific coordinates. We use a single margin-based score for both primitives (line and point). Given a predicted primitive p and a ground-truth primitive p^*, we compute a similarity score as:

$$s(p, p^*) \;=\; \max\!\Big(0,\; 1 - \tfrac{d(p,p^*)}{T_{p^*}}\Big), \qquad (5)$$

where $d(\cdot)$ is the primitive-appropriate distance function and $T_{p^*} \in \{T_x, T_y, T_p\}$ is the tolerance for the matched primitive type (x-axis line, y-axis line, or point).

For lines along axis $a \in \{x, y\}$, Let c denote the predicted line coordinate along axis a, and c_a^* the ground truth coordinate. Then,

$$d_{\text{line}} = |c - c_a^*|, \quad T_x = W/4, \quad T_y = H/4, \qquad (6)$$

where W and H are the image width and height, respectively. For points, Let $p = (x_p, y_p)$ and $p^* = (x_p^*, y_p^*)$ denote the predicted and ground-truth points. Then,

$$d_{\text{point}} = \sqrt{(x_p - x_p^*)^2 + (y_p - y_p^*)^2}, \qquad (7)$$

$$T_p = \sqrt{(W/4)^2 + (H/4)^2}. \qquad (8)$$

Intuitively, $s(p, p^*)$ is 1 when the predicted primitive exactly matches the ground truth, and decreases linearly to 0 as the prediction reaches the tolerance threshold.

For the per-state reward, let $\mathcal{C}_t^{\text{draw}}$ be the set of predicted primitives (all required lines and/or points) produced at s_t, and let $\mathcal{G}^{\text{draw}}$ be the corresponding ground-truth primitives that contain line coordinates and/or point locations. We compute a similarity score between predictions and ground truth using Hungarian matching. S_{TP} is defined as the sum of per-primitive similarities $s(p, p^*)$ for the optimal one-to-one matching between $\mathcal{C}_t^{\text{draw}}$ and $\mathcal{G}^{\text{draw}}$. So S_{TP} maximizes the total similarity.

Finally, we define the final F1-style reward for draw tasks jointly as:

$$R_{\text{draw}}(s_t, \mathcal{G}^{\text{draw}}) \;=\; \frac{2\,S_{\text{TP}}}{|\mathcal{C}_t^{\text{draw}}| + |\mathcal{G}^{\text{draw}}|}\,, \qquad (9)$$

which seamlessly unifies lines and points without separate reward formulas.

3.3. The Two-Stage Tool-supervised Curriculum

In our framework, we adopt a two-stage curriculum that decouples the mechanics of tool use from answer prediction. In **Stage 1 (Tool Supervision)**, the model learns to operate the toolbox accurately and consistently using ground-truth derived rewards; in **Stage 2 (Task Accuracy)**, it learns to produce the correct final answer while freely leveraging the tools it has mastered.

3.3.1. Stage 1: Tool-supervision

This stage optimizes task-specific tool-accuracy rewards computed directly from the tool calls. The model is prompted to explicitly identify and manipulate visual elements using the available tools. For example, a zoom-in tool-supervised question might ask:

> **Tool-supervised question.** The sentence is: "What does the label on the bottom right corner of the yellow fabric on the fourth shelf of the cabinet on the left say". First, identify the visual elements (objects or text) referenced in the sentence. Then, use zoom-in to locate each element in the image and report which zoom call finds it (index starts from 1). Example: if "bike" is found on the 3rd image and "person" on the 8th, answer: `<answer>"bike": 3, "person": 8</answer>`

Sample tool-supervised questions for other tasks are provided in the Suppl.

For our final Stage 1 reward, we define two reward components to balance exploration and task-awareness: a *global* tool reward $R_{\text{tool}}^{\text{global}}$ that evaluates the complete tool trace (encouraging broad exploration), and an *answer-conditioned* tool reward $R_{\text{tool}}^{\text{answer}}$ that evaluates only the

tool calls applied to the image referenced in the model's <answer> (enabling awareness of effective tool use).

Global-only rewards encourage exploration but may reward irrelevant steps; while answer-only rewards improve relevance but hinder discovery. Using both ($R_{\text{tool}}^{\text{global}}$ and $R_{\text{tool}}^{\text{answer}}$) balances exploration with task relevance and reduces inefficient tool uses.

