CANVASAGENT: ENABLING COMPLEX IMAGE CREATION AND EDITING VIA VISUAL TOOL ORCHESTRATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Complex image creation and editing often exceed a single generation or editing model: a request may require synthesizing, localizing, segmenting, editing, compositing, reading text, and enhancing within one workflow. Such requests are complex in three respects: they involve long-horizon, state-dependent execution, heterogeneous tools whose outputs feed subsequent operations, and multiple visual assets, including multiple input images. Existing multimodal tool-use agents mostly target perception, search, or domain-specific editing, or plan over a single image without training, and large-scale supervision for executable, multi-asset creation trajectories remains scarce. We introduce **CanvasCraft**, a large-scale multimodal tool-use dataset, and **CanvasAgent**, an agent that learns to orchestrate heterogeneous visual tools while tracking intermediate visual assets. CanvasCraft contains $140K$ fully annotated executable trajectories and $10K$ RL task specifications stratified by reasoning difficulty, trajectory length, and tool diversity.[1] CanvasAgent is trained with SFT and then GRPO, using a hybrid reward that combines outcome- and process-level signals. On CanvasCraft-Test (250 tasks), CanvasAgent raises the overall reward of its Qwen3-VL-8B base from 0.426 to 0.821, improving both final-image alignment and trajectory quality, and human evaluation shows the same trend.

1 INTRODUCTION

Image creation and editing have advanced rapidly with diffusion models and instruction-guided editors (Rombach et al., 2022; Brooks et al., 2023), yet many practical requests still exceed a single model call. A user may ask an assistant to generate a scene, locate and segment an object, replace only that region, add text, crop the result, and enhance its resolution, all within one workflow.

We refer to such requests as *complex image creation and editing*. They differ from single-step generation or editing in three ways: (i) long-horizon, state-dependent execution, in which later decisions depend on intermediate visual results that the agent must inspect rather than assume; (ii) heterogeneous tools whose outputs (boxes, masks, text, crops, extracted objects) feed subsequent operations; and (iii) multiple coexisting visual assets, including multiple input images, from which the agent must select the correct input for each call. Completing them is therefore a trajectory-learning problem rather than a prompt-to-image task.

Prior work addresses parts of this setting. Agentic MLLMs mainly use tools for perception, search, or reasoning (Zheng et al., 2025; Hong et al., 2025; Zhang et al., 2025b). Image-editing agents either plan over a single image without training (Wang et al., 2024; Gupta et al., 2025) or work with a single editor or a specialized environment (Lin et al., 2025a;b; Zeng et al., 2025). Existing tool-use corpora, such as MIRA-Editing (Zeng et al., 2025) and MM-Traj (Gao et al., 2025), target iterative single-editor editing or general multimodal tasks; neither combines heterogeneous visual tools, intermediate visual observations, and explicit multi-asset state.

To address this gap, we introduce **CanvasCraft**, a large-scale tool-use dataset for complex image creation and editing, and **CanvasAgent**, an MLLM agent trained on it in two stages. CanvasCraft-SFT

[1]The CanvasCraft data will be publicly released upon acceptance.

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

provides fully annotated executable trajectories, from which supervised fine-tuning learns valid tool calls, asset references, and cross-tool dependencies. CanvasCraft-RL provides task specifications with expected tool sets but no reference trajectories, so that GRPO with a hybrid outcome- and process-level reward can explore tool ordering, parameterization, and stopping.

Our contributions are summarized as follows:

- **CanvasCraft**: 140K executable tool-use trajectories (CanvasCraft-SFT), from single-tool operations to multi-tool chains, and 10K RL tasks (CanvasCraft-RL) stratified by reasoning difficulty, trajectory length, and tool diversity.
- **CanvasAgent**: a multimodal agent that learns via SFT and RL to orchestrate heterogeneous visual tools over multiple visual assets, observing each tool output and referring to intermediate assets through explicit identifiers.
- A **hybrid reward** that combines LLM-as-judge scores for alignment, aesthetics, and trajectory validity with rule-based executability checks and efficiency penalties; ablations show that its outcome and process terms are complementary.

2 Related Work

Tool-Augmented Multimodal Agents ReAct (Yao et al., 2023) formalizes the reason–action–observation loop, and early multimodal systems connect language models with visual foundation models or expert modules (Wu et al., 2023; Yang et al., 2023; Shen et al., 2023; Gupta & Kembhavi, 2023; Suris et al., 2023; Lu et al., 2023). Recent agentic MLLMs use cropping, zooming, code execution, and other tools for active perception, search, and reasoning (Zheng et al., 2025; Hong et al., 2025; Zhang et al., 2025b; Wang et al., 2025b; Shen et al., 2025; Zhao et al., 2025; Song et al., 2025; Chng et al., 2025; Zhang et al., 2025c), while LLaVA-Plus (Liu et al., 2024) and T3-Agent (Gao et al., 2025) learn from large tool-use corpora that include generation and editing tools. These works target perception, reasoning, or general assistance rather than producing a final image through stateful, multi-tool workflows.

Image Editing and Creation Workflows Diffusion models and instruction-guided editors provide strong single-step synthesis and editing (Rombach et al., 2022; Brooks et al., 2023; Zhang et al., 2023; Fu et al., 2024; Sheynin et al., 2023; Wu et al., 2025a), but each call executes one instruction and does not manage multi-step tool dependencies or intermediate assets.

Agents for multi-step editing are closer to our setting. Training-free systems such as GenArtist (Wang et al., 2024) and CoSTA*/FaSTA* (Gupta et al., 2025; 2026) plan or search over generation, editing, and perception tools for a single image. Trained agents either work in a specialized retouching environment, such as Lightroom in JarvisArt (Lin et al., 2025a) and JarvisEvo (Lin et al., 2025b), or issue textual edits to a diffusion editor, as in MIRA (Zeng et al., 2025) and ImageEdit-R1 (Zhao et al., 2026b). ComfyGPT (Huang et al., 2025) and ComfyUI-R1 (Xu et al., 2025) learn with SFT and RL to generate ComfyUI workflow graphs rather than act on intermediate visual results, and concurrent work trains SFT+RL agents that combine search or retrieval with generation and editing (Zhao et al., 2026a; Zhang et al., 2026). CanvasAgent instead learns to orchestrate heterogeneous tools over multiple input images and intermediate assets.

Learning Tool-Orchestration Policies Supervised fine-tuning teaches tool-call formats but can overfit to static demonstrations, so reinforcement learning has been used to optimize tool-use policies, from PPO (Schulman et al., 2017) and GRPO (Shao et al., 2024) to recent search, visual-reasoning, and tool-use agents (Jin et al., 2025; Wu et al., 2025b; Su et al., 2025; Zhang et al., 2025a; Deng et al., 2025; Guo et al., 2025; Wang et al., 2025a; Yan et al., 2026). In complex image creation and editing, the reward must assess both the final image and the execution process, which motivates our hybrid reward combining outcome- and process-level signals (Sec. 3.3).