Let $R_{\text{task}}(s_t, \mathcal{G}^{\text{task}})$ denote the appropriate per-state reward (e.g., $R_{\text{zoom-in}}$, R_{rotflip}, or R_{draw}) defined above for the tool-specific task, and let t_{answer} denote the state index referenced in the model's <answer> tag. We define

$$R_{\text{tool}}^{\text{global}} = \max_{t \in \{1,\ldots,T\}} R_{\text{task}}(s_t, \mathcal{G}^{\text{task}}), \tag{10}$$

$$R_{\text{tool}}^{\text{answer}} = R_{\text{task}}(s_{t_{\text{answer}}}, \mathcal{G}^{\text{task}}). \tag{11}$$

The final reward for Stage 1 is then defined as:

$$R_{\text{final, stage-1}} = \tfrac{1}{2}\left(R_{\text{tool}}^{\text{global}} + R_{\text{tool}}^{\text{answer}}\right) + R_{\text{format}}, \tag{12}$$

where R_{format} is the format reward as defined in [34].

3.3.2. Stage 2: Task Accuracy

In this stage, the model receives a standard QA prompt, with no tool-specific supervision applied. The model may still call tools (and typically does, at increasing rates as training progresses), but the sole objective is answer accuracy, which we measure using an LLM judge for all datasets except our synthetic chart sets. For read-value and compare-and-count tasks in our synthetic chart data, we use the normalized numerical score s_{norm} (calculating the difference of our answer and ground-truth and normalizing it by the x/y range of the chart or the number of total points).

$$R_{\text{answer}} = \begin{cases} s_{norm}(\text{ans}, \text{ans}^*), & \text{task} \in \text{synth. chart} \\ \mathbb{1}_{\text{LLM-Judge}}[\text{ans} = \text{ans}^*], & \text{else,} \end{cases} \tag{13}$$

where ans is the model's final answer, ans^* is the target answer, and $\mathbb{1}_{\text{LLM-Judge}}$ denotes a binary judgment by the LLM judge, following [34]. The final reward for this stage combines answer correctness with format compliance:

$$R_{\text{final, stage-2}} = R_{\text{answer}} + R_{\text{format}}, \tag{14}$$

where R_{format} rewards adherence to the expected output structure, again following its definition in [34].

3.3.3. Why does curriculum learning matter?

The two-stage curriculum is crucial for effective tool learning. By decoupling tool mastery from answer accuracy, tool-supervision stage (Stage 1) allows the model to focus exclusively on learning *how* to use tools correctly without the confounding pressure of producing correct answers. Once the model has internalized tool usage patterns in Stage 1, it can naturally leverage these learned capabilities in task accuracy stage (Stage 2) to improve answer accuracy. In contrast, training directly on answer accuracy from the start causes the model to prefer text-based reasoning over tool use. Examples of this can be seen in Figure 4. We provide detailed ablation studies comparing our curriculum approach against combined reward training in Table 2.

4. Experiments

4.1. Settings

Training datasets. To train and evaluate tool-use capabilities of ToolsRL under diverse visual reasoning scenarios, we curate a corpus covering document understanding, spatial reasoning, and chart understanding. Examples are provided in the supplementary material. Specifically,

- **Document understanding:** 3k samples are randomly selected from DocVQA [14] and augmented with rotation and flip transformations as a training set.
- **Spatial reasoning:** 6k samples from SealVQA [29] and 8k high-resolution samples from Visual Probe [9] are used to train the model on fine-grained localization and spatial understanding.
- **Chart/table understanding:** 2k samples from ChartQA [12] and 2k samples from ArxivQA [11] are utilized, complemented by our synthetic datasets Read-Value (2k samples) and Compare-and-Count (4k samples) where the model reads the x/y values of a point or count the number of points that satisfy a condition.

All datasets mentioned above are used during both Stage 1 and Stage 2 training, with the exception of ChartQA and ArxivQA, which are omitted from Stage 1 due to lack of ground-truth annotations for effective tool-supervision.

Evaluation datasets. We evaluate our method on benchmarks spanning the same three domains as the training set:

- **Document understanding:** We use DocVQA [14] and InfoVQA [15] and augment their test sets with rotation and flip transformations to form DocVQA-RF and InfoVQA-RF. Rotations (90°, 180°, or 270°) and flips (vertical or horizontal) are sampled uniformly and applied to the image with 0.7 probability.
- **Spatial reasoning:** HR-Bench [24] and V-Star [29] are used for single- and cross-image high-resolution perception evaluation. Visual Probe [9] is used with easy, medium, and hard splits. We also construct InfoVQA-Res [15], by selecting high-resolution images (max edge length > 1024 pixels) and resizing them to $\leq 512 \times 512$ pixels, to evaluate the model's reasoning performance on high-resolution infographics.
- **Chart/table understanding:** Evaluation is conducted on ChartQA [12] test set, CharXiv [26] reasoning split, ChartQA-Pro [13], and TableVQA [7].

Table 1. Comparison with SOTA on document understanding, spatial reasoning, and chart understanding groups. "–" denotes that the method was not evaluated due to the lack of open-sourced models. "*" indicates that the results are evaluated by us using open-sourced weights. All methods use Qwen2.5-VL-7B as the base model.

Method	Tool-Use Training		Document Understanding		Spatial Reasoning											Chart/Table Understanding		
	SFT	RL	DocVQA-RF	InfoVQA-RF	InfoVQA-Res	V-Star			HR-Bench 4K			HR-Bench 8K			VisualProbe	CharXiv	ChartQA-Pro	TableVQA
						V*-S	V*-C	Avg	HR-4K-S	HR-4K-C	Avg	HR-8K-S	HR-8K-C	Avg				
Qwen2.5-VL [1]	-	-	50.2*	53.8*	50.9*	78.2*	73.6*	75.9*	83.8*	56.9*	70.4*	78.8*	51.8*	65.3*	28.4*	41.2*	31.7*	66.2*
Point-RFT [16]	✓	✓	–	–	–	–	–	–	–	–	–	–	36.20	–	–			
ZoomEye [20]	-	-	–	–	–	93.9	85.5	89.7	84.3	55.0	69.7	88.5	50.0	69.3	–	–	–	–
Simple o3 [25]	✓	-	–	–	–	–	–	90.4	–	–	76.2	–	–	–	–	41.8	–	–
Pixel-Reasoner [21]	✓	✓	–	–	–	–	–	86.3	–	–	74.0	–	–	66.9	38.9	–	–	–
Mini-o3 [9]	✓	✓	52.9*	31.3*	58.2*	–	–	88.2	–	–	**77.5**	–	–	73.3	**55.1**	37.3*	32.9*	56.5*
DeepEyes [34]	-	✓	61.3*	59.7*	59.5*	91.3	88.2	89.8	**91.3**	59.0	75.2	86.8	58.5	72.7	41.6	38.5*	37.2*	67.4*
ToolsRL (ours)	-	✓	**77.3**	**61.4**	**71.0**	**95.6**	**89.4**	**92.5**	91.2	**60.6**	75.9	88.1	58.3	73.2	46.5	**43.5**	**38.8**	**70.2**

Table 2. Ablation results for the components of our framework that cover different reward strategies as well as training strategies (with or without curriculum).

Method	R_{answer}	$R_{\text{tool\_cond}}$	$R_{\text{tool}}^{\text{global}}$	$R_{\text{tool}}^{\text{answer}}$	Curric.	Document Understanding		Spatial Reasoning						Chart/Table Understanding		
						DocVQA-RF	InfoVQA-RF	InfoVQA-Res	VStar-S	VStar-C	HR-4K-S	HR-4K-C	VisualProbe	CharXiv	ChartQA-Pro	TableVQA
Qwen2.5-VL-7B	-	-	-	-	-	50.2*	53.8*	50.9*	78.2*	73.6*	83.8*	56.9*	28.4*	41.2*	31.7*	66.2*
Ans. Reward	✓					62.6	60.0	60.2	93.0	89.5	92.2	57.9	41.9	42.0	37.6	**70.6**
Tool-Cond. & Ans. Reward	✓	✓				71.1	59.1	62.5	92.2	80.3	84.3	57.4	44.1	43.0	35.2	68.3
Tool-Sup & Ans. Reward	✓		✓	✓		58.1	60.9	55.7	90.2	89.5	87.8	53.4	41.4	41.6	35.9	70.5
Global Tool-Sup only	✓		✓		✓	60.3	58.4	58.4	95.1	88.4	89.8	56.0	43.4	**44.1**	**40.4**	69.9
Answer Tool-Sup only	✓			✓	✓	65.4	58.7	61.4	92.3	**90.1**	90.1	57.7	39.7	43.3	37.7	68.8
ToolsRL (ours)	✓		✓	✓	✓	**77.3**	**61.4**	**71.0**	**95.6**	89.4	**91.2**	**60.6**	**46.5**	43.5	38.8	70.2

Table 3. Ablation of key design choices in tool reward formulation across task types. Columns list the key components that differ. "Acc." denotes accuracy and "T." denotes the average number of tool calls.