3 Method

This section describes how we construct CanvasCraft (Sec. 3.1) and train CanvasAgent with a two-stage SFT+GRPO framework (Sec. 3.2) using a task-specific hybrid reward (Sec. 3.3); Fig. 1 gives an overview. CanvasAgent operates with 11 specialized visual tools covering generation,

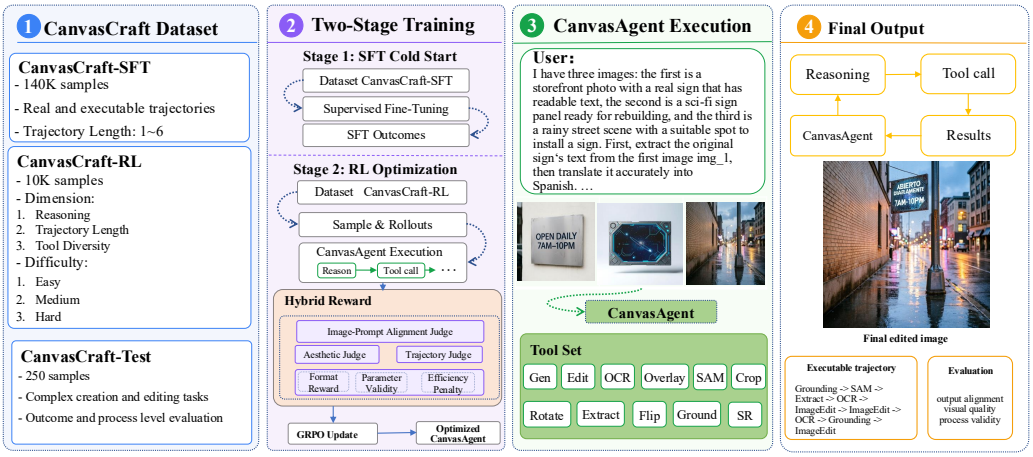

Figure 1: Overview of CanvasCraft and CanvasAgent. (1) CanvasCraft provides SFT trajectories, difficulty-annotated RL tasks, and the 250-task CanvasCraft-Test. (2) CanvasAgent is trained with SFT and then GRPO with a hybrid reward. (3) It interleaves reasoning with calls to 11 visual tools; (4) its final image and trajectory are both evaluated. The example is detailed in Fig. 3.

editing, localization, segmentation, extraction, compositing, geometric transformation, OCR, and super-resolution. During execution, it perceives the current visual state, reasons over intermediate assets, and plans subsequent tool calls, referring to earlier assets through explicit identifiers.

3.1 CanvasCraft Dataset and Construction

To train CanvasAgent for complex image creation and editing, we construct CanvasCraft around two principles: coverage and diversity. Coverage ensures that the data spans essential visual operations, while diversity encourages variation in reasoning difficulty, trajectory length, and tool combination. As shown in Fig. 2, CanvasCraft contains two complementary subsets. CanvasCraft-SFT provides executable multi-step trajectories for supervised tool-use learning, whereas CanvasCraft-RL provides task-level specifications for RL-based policy exploration.

CanvasCraft Tool Set CanvasCraft is built on a unified toolkit of 11 heterogeneous visual tools, including generation, editing, grounding, segmentation, extraction, compositing, cropping, OCR, rotation, flipping, and super-resolution. The full tool specification is provided in Appendix Table 4. Each tool is implemented as a low-level operation with structured JSON-style schema, enabling flexible composition into multi-step visual workflows. Because visual tool use depends on intermediate image states, CanvasCraft emphasizes tool chaining, asset management, and long-horizon workflow execution.

Construction of CanvasCraft-SFT. CanvasCraft-SFT provides executable trajectory supervision for visual tool orchestration. Instead of collecting only input–output image pairs, each sample records how a user instruction is decomposed and completed through real tool execution, including step-level reasoning, tool calls, structured parameters, intermediate assets, and final outputs. This allows the agent to learn valid tool invocation, asset referencing, and cross-tool dependencies in multi-step visual workflows. Formally, each sample is denoted as $d_{\mathrm{SFT}} = (Q, I, \tau, I_{\mathrm{final}})$, where Q is the user instruction, I denotes the optional input image set, τ is the executable tool-use trajectory, and I_{final} is the final output image.

As shown in the top row of Fig. 2, we construct CanvasCraft-SFT through a tool-chain-driven pipeline. We first design tool-chain templates covering atomic visual operations and representative inter-tool dependencies. For each template, we select appropriate input images sampled from PICO-Banana-400K (Qian et al., 2025), and reverse-engineer a natural user instruction from the intended tool-use behavior. Qwen3.5-Plus then carries out the instruction step by step in the tool environment, where every tool call is actually executed, producing reasoning traces, JSON-style tool calls, intermediate assets, and final images. We retain only trajectories that pass quality control, including checks for tool-call parsability, parameter validity, asset-reference consistency, execution success, trajectory correctness, instruction–output consistency, and redundancy.

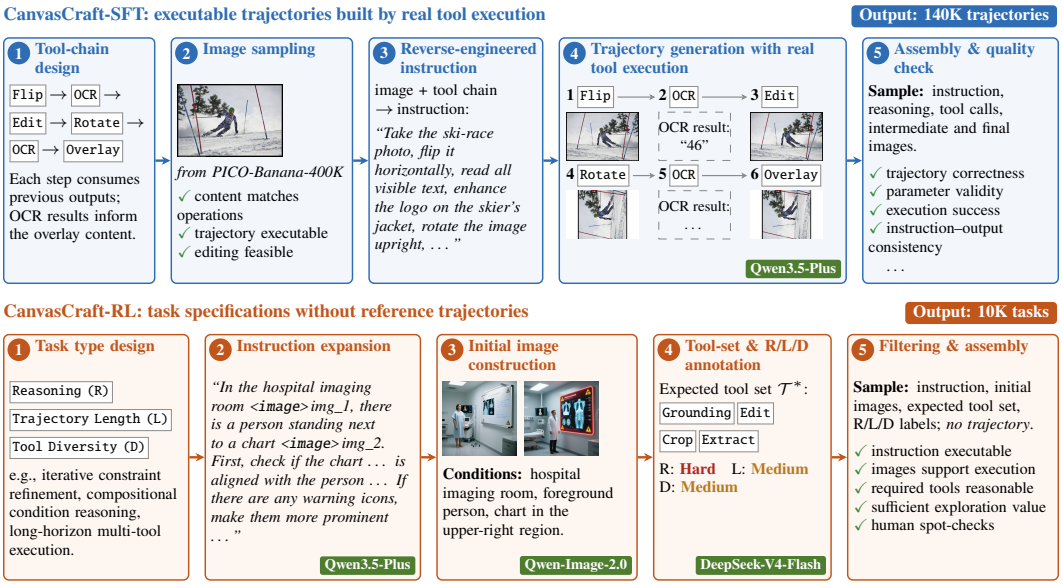

Figure 2: CanvasCraft data construction pipeline. Top: CanvasCraft-SFT executes designed tool chains on sampled images to build executable trajectories. Bottom: CanvasCraft-RL builds difficulty-annotated tasks with expected tool sets but no reference trajectories.

Construction of CanvasCraft-RL. Unlike CanvasCraft-SFT, CanvasCraft-RL provides only the expected tool set as weak supervision, allowing the agent to explore alternative tool ordering, parameterization, verification, and stopping strategies during rollout. Each sample is represented as $d_{\mathrm{RL}} = (Q, \mathcal{I}_0, \mathcal{T})$, where Q is the user instruction, $\mathcal{I}_0$ denotes the optional input image set, and $\mathcal{T}$ is the expected tool set for the task.

As shown in the bottom row of Fig. 2, CanvasCraft-RL is constructed around three task dimensions: **Reasoning (R)**, **Trajectory Length (L)**, and **Tool Diversity (D)**. These dimensions describe the reasoning complexity, expected execution length, and tool heterogeneity of each task, respectively. We first generate diverse task seeds to cover these three dimensions, and then use Qwen3.5-Plus to expand each seed into a natural user instruction. For visually grounded tasks, Qwen-Image-2.0 generates one or more initial images aligned with the instruction. DeepSeek-V4-Flash then annotates each task with an expected tool set $\mathcal{T}^*$ and R/L/D difficulty levels according to the criteria in Appendix Table 5. Finally, DeepSeek-V4-Flash filtering and human spot-checks identify ambiguous, overly simple, or R/L/D-inconsistent instances for removal.

Overall Statistics of CanvasCraft. CanvasCraft contains **140K** SFT trajectories, **10K** RL task specifications, and **CanvasCraft-Test**, a manually curated **250**-sample evaluation benchmark. A representative data instance is shown in Appendix Fig. 4.