Zoom-in	VisualProbe		Rot./Flip	DocVQA-RF		Draw	ChartQA-Pro	
w_{fp}	Acc.	T.	data mix	Acc.	T.	reward type	Acc.	T.
w_{fp}=1	42.9	2.13	aug + orig.	67.1	6.98	discrete reward	37.9	2.43
w_{fp}=0.1	46.3	3.20	aug only	79.4	4.26	cont. reward	39.1	2.65

Except for the orientation and resizing augmentations applied to DocVQA and InfoVQA, all other evaluation benchmarks are used in their standard settings.

Implementation Details. We initialize our model from Qwen2.5-VL-7B-Instruct [1] and train with Group Relative Policy Optimization (GRPO) [19], sampling 16 trajectories per input. Training is performed for 200 steps per stage with a learning rate 1×10^{-6}, batch size 256, ratio clipping 0.2, and no KL penalty. Each trajectory allows up to 10 tool-use turns. For zoom-in rewards, we utilize IoU threshold to 0.5 and set $w_{\text{fp}} = 0.1$ and $w_{\text{fn}} = 1$. LLM-Judge in equation (13) is obtained using Qwen2.5-VL-72B [1] and prompted for binary decisions with temperature 0.3. Training is conducted on 4 nodes of $8\times$H200 GPUs with FSDP.

Evaluation Metrics. Following [16, 22, 28, 34], we report accuracy for all of our evaluation benchmarks except DocVQA-RF, InfoVQA-RF and InfoVQA-Res. Accuracy is computed using Qwen2.5-VL-72B [1] as an LLM judge to evaluate answer correctness. For DocVQA-RF, InfoVQA-RF and InfoVQA-Res we instead report ANLS score following [14, 15].

4.2. Experimental Results

4.2.1. Main Results

Table 1 presents the performance comparison of our proposed ToolsRL against SOTA across three regimes. Our method consistently achieves SOTA performance across the majority of evaluated benchmarks, demonstrating the efficacy and generalizability for tool-use in complex reasoning tasks. On document understanding, ToolsRL achieves a significant lead: 77.3% on DocVQA-RF and 61.4% on InfoVQA-RF. These scores surpass DeepEyes by substantial margins. On spatial understanding, ToolsRL demonstrates strong overall performance on HR-Bench together with clear gains on V-Star and InfoVQA-Res, marking 4.3 and 12.8 point improvements over Mini-o3 on the latter two benchmarks. On Chart/Table understanding, ToolsRL consistently outperforms all other approaches with notable improvements. These outcomes validate tool-supervised RL as an effective paradigm for multi-domain reasoning. We provide additional qualitative results and analyses (e.g., comparison case study and tool usage comparison) in the supplementary material.

4.2.2. Ablation Studies

Reward Design and Curriculum. To better understand the contributions of each component in ToolsRL, we perform an ablation study covering different reward designs

and training strategies, as summarized in Table 2. First, introducing the answer reward alone substantially improves performance over the base Qwen2.5-VL-7B model on all benchmarks, e.g., DocVQA-RF rises from 50.2% to 62.6% and TableVQA from 66.2% to 70.6%, demonstrating that reward-driven learning helps the model optimize task outcomes even without explicit tool guidance. On top of it, adding a conditional tool reward ($R_{\text{tool\_cond}}$) [34] alongside answer reward yields mixed improvements. It increases DocVQA-RF accuracy to 71.1% but slightly reduces InfoVQA-RF performance. Similarly, applying tool-supervised rewards alone without curriculum leads to inconsistent gains. These observations motivate our two-stage curriculum pipeline. Examining global and answer-conditioned tool supervision individually reveals complementary effects: global tool supervision mainly benefits chart understanding tasks, while answer-conditioned tool supervision improves spatial reasoning and certain document understanding metrics. Combining these rewards within the ToolsRL framework fully leverages their complementary strengths. With curriculum training, ToolsRL achieves the highest accuracy across nearly all benchmarks, demonstrating that both tool supervision and staged training are essential for effective multi-step visual reasoning.

Tool-supervised Reward Component Design. We ablate key design choices in our tool reward formulation (Table 3). **(1) Zoom-in: False Positive Weight.** Reducing the false-positive weight from 1.0 to 0.1 decreases the penalty for incorrect zoom attempts, encouraging exploration of the zoom-in tool. This change increases VisualProbe accuracy (42.9% → 46.3%) and average tool calls (2.13 → 3.20), showing more effective use of zoom-in actions. **(2) Rotate and Flip: Augmented Data Only in Stage 1.** When Stage 1 is trained on a mix of augmented (rotated/flipped) and original images, the model often predicts the original image index as correct, ignoring rotated/flipped images. This occurs because original images are substantially easier to answer correctly, creating a shortcut that undermines tool learning. Restricting Stage 1 to augmented data forces the model to actively detect and correct orientation issues, improving DocVQA-RF accuracy (from 67.1% to 79.4%) while reducing excessive tool calls (from 6.98 to 4.26). **(3) Draw: Continuous v.s. Discrete Reward.** We compare a discrete, threshold-based reward, assigning full credit only when predicted primitives fall within 10 pixels of the ground truth, with a continuous, margin-based reward (Sec. 3.2.2) that provides graded feedback proportional to prediction accuracy. The continuous reward improves optimization stability and facilitates learning of point/line drawing behaviors, resulting in higher ChartQA-Pro accuracy (37.9% → 39.1%) and a modest increase in average tool calls (2.43 → 2.65).

Table 4. **Native tool support and usage across methods.** We show which native visual tools each method supports (✓) and their average tool calls per sample during training. Our ToolsRL approach is the only method that supports the full suite of native tools and achieves significantly higher tool usage (3.4 calls) compared to most prior work (≤1 call).

Method	Zoom-in	Rotate	Flip	Line	Point	Avg tool calls
DeepEyes [34]	✓	–	–	–	–	1.0
Mini-o3 [9]	✓	–	–	–	–	5.5
ReVPT [35]	✓	–	–	–	–	0.6
Pixel Reasoner [21]	✓	–	–	–	–	0.8
VTool-R1 [28]	–	–	–	✓	–	0.3
OpenThinkIMG [22]	✓	–	–	✓	✓	0.7
ToolsRL (Ours)	✓	✓	✓	✓	✓	**3.4**

Table 5. **Tool-type distribution and composite usage.** We report tool usage distributions grouped by benchmark categories, and the ratio of cases with composite tool use (mixing multiple tools).

Benchmark group	Zoom-in	Rotate	Flip	Line	Point	Comp. ratio
Document understanding	31.0%	33.4%	33.2%	1.9%	0.5%	98.9%
Spatial reasoning	89.8%	0.7%	4.4%	0.0%	5.1%	82.1%
Chart understanding	82.8%	0.6%	1.7%	8.7%	6.2%	95.6%

4.3. Results Analysis

4.3.1. Native Tool Support and Usage

Table 4 compares native tool support and average tool calls across existing methods and ToolsRL. All prior approaches either support only a single tool or a small set of tools, and most invoke tools rarely (typically ≤1 calls per sample, except Mini-o3), relying primarily on text-only reasoning. In contrast, ToolsRL is the only model that supports a wide native toolbox (zoom-in, rotate, flip, draw line, and draw point) and also uses these tools substantially more frequently (averaging 3.4 calls per sample during training).

Tool-type Distribution. Table 5 further breaks down tool usage by benchmark category. Although each benchmark does exhibit some tool preference, we do not find overly homogeneous tool usage or overly task-specific tool behaviors in general. The high composite ratios across all categories (82–99%) indicate that ToolsRL learns to combine multiple tools flexibly for complex reasoning.

4.3.2. Comparison with Different Training Settings

As illustrated in Figure 4, we qualitatively compare how different training strategies shape tool behavior. Using **Accuracy Reward Only**, the model often skips tools and directly guesses the answer from the raw image, occasionally issuing a single zoom-in that does not materially change its prediction. Adding a **Tool-Conditioned Reward** to the accuracy reward (following [34]) encourages more frequent tool usage. However, we observe that these tool calls can be noisy or redundant. The agent often outputs an answer before using the tools, indicating reward hacking in the training. In contrast, our **Tool-supervision Curriculum** pro-