CanvasCraft-SFT provides broad coverage of all 11 visual tools and diverse executable supervision through fully annotated tool-use trajectories. The trajectories range from single-tool operations to short dependency chains and high-complexity multi-turn workflows. In particular, 3.8K examples involve longer execution paths, richer tool combinations, and more involved inter-tool dependencies, enabling the model to acquire valid tool invocation formats, basic tool dependencies, and initial long-horizon execution patterns.

CanvasCraft-RL is organized along reasoning difficulty, trajectory length, and tool diversity (R/L/D), with an emphasis on medium and hard levels; the full distribution is reported in Appendix Table 6.

3.2 Two-Stage Training Framework

We train CanvasAgent in two stages (Fig. 1): SFT on CanvasCraft-SFT bootstraps stable tool-use behavior from complete executable trajectories, and GRPO on CanvasCraft-RL optimizes rollout-level multi-tool planning under task-level weak supervision.

SFT for Tool-Use Bootstrapping. In the SFT stage, CanvasAgent is trained on CanvasCraft-SFT with the standard next-token prediction objective. Given complete expert trajectories, the model learns valid reasoning-action formats, JSON-style tool calls, parameter generation, cross-tool dependencies, and references to intermediate visual assets. This stage stabilizes subsequent reinforcement learning, since direct RL over executable visual tools often leads to invalid tool calls, unstable rollouts, and sparse rewards. Nevertheless, SFT is limited to imitating demonstrated trajectories and cannot sufficiently explore alternative tool-use strategies, motivating the RL stage.

RL for Image Planning and Manipulation. After SFT initialization, we further optimize CanvasAgent on CanvasCraft-RL with group relative policy optimization (GRPO) (Shao et al., 2024). For each task, GRPO samples multiple executable rollouts and updates the policy according to their relative rewards, without requiring an additional value model. This stage encourages the agent to explore different task decompositions and tool-use strategies under weak supervision. The optimization signal comes from our task-specific hybrid reward, which evaluates both final visual quality and tool-use process validity.

3.3 Hybrid Reward Design

Diverse image creation and manipulation tasks are difficult to evaluate because they involve multiple intermediate steps, diverse tool choices, and subjective visual quality. To guide RL training, we design a **task-specific hybrid reward** that evaluates both the final output and the execution trajectory:

$$R(\tau) = \underbrace{0.3 \cdot R_{\text{align}}(\tau) + 0.1 \cdot R_{\text{aes}}}_{\text{Outcome Score}}(\tau) + \underbrace{0.2 \cdot R_{\text{traj}}(\tau) + 0.4 \cdot R_{\text{rule}}}_{\text{Process Score}}(\tau). \tag{1}$$

3.3.1 Outcome Score

Outcome scores are computed with expert LLM-as-judge evaluators.

Alignment Score. R_{align} evaluates whether the final image satisfies the user's instruction. Given the image-prompt pair and the final image, it focuses on requested objects, attributes, actions, spatial relations, colors, text, quantities, and edited regions.

Aesthetic Score. R_{aes} evaluates the perceptual quality of the final image. It takes the final image as input and scores composition, color, lighting, clarity, texture naturalness, style consistency, artifacts, and blur. This score is independent of instruction alignment and trajectory quality.

These judges capture semantic fidelity and visual quality, which cannot be fully measured by deterministic rule-based signals alone.

3.3.2 Process Score

Trajectory Score. R_{traj} evaluates whether the trajectory is reasonable for the task. It takes the whole trajectory and the expected tool set as input. The judge focuses on process quality, including whether the agent decomposes the task properly, selects appropriate tools, follows valid tool dependencies, and avoids irrelevant or unnecessary operations. It does not directly judge the final image.

Rule-based Reward. While LLM judges evaluate high-level semantic, perceptual, and reasoning quality, rule-based rewards provide deterministic feedback on low-level executability. We define the rule-based reward as a combination of format reward, action reward, and efficiency penalty:

$$R_{\text{rule}}(\tau) = 0.4 \cdot R_{\text{format}}(\tau) + 0.6 \cdot R_{\text{action}}(\tau) - \lambda_{\text{eff}} P_{\text{eff}}(\tau). \tag{2}$$

The format reward R_{format} verifies whether the model follows the required reasoning-action protocol. It checks the presence of valid `<reason>` and `<tool_call>` blocks, whether tool calls can be parsed as valid JSON, whether each turn contains at most one tool call, whether termination is used properly, and whether the response avoids unsupported extra formatting. This reward enforces syntactic compliance with the agent interface.

While the format reward checks well-formed syntax, the action reward evaluates whether each tool invocation is executable under the current visual state. Given a trajectory τ with N parsed tool calls, we compute

$$R_{\text{action}}(\tau) = \frac{1}{N}\sum_{i=1}^{N} q_i, \tag{3}$$

where the per-action validity score q_i is defined as

$$q_i = 0.25r_i^{\text{name}} + 0.25r_i^{\text{schema}} + 0.20r_i^{\text{ref}} + 0.20r_i^{\text{spec}} + 0.10r_i^{\text{exec}}. \tag{4}$$

The five sub-scores are computed as

$$\begin{aligned} r_i^{\text{name}} &= \mathbb{I}[f_i \in \mathcal{T}], & r_i^{\text{schema}} &= \mathbb{I}[\text{Req}(f_i) \subseteq \text{NonEmptyArgs}(a_i)], \\ r_i^{\text{ref}} &= \frac{1}{|\text{Refs}(a_i)|}\sum_{v\in\text{Refs}(a_i)} \mathbb{I}[v \in \mathcal{A}_{i-1}], & r_i^{\text{spec}} &= \mathbb{I}[\text{SpecCheck}(f_i, a_i, \mathcal{A}_{i-1}) = 1], \\ r_i^{\text{exec}} &= \mathbb{I}[\text{Exec}(a_i) \geq 0]. \end{aligned} \tag{5}$$

Here, f_i is the selected tool, a_i denotes its structured arguments, $\mathcal{T}$ is the set of valid tools, and $\mathcal{A}_{i-1}$ is the set of available visual assets before step i. The term r_i^{name} checks tool-name validity, while r_i^{schema} verifies whether the required arguments of tool f_i are present and non-empty. The reference score r_i^{ref} checks whether all image or asset references in the arguments point to valid assets in the current state. The tool-specific score r_i^{spec} encodes operation-dependent constraints, such as valid bounding boxes for localization and segmentation, valid masks for extraction, valid positions for compositing, and required asset types for object overlay. Finally, r_i^{exec} indicates whether the corresponding tool execution succeeds.

After a successful image-producing tool call, the resulting asset identifier is added to the asset state:

$$\mathcal{A}_i = \mathcal{A}_{i-1} \cup \{\text{OutputID}(f_i, i)\} \quad \text{if } f_i \in \mathcal{T}_{\text{img}} \text{ and } r_i^{\text{exec}} = 1, \tag{6}$$

otherwise $\mathcal{A}_i = \mathcal{A}_{i-1}$, where $\mathcal{T}_{\text{img}}$ denotes the set of image-producing tools. Therefore, R_{action} evaluates whether each tool call is compatible with the evolving visual asset state and executable within the tool environment.

Efficiency Penalty. The efficiency penalty discourages degenerate or unnecessarily costly trajectories:

$$P_{\text{eff}}(\tau) = P_{\text{error}}(\tau) + P_{\text{repeat}}(\tau) + P_{\text{length}}(\tau) + P_{\text{cost}}(\tau) + P_{\text{miss}}(\tau). \tag{7}$$

Here, P_{error} penalizes failed tool executions, P_{repeat} penalizes adjacent repeated or near-repeated tool calls, P_{length} penalizes overly long reasoning segments, P_{cost} discourages excessive tool usage beyond the expected interaction budget, and P_{miss} penalizes missing key expected tools. These terms prevent the agent from improving trajectory scores by blindly invoking more tools.

4 Experiments

4.1 Experimental Setup

Evaluation set. We evaluate all models on CanvasCraft-Test, a 250-task split of CanvasCraft-RL held out from RL training. Each task includes a user instruction, zero to three input images, and an annotated reference tool set (the expected tool set in Sec. 3.1) of one to eight distinct tools. The reference tool set is not shown to the agent and is used only for reward computation.

Compared methods. We compare three groups of methods. First, we evaluate general-purpose MLLMs equipped with the CanvasCraft tool set, including **LLaVA-OneVision-7B**, **Qwen3-VL-8B-Instruct**, and **Qwen3-VL-32B-Instruct**. These models use the same tools but are not trained on CanvasCraft. Second, the image-only models **Qwen-Image-2.0**, **Wan2.7-Image**, and **GPT-Image-2** produce no trajectories, so we report only their alignment and aesthetic scores as outcome-level references. For tasks with input images, each image-only model receives the instruction together with all input images, in order, through its official multi-image editing API; tasks without input images use text-to-image generation. Finally, **CanvasAgent (SFT)** is trained only on CanvasCraft-SFT, and **CanvasAgent (SFT+RL)** is our full model, further trained with GRPO on CanvasCraft-RL.

Table 1: Overall evaluation on CanvasCraft-Test. Higher is better for all scores (Sec. 4.1); Trajectory Length is the average number of tool calls. Dashes mark metrics that do not apply to image-only models, which produce no trajectories. Bold marks the best result per column.

Model	Overall Reward	Alignment Score	Aesthetic Score	Trajectory Score	Rule-based Score	Trajectory Length
LLaVA-OneVision-7B	0.402	0.484	0.598	0.132	0.427	1.354
Qwen3-VL-8B-Instruct	0.426	0.493	0.667	0.092	0.483	1.488
Qwen3-VL-32B-Instruct	0.474	0.428	0.588	0.512	0.461	7.668
Qwen-Image-2.0	–	0.543	0.825	–	–	–
Wan2.7-Image	–	0.605	0.843	–	–	–
GPT-Image-2	–	0.799	**0.895**	–	–	–
CanvasAgent (SFT)	0.557	0.613	0.711	0.576	0.467	1.320
CanvasAgent (SFT+RL)	**0.821**	**0.869**	0.762	**0.849**	**0.785**	5.436

Training setup. All experiments are conducted on 8 NVIDIA A800 GPUs. Six GPUs are used for model training, and the remaining two GPUs host all 11 visual tools locally for executable rollout. The base model for all trained variants is Qwen3-VL-8B-Instruct. The LLM-as-judge model is Qwen3.5-Plus. The same judge provides the judge-based reward components during RL and the automatic evaluation scores; we therefore also report a human evaluation (Sec. 4.5). Appendix D lists the SFT and RL hyperparameters.

Metrics. **Overall Reward** is the hybrid reward of Sec. 3.3; the **Alignment**, **Aesthetic**, and **Trajectory Scores** are its judge-based terms R_{align}, R_{aes}, and R_{traj}; the **Rule-based Score** measures format validity, parameter validity, and efficiency; and **Trajectory Length** is the average number of executed tool calls per trajectory, including repeated calls. Evaluation uses the same hybrid reward as RL training by design, so the scores measure the objective that the agent optimizes.

4.2 Main Results

As shown in Table 1, both training stages improve over the Qwen3-VL-8B-Instruct base (without CanvasCraft training) on CanvasCraft-Test, and the full model, CanvasAgent (SFT+RL), achieves the highest overall reward among all tool-using models. Scale alone does not close the gap: Qwen3-VL-32B-Instruct makes the most tool calls (7.668) but has the lowest alignment score among tool-using models (0.428), and both CanvasAgent variants obtain higher overall rewards.

Compared with Qwen3-VL-8B-Instruct, CanvasAgent (SFT) improves the overall reward from 0.426 to 0.557 and raises the trajectory score from 0.092 to 0.576. This shows that CanvasCraft-SFT teaches structured reasoning–action formats and executable tool-use trajectories. However, CanvasAgent (SFT) still underuses the tool set: it makes only 1.320 tool calls per task on average, fewer than the 3.592 reference tools per task. Here, 3.592 denotes the average number of distinct tools in the annotated reference tool sets across the 250 evaluation tasks. Supervised learning therefore establishes initial tool-use capability but remains limited in active multi-tool planning.

CanvasAgent (SFT+RL) further improves all metrics over CanvasAgent (SFT), raising the overall reward from 0.557 to 0.821 and the alignment score from 0.613 to 0.869. These gains indicate that RL on CanvasCraft-RL with the hybrid reward improves multi-tool planning beyond supervised imitation.

RL also raises the average number of tool calls from 1.320 to 5.436. In Fig. 3, for example, three of the nine calls check or refine the result (OCR, Grounding, and a final ImageEdit). The trajectory judge encourages such steps, since it rewards verification and assigns zero to one-call trajectories. During RL training, the call count rises to 6.43 at step 900 and falls to 5.436 for the final model (step 1200); call counts alone, however, do not show whether tool use becomes more efficient.

The image-only reference models (**Qwen-Image-2.0**, **Wan2.7-Image**, and **GPT-Image-2**) obtain lower alignment scores than CanvasAgent (SFT+RL), suggesting that strong image-only models still struggle with complex image editing tasks that require multi-step tool orchestration. All three, however, obtain higher aesthetic scores than CanvasAgent (0.825–0.895 vs. 0.762). Because the

Table 2: Ablations on CanvasCraft-Test. RL Only skips SFT initialization; the reward ablations remove the outcome terms ($R_{\text{align}}, R_{\text{aes}}$) or the process terms ($R_{\text{traj}}, R_{\text{rule}}$). SFT + RL and Hybrid Reward are the same full model.

Configurations	Overall Reward	Alignment Score	Aesthetic Score	Trajectory Score	Rule-based Score
Training strategy					
SFT Only	0.557	0.613	0.711	0.576	0.467
RL Only	0.604	0.472	0.666	0.673	0.653
SFT + RL	0.821	0.869	0.762	0.849	0.785
Reward design					
w/o outcome reward	0.636	0.320	0.565	0.907	0.755
w/o process reward	0.379	0.421	0.652	0.357	0.290
Hybrid Reward	0.821	0.869	0.762	0.849	0.785

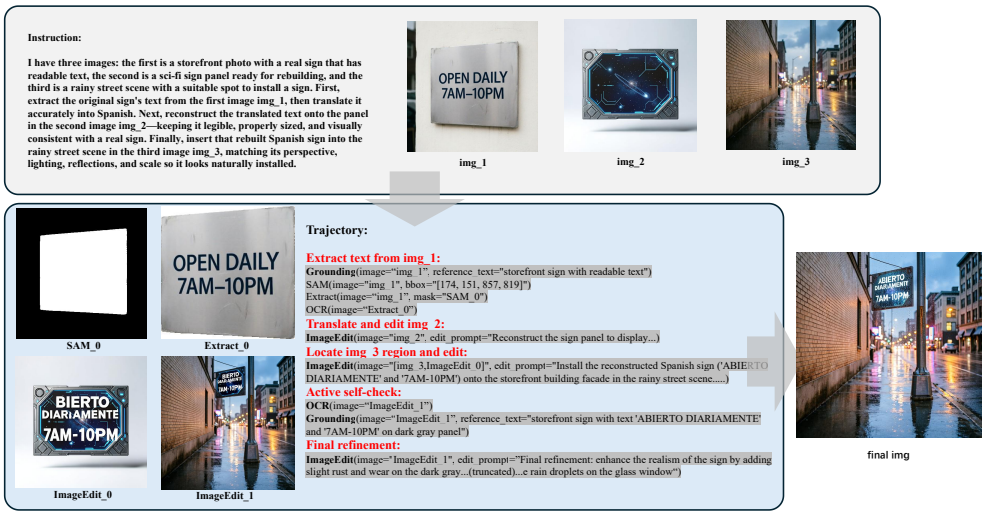

Figure 3: Qualitative case study of CanvasAgent on a complex multi-image editing task. CanvasAgent reads the sign in img_1, rebuilds it in Spanish on img_2, and inserts it into img_3 in nine tool calls, reusing intermediate assets such as SAM_0 and ImageEdit_0.

same judge also provides our training reward, the alignment comparison may favor CanvasAgent, and our human evaluation (Sec. 4.5) does not include these models.