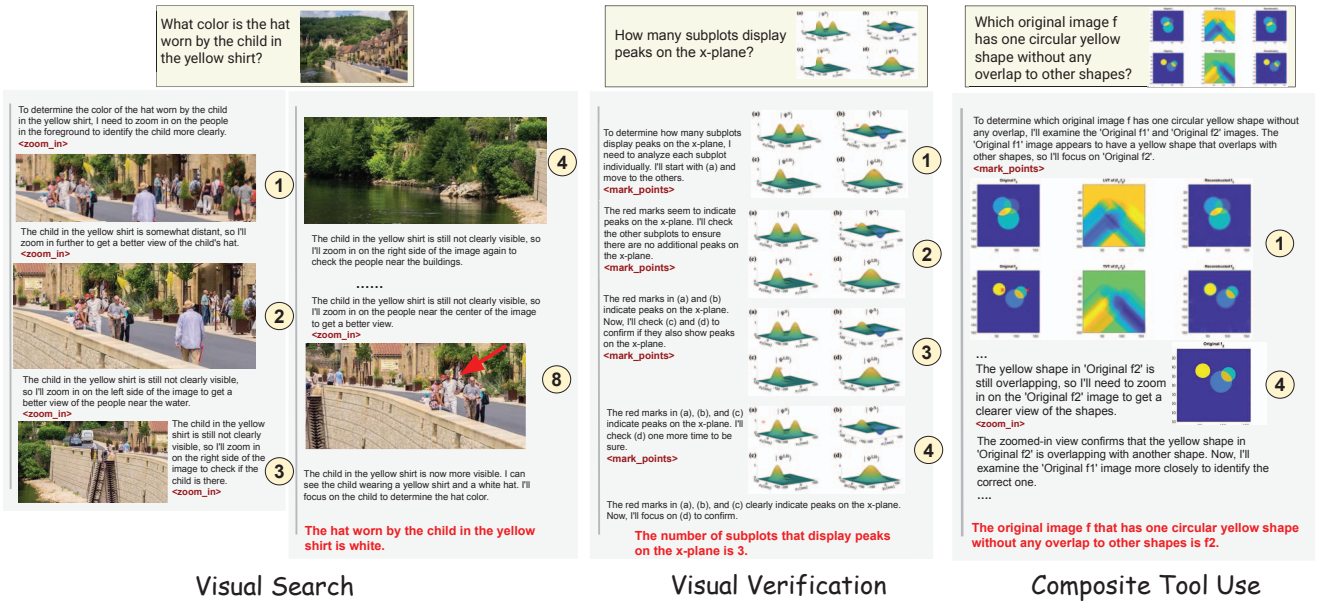

Figure 3. Case studies of ToolsRL. **Left:** Visual search on high-resolution benchmarks, where the agent iteratively zooms in to localize the queried region before answering. Red arrow in the last image indicates the target region. **Middle:** Visual verification on charts, where the agent marks key points to check the presence of peaks on the x-axis. **Right:** Composite tool use, where the agent combines zoom-in and point-drawing operations to disambiguate overlapping shapes and identify the correct answer.

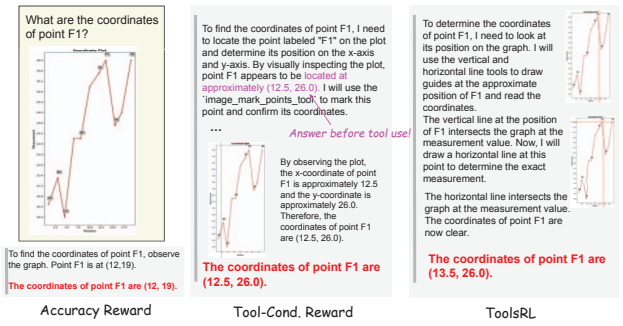

Figure 4. Comparison case study of tool usage across different training settings. The original image and prompt are given in the yellow box, while model answers for different training settings are provided in gray blocks.

duces reasonable tool trajectories that help the model answer the question correctly. Additional qualitative comparisons with baselines are provided in the supplementary material.

4.3.3. Self-Learned Reasoning Patterns

ToolsRL naturally produces long, tool-driven traces that interleave visual search, measurement, and verification using its full toolbox (zoom-in, rotate, flip, line, and point). Figure 3 visualizes three representative behaviors. ToolsRL can perform *multi-step visual search*: it progressively zooms into promising regions, examines local evidence, and refines its hypothesis before answering. On chart domain, it can conduct *visual verification* using pointing to explicitly mark candidate peaks and verify whether they lie on the queried axis. Finally, it also shows *composite tool use*, chaining zoom-in and draw-point tool calls to resolve ambiguous shapes and reason about occlusions. We observe trajectories with up to 8 tool calls that still converge to correct answers, indicating that the agent has learned stable, compositional tool-use policies. Importantly, these behaviors emerge purely from our tool-supervision signals instead of complete tool-use trajectories.