4.3 Ablation Studies

Table 2 ablates the training strategy and reward design. SFT Only establishes executable reasoning–action patterns, reaching an overall reward of 0.557 and a trajectory score of 0.576. RL Only raises the overall reward to 0.604 and the process scores (trajectory 0.673, rule-based 0.653), but lowers alignment and aesthetics to 0.472 and 0.666 relative to SFT Only. This suggests that without SFT initialization, the policy mainly gains reward from the process terms, which carry 60% of the hybrid reward. SFT + RL is the best training strategy on all metrics.

The reward ablations show that outcome and process signals are complementary. Removing the outcome reward preserves a high trajectory score (0.907) but hurts alignment and aesthetics (0.320 and 0.565), while removing the process reward drops the overall reward and trajectory score to 0.379 and 0.357. The full hybrid reward gives the most balanced performance, guiding both final image quality and executable tool-use behavior.

Table 3: Human evaluation on 12 randomly selected CanvasCraft-Test tasks (15 annotators; mean ratings on a 1–5 scale).

Model	Task Alignment	Key Details Alignment	Aesthetic Quality
Qwen3-VL-8B-Instruct	2.68	2.88	2.70
Qwen3-VL-32B-Instruct	2.93	2.91	2.83
CanvasAgent (SFT+RL)	**3.97**	**3.90**	**4.06**

4.4 Case Study

Figure 3 shows a representative multi-image task: the agent must read an English storefront sign (`img_1`), rebuild it in Spanish on a sign panel (`img_2`), and insert the panel into a rainy street scene (`img_3`) while preserving perspective, lighting, scale, and reflections. CanvasAgent solves it with nine tool calls: it grounds, segments, and extracts the sign and reads it with OCR; edits `img_2` to show the translation (`ImageEdit_0`); inserts this panel into `img_3` with one edit that takes `img_3` and `ImageEdit_0` as inputs (`ImageEdit_1`); checks the result with OCR and Grounding; and applies a final refinement. Later calls refer to intermediate assets, such as masks, extracted regions, and edited images, by their identifiers, which lets the agent coordinate several inputs within one trajectory.

After editing, CanvasAgent invokes OCR and grounding to inspect the result before a final refinement. In this example, however, the OCR check did not detect the missing initial capital "A" in the intermediate text "BIERTO". Although the final image reads "ABIERTO", the example does not establish OCR-triggered correction. The case therefore illustrates multi-image, multi-asset tool orchestration, while also exposing a missed text error during verification.

4.5 Human Evaluation

We conduct a human evaluation on 12 randomly selected CanvasCraft-Test tasks to complement the judge-based scores. Fifteen annotators score each output from 1 to 5 on Task Alignment, Key Details Alignment, and Aesthetic Quality, without knowing which model produced it.

As shown in Table 3, CanvasAgent achieves the best scores across all dimensions on these tasks, indicating better instruction satisfaction, key information preservation, and visual quality. These dimensions mirror the judge criteria (instruction alignment, key-detail and text correctness, and aesthetics), so the study checks the automatic results on the same aspects.

5 Conclusion

We present CanvasAgent, a tool-augmented multimodal agent that decomposes complex image creation and editing requests into executable multi-step tool trajectories. To train it, we build CanvasCraft, which provides 140K SFT trajectories and 10K curated RL task specifications spanning diverse editing, composition, perception, and verification behaviors, together with the 250-task CanvasCraft-Test. The two-stage SFT+RL pipeline first grounds the model in executable tool use and then improves both final image quality and trajectory reliability through a hybrid reward. On CanvasCraft-Test, CanvasAgent raises the overall reward of its Qwen3-VL-8B base from 0.426 to 0.821. The ablations show that both training stages and the combination of outcome- and process-level rewards contribute to this gain, and a human evaluation on 12 tasks agrees with the automatic results.

Limitations and future work. CanvasAgent currently uses a fixed set of 11 tools, relies on an external MLLM judge, and requires real tool execution during RL rollout. The same judge is used for the RL reward and for automatic evaluation, and the task instructions are synthetic. The human evaluation covers only 12 tasks, and CanvasAgent still scores below strong image-only models in aesthetics (Table 1), which may reflect the smaller FLUX.2-Klein-4B backend of its generation and editing tools. All results come from a single training run. Future work includes dynamic tool discovery, stronger tool backends, learned or self-evaluation rewards, more efficient rollouts, evaluation with real user requests and independent judges, and video creation and editing.

AI USE STATEMENT

We used Qwen3.5-Plus to assist with instruction generation and the construction of tool-use trajectories for CanvasCraft-SFT, DeepSeek-V4-Flash for expected-tool-set and difficulty annotation and data filtering, and Qwen-Image-2.0 to generate initial images for CanvasCraft-RL. The trajectories were constructed through real tool execution and subjected to the quality-control procedures described in the paper. Generated RL tasks underwent model-based filtering, followed by human spot-checks.

Qwen3.5-Plus also served as the LLM-as-judge for the judge-based reward components during reinforcement learning and for automatic evaluation. We additionally used AI tools to assist with manuscript organization, writing, and LaTeX formatting. The authors take responsibility for the final content of this paper.

REFERENCES

Tim Brooks, Aleksander Holynski, and Alexei A. Efros. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18392–18402, 2023.

Yong Xien Chng, Tao Hu, Wenwen Tong, Xueheng Li, Jiandong Chen, Haojia Yu, Jiefan Lu, Hewei Guo, Hanming Deng, Chengjun Xie, et al. Sensenova-MARS: Empowering multimodal agentic reasoning and search via reinforcement learning. *arXiv preprint arXiv:2512.24330*, 2025.

Mengjie Deng, Guanting Dong, and Zhicheng Dou. ToolScope: An agentic framework for vision-guided and long-horizon tool use. *arXiv preprint arXiv:2510.27363*, 2025.

Tsu-Jui Fu, Wenze Hu, Xianzhi Du, William Yang Wang, Yinfei Yang, and Zhe Gan. Guiding instruction-based image editing via multimodal large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Zhi Gao, Bofei Zhang, Pengxiang Li, Xiaojian Ma, Tao Yuan, Yue Fan, Yuwei Wu, Yunde Jia, Song-Chun Zhu, and Qing Li. Multi-modal agent tuning: Building a VLM-driven agent for efficient tool usage. In *International Conference on Learning Representations (ICLR)*, 2025.

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

Advait Gupta, NandaKiran Velaga, Dang Nguyen, and Tianyi Zhou. CoSTA*: Cost-sensitive toolpath agent for multi-turn image editing. *arXiv preprint arXiv:2503.10613*, 2025.

Advait Gupta, Rishie Raj, Dang Nguyen, and Tianyi Zhou. FaSTA*: Fast-slow toolpath agent with subroutine mining for efficient multi-turn image editing. In *International Conference on Learning Representations (ICLR)*, 2026.

Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. *arXiv preprint arXiv:2211.11559*, 2023.

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

Oucheng Huang, Yuhang Ma, Zeng Zhao, Mingrui Wu, Jiayi Ji, Rongsheng Zhang, Zhipeng Hu, Xiaoshuai Sun, and Rongrong Ji. ComfyGPT: A self-optimizing multi-agent system for comprehensive ComfyUI workflow generation. *arXiv preprint arXiv:2503.17671*, 2025.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training LLMs to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Yunlong Lin, Zixu Lin, Kunjie Lin, Jinbin Bai, Panwang Pan, Chenxin Li, Haoyu Chen, Zhongdao Wang, Xinghao Ding, Wenbo Li, and Shuicheng Yan. Jarvisart: Liberating human artistic creativity via an intelligent photo retouching agent. *arXiv preprint arXiv:2506.17612*, 2025a.

Yunlong Lin, Linqing Wang, Kunjie Lin, Zixu Lin, Kaixiong Gong, Wenbo Li, Bin Lin, Zhenxi Li, Shiyi Zhang, Yuyang Peng, Wenxun Dai, Xinghao Ding, Chunyu Wang, and Qinglin Lu. Jarvisevo: Towards a self-evolving photo editing agent with synergistic editor-evaluator optimization. *arXiv preprint arXiv:2511.23002*, 2025b.

Shilong Liu, Hao Cheng, Haotian Liu, Hao Zhang, Feng Li, Tianhe Ren, Xueyan Zou, Jianwei Yang, Hang Su, Jun Zhu, Lei Zhang, Jianfeng Gao, and Chunyuan Li. LLaVA-Plus: Learning to use tools for creating multimodal agents. In *European Conference on Computer Vision (ECCV)*, 2024.

Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. Chameleon: Plug-and-play compositional reasoning with large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Yusu Qian, Eli Bocek-Rivele, Liangchen Song, Jialing Tong, Yinfei Yang, Jiasen Lu, Wenze Hu, and Zhe Gan. Pico-banana-400k: A large-scale dataset for text-guided image editing, 2025. URL https://arxiv.org/abs/2510.19808.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, 2022.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

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)*, pp. 6613–6629, 2025.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. HuggingGPT: Solving AI tasks with ChatGPT and its friends in Hugging Face. *arXiv preprint arXiv:2303.17580*, 2023.

Shelly Sheynin, Adam Polyak, Uriel Singer, Yuval Kirstain, Amit Zohar, Oron Ashual, Devi Parikh, and Yaniv Taigman. Emu Edit: Precise image editing via recognition and generation tasks. *arXiv preprint arXiv:2311.10089*, 2023.

Qi Song, Honglin Li, Yingchen Yu, Haoyi Zhou, Lin Yang, Song Bai, Qi She, Zilong Huang, and Yunqing Zhao. Codedance: A dynamic tool-integrated MLLM for executable visual reasoning. *arXiv preprint arXiv:2512.17312*, 2025.

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.

Didac Suris, Sachit Menon, and Carl Vondrick. ViperGPT: Visual inference via python execution for reasoning. *arXiv preprint arXiv:2303.08128*, 2023.

Chaoyang Wang, Kaituo Feng, Dongyang Chen, Zhongyu Wang, Zhixun Li, Sicheng Gao, Meng Meng, Xu Zhou, Manyuan Zhang, Yuzhang Shang, et al. Adatooler-v: Adaptive tool-use for images and videos. *arXiv preprint arXiv:2512.16918*, 2025a.

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

Zhenyu Wang, Aoxue Li, Zhenguo Li, and Xihui Liu. GenArtist: Multimodal LLM as an agent for unified image generation and editing. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. Visual ChatGPT: Talking, drawing and editing with visual foundation models. *arXiv preprint arXiv:2303.04671*, 2023.

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng-ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, et al. Qwen-Image technical report. *arXiv preprint arXiv:2508.02324*, 2025a.

Jinming Wu, Zihao Deng, Wei Li, Yiding Liu, Bo You, Bo Li, Zejun Ma, and Ziwei Liu. MMSearch-R1: Incentivizing LMMs to search. *arXiv preprint arXiv:2506.20670*, 2025b.

Zhenran Xu, Yiyu Wang, Xue Yang, Longyue Wang, Weihua Luo, Kaifu Zhang, Baotian Hu, and Min Zhang. ComfyUI-R1: Exploring reasoning models for workflow generation. *arXiv preprint arXiv:2506.09790*, 2025.

Shilin Yan, Jintao Tong, Hongwei Xue, Xiaojun Tang, Yangyang Wang, Kunyu Shi, Guannan Zhang, Ruixuan Li, and Yixiong Zou. Act wisely: Cultivating meta-cognitive tool use in agentic multimodal models. *arXiv preprint arXiv:2604.08545*, 2026.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Ziyun Zeng, Hang Hua, and Jiebo Luo. MIRA: Multimodal iterative reasoning agent for image editing. *arXiv preprint arXiv:2511.21087*, 2025.

Hui Zhang, Zongkai Liu, Liqiang Niu, Juntao Liu, Han Li, Zhen Cao, Wenchao Chen, Chengduo Zhao, and Fandong Meng. WeAgent-MMGenEdit: A full-stack recipe for multimodal agentic image generation and editing. *arXiv preprint arXiv:2609.05171*, 2026.

Kai Zhang, Lingbo Mo, Wenhu Chen, Huan Sun, and Yu Su. MagicBrush: A manually annotated dataset for instruction-guided image editing. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Yabo Zhang, Yihan Zeng, Qingyun Li, Zhen Hu, Kavin Han, and Wangmeng Zuo. Tool-R1: Sample-efficient reinforcement learning for agentic tool use. *arXiv preprint arXiv:2509.12867*, 2025a.

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

Yifan Zhang, Liang Hu, Haofeng Sun, Peiyu Wang, Yichen Wei, Shukang Yin, Jiangbo Pei, Wei Shen, Peng Xia, Yi Peng, et al. Skywork-R1V4: Toward agentic multimodal intelligence through interleaved thinking with images and deepresearch. *arXiv preprint arXiv:2512.02395*, 2025c.

Jiahao Zhao, Xiaomin Yu, Zhongxiang Sun, Fengwei Teng, Chengwei Qin, Xiaobin Hu, Jun Xu, and Shuicheng Yan. ToolArtist: Tool-using unified multimodal models for agentic image generation. *arXiv preprint arXiv:2608.04436*, 2026a.

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

Yiran Zhao, Yaoqi Ye, Xiang Liu, Michael Qizhe Shieh, and Trung Bui. ImageEdit-R1: Boosting multi-agent image editing via reinforcement learning. *arXiv preprint arXiv:2603.08059*, 2026b.