5. Conclusion

We introduce *ToolsRL*, a two-stage tool-supervised RL curriculum that decouples tool mastery from answer optimization. In Stage 1, the model learns tool behaviors from ground-truth-derived, per-tool rewards; in Stage 2, it optimizes answer accuracy with GRPO while freely invoking the learned tools. Across document understanding, spatial reasoning, and chart/table understanding, this curriculum yields more stable training, higher accuracy, and stronger visual tool-use patterns than existing methods without requiring expensive, curated tool-use trajectories. The core ideas—densifying credit assignment via process-level rewards and using a staged curriculum—apply beyond visual tools (e.g., code generation, embodied agents).

References

[1] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025. 1, 6

[2] Yang Chen, Yufan Shen, Wenxuan Huang, Sheng Zhou, Qunshu Lin, Xinyu Cai, Zhi Yu, Jiajun Bu, Botian Shi, and Yu Qiao. Learning only with images: Visual reinforcement learning with reasoning, rendering, and visual feedback. *arXiv preprint arXiv:2507.20766*, 2025. 1, 2, 3

[3] Qihua Dong, Luis Figueroa, Handong Zhao, Kushal Kafle, Jason Kuen, Zhihong Ding, Scott Cohen, and Yun Fu. Cot referring: Improving referring expression tasks with grounded reasoning, 2025. 1

[4] 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. *arXiv preprint arXiv:2406.09403*, 2024. 1, 2

[5] Wenxuan Huang, Bohan Jia, Zijie Zhai, Shaosheng Cao, Zheyu Ye, Fei Zhao, Yao Hu, and Shaohui Lin. Vision-r1: Incentivizing reasoning capability in multimodal large language models, 2025. 2

[6] Hangzhan Jin, Sicheng Lv, Sifan Wu, and Mohammad Hamdaqa. Rl is neither a panacea nor a mirage: Understanding supervised vs. reinforcement learning fine-tuning for LLMs. *arXiv preprint arXiv:2508.16546*, 2025. 1

[7] Yoonsik Kim, Moonbin Yim, and Ka Yeon Song. TableVQA-Bench: A visual question answering benchmark on multiple table domains, 2024. 5

[8] Sunil Kumar, Bowen Zhao, Leo Dirac, and Paulina Varshavskaya. Reinforcing vlms to use tools for detailed visual reasoning under resource constraints. *arXiv preprint arXiv:2506.14821*, 2025. 1, 2

[9] 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. 1, 2, 5, 6, 7

[10] Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. Llava-onevision: Easy visual task transfer, 2024. 1

[11] Lei Li, Yuqi Wang, Runxin Xu, Peiyi Wang, Xiachong Feng, Lingpeng Kong, and Qi Liu. Multimodal arxiv: A dataset for improving scientific comprehension of large vision-language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14369–14387, Bangkok, Thailand, 2024. Association for Computational Linguistics. 5

[12] 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*, pages 2263–2279, Dublin, Ireland, 2022. Association for Computational Linguistics. 5

[13] Ahmed Masry, Mohammed Saidul Islam, Mahir Ahmed, Aayush Bajaj, Firoz Kabir, Aaryaman Kartha, Md Tahmid Rahman Laskar, Mizanur Rahman, Shadikur Rahman, Mehrad Shahmohammadi, Megh Thakkar, Md Rizwan Parvez, Enamul Hoque, and Shafiq Joty. ChartQAPro: A more diverse and challenging benchmark for chart question answering. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 19123–19151, Vienna, Austria, 2025. Association for Computational Linguistics. 5

[14] Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. DocVQA: A dataset for vqa on document images. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2200–2209, 2021. arXiv pre-print arXiv:2007.00398. 5, 6

[15] Minesh Mathew, Viraj Bagal, Rubén Pérez Tito, Dimosthenis Karatzas, Ernest Valveny, and C. V. Jawahar. Infographicvqa. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2582–2591, 2022. 5, 6

[16] Minheng Ni, Zhengyuan Yang, Linjie Li, Chung-Ching Lin, Kevin Lin, Wangmeng Zuo, and Lijuan Wang. Point-rft: Improving multimodal reasoning with visually grounded reinforcement finetuning, 2025. 2, 6