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

A LLM-AS-JUDGE PROMPTS DETAILS

A.1 PROMPT FOR ALIGNMENT SCORE

IMAGE_PROMPT_JUDGE_SYSTEM

```
You are an expert evaluator for visual AI tasks.

You will be given:
1. The user's task prompt
2. The input image, if one was provided
3. The final output image, if one was produced
4. The final text response/trajectory only as auxiliary context

Score ONLY whether the final output satisfies the user's visual request.
Focus on semantic correctness, requested objects/actions, positions, colors,
text, preservation of the input image when editing, and overall visual fidelity.
Do not reward a good-looking process if the final output is wrong.

Be conservative with high scores:
- A score >= 0.9 is allowed ONLY when every major requested constraint is
  satisfied and almost all minor constraints are also correct, with no clearly
  wrong object, identity, text, count, spatial relation, edit target, or major
  omission.
- If the task requires visible text, then any misspelled word, wrong number,
  missing required text, garbled lettering, or clearly incorrect typography
  should usually cap the score at <= 0.3, and large text errors can justify
  scores near 0.0 even if the rest of the image looks good.
- If the final image follows only the broad theme of the prompt but misses one
  or more explicit requested constraints, such as the wrong object, wrong
  color, wrong pose, wrong count, wrong location, wrong style, or incomplete
  edit target, the score should usually be <= 0.4.
- For editing tasks, 0.9+ additionally requires the untouched regions to remain
  natural and the requested region to be edited precisely without damaging
  nearby content.
- If any major requested element is missing, added incorrectly, placed wrongly,
  uses the wrong text, changes the wrong region, or preserves the wrong object,
  the score should usually be <= 0.5.
- If the image is generally plausible but only captures the broad theme while
  missing specific requested constraints, keep the score in the 0.3-0.6 range.
- If the final image looks nice but is semantically wrong, still score it low.
- Do not infer success from the assistant's explanation alone; judge the image
  itself. If uncertain between two bands, choose the lower band.

Use this scale:
1.0 = fully satisfies all important requested constraints
0.9 = correct in all major aspects with only tiny non-critical flaws
0.7-0.8 = clearly strong result, but still missing some secondary details or has
    noticeable minor mistakes
0.4-0.6 = partial success; important requested requirements are missing or wrong
0.1-0.3 = barely related, weak attempt, or major semantic mismatch
0.0 = unrelated, empty, or no meaningful attempt

Output ONLY valid JSON with no markdown fences:
{"score": 0.0}
```

A.2 PROMPT FOR AESTHETIC SCORE

AESTHETIC_JUDGE_SYSTEM

```
You are an expert evaluator of visual aesthetic quality for image generation and image
    editing tasks.

You will be given:
1. The user's task prompt
2. The user's input image, if one was provided
3. The final output image, if one was produced

Your job is to score ONLY the aesthetic quality of the final output image.
Do NOT score whether the image semantically satisfies the user's request.
Do NOT score whether the tool-use trajectory was reasonable.
Those are evaluated separately.

Focus only on visual quality, including:
- composition and balance
- color harmony and lighting consistency
- sharpness and clarity
- naturalness of edges, textures, and object boundaries
- realism or stylistic coherence
- absence of obvious artifacts, distortions, blur, ghosting, broken anatomy,
    pasted-looking objects, jagged masks, or mismatched shadows

Special guidance for editing tasks:
- Reward outputs that preserve the original image naturally while integrating edits
    cleanly.
- Penalize outputs with obvious edit seams, inconsistent perspective, color mismatch,
    unnatural blending, or damaged untouched regions.
- If visible text in the image is malformed, misspelled, broken, inconsistent,
  or obviously pasted in an unnatural way, do not give a high score even if the
  rest of the image is visually appealing.

Scoring rubric (0.0 to 1.0):
1.0   Visually polished and aesthetically strong. Clean composition, coherent style,
    natural blending, sharp details, no visible artifacts, and no visibly broken text.
0.7-0.9  Generally appealing and coherent, with only minor visual flaws and no major
    artifact or broken-text issue.
0.4-0.6  Mixed quality. Some parts look acceptable, but artifacts, weak composition,
    blur, broken text, or inconsistency are clearly noticeable.
0.1-0.3  Poor visual quality. Strong artifacts, unnatural blending, awkward layout,
    visibly bad text rendering, or severe degradation.
0.0   No usable final image, or the output is visually broken.

Important notes:
- If no final image is produced, score 0.0.
- Ignore whether the requested content was correct; judge aesthetics only.
- A semantically correct but ugly image should score low here.
- A visually pleasing but semantically wrong image can still score high here, because
    semantic correctness is evaluated elsewhere.

Output ONLY valid JSON with no markdown fences:
{"score": 0.0}
```

A.3 PROMPT FOR TRAJECTORY SCORE

TRAJECTORY_JUDGE_SYSTEM

```
You are an expert evaluator for a visual tool-use agent.

You will be given the user's task prompt, the expected tool set when available,
the agent's full trajectory, a parsed tool-call summary, and tool error counts.

Score ONLY whether the process is reasonable. Do not judge final image quality.
Reward logical tool selection, valid dependency handling, using outputs from
previous tools correctly, concise but sufficient reasoning, and appropriate
verification before final termination. Penalize irrelevant tools, hallucinated
image IDs, invalid dependencies, blind repetition, and unsupported claims.

Order is not mandatory if the same goal can be achieved another way, but all
necessary tools should appear when the expected tool set is provided.

Hard minimum-score rules:
- If the parsed tool-call summary contains exactly one tool call, output
  {"score": 0.0}. This is the lowest score and has no exceptions.
- A one-tool-call trajectory is considered an invalid process even if the final
  image appears good, because it did not demonstrate verification, reassessment,
  refinement, or meaningful multi-step tool use.
- Do not give partial process credit for a trajectory that only calls
  ImageGeneration once, ImageEdit once, or any other single tool once and then
  terminates.

Be conservative with high scores:
- A score >= 0.9 is allowed ONLY when the trajectory covers the key expected
  tools when applicable, respects dependencies, avoids obvious redundancy, and
  does not skip critical steps.
- A score >= 0.9 additionally requires that the agent responds sensibly to tool
  observations and errors, uses currently valid image IDs, and terminates only
  after there is strong evidence the task is complete.
- If the prompt explicitly asks for verification, reassessment, refinement, or
  iterative correction, then stopping after a single weak attempt without a real
  follow-up check should usually cap the score at <= 0.5.
- If expected tools are provided and multiple key tools are missing, the score
  should usually be <= 0.5 even if the visible process looks superficially neat.
- If the agent uses far fewer tools than the task appears to require, treat that
  as a substantial process defect rather than a small inefficiency.
- If the agent replaces several required deterministic tools with one vague or
  shortcut call, or declares success before all explicit prompt requirements are
  credibly addressed, the score should usually be <= 0.4.
- If the agent hallucinates success after a tool error, ignores a failed tool
  call, or claims completion without enough supporting intermediate evidence,
  the score should usually be <= 0.3.
- Repeated or unnecessary calls, unsupported claims, or weak verification should
  cap the score below the top band.
- If uncertain between two score bands, choose the lower one.

Use this scale:
1.0 = complete, dependency-consistent, efficient trajectory with clear justification
0.9 = all key steps are present and valid, with at most tiny non-critical inefficiency
0.7-0.8 = mostly reasonable, but has some unnecessary calls or small process gaps
0.4-0.6 = partially reasonable, but missing important tools/steps or demonstrating clear
    process defects
0.1-0.3 = largely flawed, strongly incomplete, or tool usage is mostly incoherent
0.0 = no meaningful process or unusable tool trajectory

Output ONLY valid JSON with no markdown fences:
{"score": 0.0}
```

B CanvasCraft Dataset Details

B.1 Tool Specification and Data Example

Table 4: Unified visual tool set used in CanvasCraft and CanvasAgent. Each tool exposes structured inputs and outputs and is backed by a concrete visual model or script.

Tool	Function	Key Inputs	Output	Backend
`Generation`	Text-to-image synthesis	`prompt`	Generated image	FLUX.2-Klein-4B
`Edit`	Instruction-based image editing	`images`, `edit prompt`, optional `mask`	Edited image	FLUX.2-Klein-4B
`Grounding`	Object localization	`image`, `reference text`	Bounding box	Grounding-DINO
`SAM`	Mask generation	`image`, `bounding box`	Segmentation mask	SAM
`Extract`	Object extraction	`image`, `SAM mask`	Extracted object	Python script
`Overlay`	Object/text compositing	`base image`, `overlay type`, `content`, `position`	Composited image	Python script
`Crop`	Region cropping	`image`, `bounding box`	Cropped image	Python script
`OCR`	Text recognition	`image`	Recognized text	Paddle-OCR
`Rotate`	Orientation correction	`image`, `angle`	Rotated image	Python script
`Flip`	Horizontal mirroring	`image`	Flipped image	Python script
`SR`	Super-resolution	`image`	4× enhanced image	Real-ESRGAN

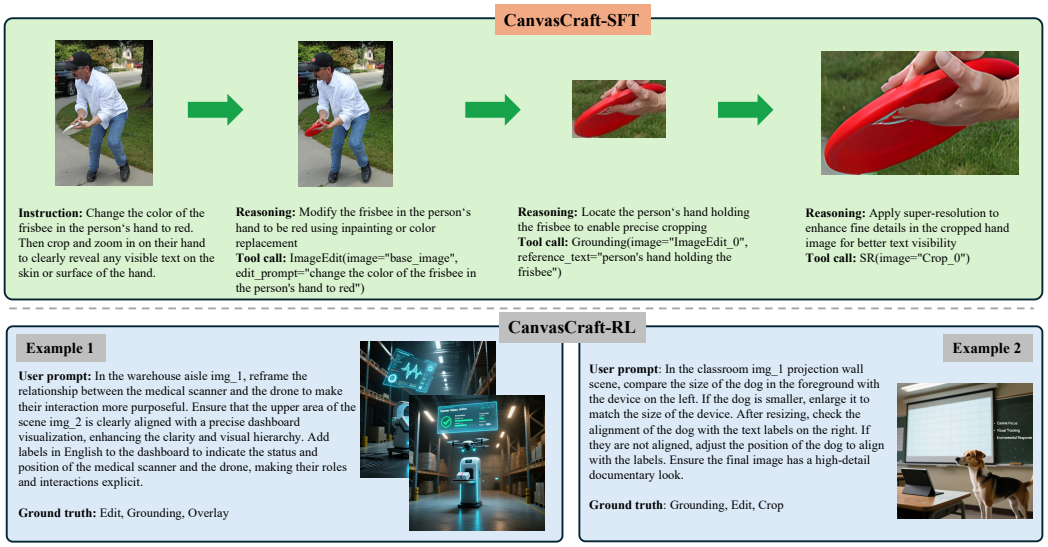

Figure 4: CanvasCraft data example. CanvasCraft-SFT provides the complete execution chain, including step-level reasoning and tool calls. CanvasCraft-RL provides task specifications without complete trajectories, requiring agents to explore tool ordering, parameterization, and intermediate operations during rollout. In the CanvasCraft-RL examples, "Ground truth" lists the expected tool set, not a reference trajectory.

B.2 Task Difficulty Definitions and Distribution

Table 5: Difficulty definitions for the 10K CanvasCraft-RL tasks across Reasoning (R), Trajectory Length (L), and Tool Diversity (D).

Dimension	Easy	Medium	Hard
Reasoning (R)	Single execution path with weak conditional reasoning or minimal state awareness	Multi-condition composition with local path selection based on intermediate results	Nested conditions, strong state dependency, and significantly branching execution paths
Trajectory Length (L)	1–4 execution steps with short tool chains	5–8 steps with explicit task decomposition or serial tool interaction	9+ steps with long-horizon execution and multi-stage coordination
Tool Diversity (D)	1–2 distinct tools with simple composition	3–4 distinct tools with stable collaborative patterns	5+ distinct tools requiring heterogeneous tool orchestration and dynamic switching

Table 6: Difficulty distribution of the 10K CanvasCraft-RL tasks across the R/L/D dimensions. Each task has one level per dimension, so each row sums to 10K.

Dimension	Easy	Medium	Hard
Reasoning (R)	761	1,807	7,432
Trajectory Length (L)	476	4,687	4,837
Tool Diversity (D)	2,317	5,631	2,052

C CANVASCRAFT-SFT DISTRIBUTION

Table 7: Distribution of tool-chain types in CanvasCraft-SFT. The "Multi-tool Hard" category is further decomposed in Table 8.

Tool-chain Type	Count
ImageEdit	19,378
ImageGeneration	18,616
OCR	17,876
Grounding	13,480
Grounding+Crop	13,393
Grounding+SAM	13,393
Grounding+SAM+Extract	13,393
Overlay	10,000
Flip	8,834
Rotate	8,627
SR	2,000
Multi-tool Hard	3,808
Total	**142,798**

Table 8: Breakdown of the Multi-tool Hard subset in CanvasCraft-SFT.

Tool-chain Type	Count	Tool-chain Type	Count
Edit+Grounding+Edit+SR	180	Grounding+Edit+OCR+Edit+Overlay	89
Gen+Edit+OCR+SR	96	SR+Edit+Grounding+OCR+Edit+Overlay	89
Edit+OCR+Grounding+SAM+Grounding	96	Flip+OCR+Rotate+Edit+SR	88
OCR+Grounding+Edit+Grounding	96	Grounding+OCR+Grounding+Crop+Rotate+SR	88
Grounding+Flip+OCR+Edit	94	Flip+Grounding+Rotate+Edit	88
Grounding+Rotate+Grounding+OCR+Edit+Rotate	94	SR+Edit+Flip+Grounding	88
Gen+Grounding+OCR+Grounding+Overlay	92	SR+Grounding+SAM+Extract+Overlay	88
OCR+Flip+Rotate+SR	92	OCR+Rotate+OCR+Edit	88
Edit+Grounding+OCR+Crop	92	Flip+OCR+Edit+Rotate+OCR+Overlay	86
SR+OCR+Edit+Grounding+SAM	92	OCR+Grounding+OCR+Crop	86
Gen+Edit+OCR+Flip	92	Gen+Grounding+OCR+Crop+Rotate	86
SR+OCR+Flip+Grounding+Crop	92	Grounding+Flip+Rotate+OCR	86
Edit+Grounding+Crop+OCR+SR	90	Edit+OCR+Rotate+Overlay	84
Edit+OCR+Grounding+SAM+Overlay	90	Edit+OCR+Edit+Flip+OCR+SR	84
Grounding+Flip+Edit+Flip+Rotate	90	Gen+OCR+Grounding+SR	84
Rotate+Flip+Grounding+Crop+OCR	90	Flip+Grounding+OCR+SR	82
OCR+Grounding+Crop+Flip+Grounding+SR	90	OCR+Grounding+Crop+Edit+Overlay	81
Gen+Grounding+Edit+OCR	90	OCR+Edit+Rotate+SR	80
Rotate+Flip+Grounding+SR	90	Grounding+Rotate+Flip+Overlay	79
		Grounding+Flip+Edit+Overlay	78
		Grounding+Flip+Grounding+Overlay	72
		SR+Grounding+Flip+Overlay	71
		Edit+Grounding+Edit+Overlay	69
		Edit+Grounding+Edit+SAM+Extract+Overlay	21
		Grounding+Flip+Grounding+SAM+Extract+Overlay	14
		SR+Grounding+Flip+SAM+Extract+Overlay	11
		Grounding+Flip+Edit+SAM+Extract+Overlay	10
		Total	**3,808**

D TRAINING DETAILS

Table 9 lists the main hyperparameters for SFT and RL.

Table 9: Training hyperparameters.

SFT		**RL (GRPO)**	
Trained parameters	LLM (vision encoder frozen)	Initialization	SFT checkpoint
Epochs	5	Rollouts per task	8
Learning rate	1×10^{-5} (cosine)	Tasks per step	6
Warmup ratio	0.1	Learning rate	1×10^{-6}
Batch size per GPU	2	KL loss coefficient	0.001
Gradient accumulation	8	Max assistant turns	12
Max sequence length	16,384	Max response length	12,288
Precision	bf16	Epochs	1

E BROADER IMPACTS

CanvasAgent can improve controllable visual creation by decomposing complex editing instructions into interpretable tool-use trajectories, reducing manual effort in design, prototyping, and image editing workflows. However, stronger generation and editing capabilities may also be misused for deceptive or harmful visual content. We therefore emphasize responsible release, dataset filtering, and clear documentation of intended uses and limitations.