[17] OpenAI. Gpt-4o system card. arXiv preprint arXiv:2410.21276, 2024. 1

[18] OpenAI. Openai o3 system card. https://openai.com/o3, 2024. Accessed: 2024-12-20. 1

[19] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y.K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. 1, 6

[20] 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 (EMNLP)*, pages 6613–6629, Suzhou, China, 2025. Association for Computational Linguistics. 6

[21] Alex Su, Haozhe Wang, Weiming Ren, Fangzhen Lin, and Wenhu Chen. Pixel reasoner: Incentivizing pixel-space reasoning with curiosity-driven reinforcement learning. *arXiv preprint arXiv:2505.15966*, 2025. 6, 7

[22] Zhaochen Su, Linjie Li, Mingyang Song, Yunzhuo Hao, Zhengyuan Yang, Jun Zhang, Guanjie Chen, Jiawei Gu, Juntao Li, Xiaoye Qu, and Yu Cheng. Openthinkimg: Learning to think with images via visual tool reinforcement learning. *arXiv preprint arXiv:2505.08617*, 2025. 1, 2, 6, 7

[23] Huajie Tan, Yuheng Ji, Xiaoshuai Hao, Minglan Lin, Pengwei Wang, Zhongyuan Wang, and Shanghang Zhang. Reason-rft: Reinforcement fine-tuning for visual reasoning, 2025. 2

[24] 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*, pages 7907–7915, 2025. 5

[25] Ye Wang, Qianglong Chen, Zejun Li, Siyuan Wang, Shijie Guo, Zhirui Zhang, and Zhongyu Wei. Simple o3: Towards interleaved vision-language reasoning. *arXiv preprint arXiv:2508.12109*, 2025. 1, 2, 6

[26] Zirui Wang, Mengzhou Xia, Luxi He, Howard Chen, Yitao Liu, Richard Zhu, Kaiqu Liang, Xindi Wu, Haotian Liu, Sadhika Malladi, Alexis Chevalier, Sanjeev Arora, and Danqi Chen. CharXiv: Charting gaps in realistic chart understanding in multimodal llms. *CoRR*, abs/2406.18521, 2024. 5

[27] Junfei Wu, Jian Guan, Kaituo Feng, Qiang Liu, Shu Wu, Liang Wang, Wei Wu, and Tieniu Tan. Reinforcing spatial reasoning in vision-language models with interwoven thinking and visual drawing. *arXiv preprint arXiv:2506.09965*, 2025. 1, 2, 3

[28] Mingyuan Wu, Jingcheng Yang, Jize Jiang, Meitang Li, Kaizhuo Yan, Hanchao Yu, Minjia Zhang, Chengxiang Zhai, and Klara Nahrstedt. Vtool-r1: Vlms learn to think with images via reinforcement learning on multimodal tool use. *arXiv preprint arXiv:2505.19255*, 2025. 1, 2, 6, 7

[29] 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 (CVPR)*, 2024. 5

[30] Shuo Yang, Yuwei Niu, Yuyang Liu, Yang Ye, Bin Lin, and Li Yuan. Look-back: Implicit visual re-focusing in mllm reasoning. *arXiv preprint arXiv:2507.03019*, 2025. 2

[31] Huanjin Yao, Qixiang Yin, Jingyi Zhang, Min Yang, Yibo Wang, Wenhao Wu, Fei Su, Li Shen, Minghui Qiu, Dacheng Tao, and Jiaxing Huang. R1-sharevl: Incentivizing reasoning capability of multimodal large language models via share-grpo, 2025. 2

[32] Wenhao Zhang, Yuexiang Xie, Yuchang Sun, Yanxi Chen, Guoyin Wang, Yaliang Li, Bolin Ding, and Jingren Zhou. On-policy RL meets off-policy experts: Harmonizing supervised fine-tuning and reinforcement learning via dynamic weighting. *arXiv preprint arXiv:2508.11408*, 2025. 1

[33] Xintong Zhang, Zhi Gao, Bofei Zhang, Pengxiang Li, Xiaowen Zhang, Yang Liu, Tao Yuan, Yuwei Wu, Yunde Jia, Song-Chun Zhu, and Qing Li. Chain-of-focus: Adaptive visual search and zooming for multimodal reasoning via rl. *arXiv preprint arXiv:2505.15436*, 2025. 1, 2

[34] 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. *CoRR*, abs/2505.14362, 2025. 1, 2, 3, 5, 6, 7

[35] 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. 7