CanvasAgent: Enabling Complex Image Creation and Editing via Visual Tool Orchestration

Hairui Zhu[1] Yiying Yang[1] Tengjin Weng[1] Ziyu Lu[1]
Xiao Yao[1] Xiaoyang Ye[1] Lin Ma Wenhao Jiang[1*]

[1]Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ)
Shenzhen, China

Abstract

Complex image creation and editing often require more than a single generation or editing model. A user request may involve synthesizing images, localizing objects, segmenting regions, editing selected content, compositing intermediate assets, reading text, and enhancing the final result. Such tasks shift multimodal agents from perception-augmented reasoning to manipulation-centered visual creation, where tools must actively transform visual states rather than merely inspect them. However, existing multimodal tool-use agents are mostly optimized for perception, search, or domain-specific editing, and lack large-scale supervision for executable image-creation trajectories. In this paper, we introduce **CanvasCraft**, a large-scale multimodal tool-use dataset for complex image creation and editing, and **CanvasAgent**, a tool-augmented multimodal agent that learns to orchestrate heterogeneous visual tools through multi-turn interaction. CanvasCraft contains $140K$ fully annotated executable trajectories and $10K$ RL task specifications. CanvasAgent is first trained with SFT to learn executable reasoning-action trajectories, and is then optimized with GRPO using a hybrid reward that combines outcome- and process-level signals. During rollout, CanvasAgent inspects intermediate results, tracks visual assets, and adapts tool decisions to the evolving visual state. Experiments evaluate both final image quality and trajectory behavior, demonstrating the effectiveness of **CanvasAgent** and the proposed dataset for complex multi-tool image creation workflows.

1 Introduction

Image creation and editing have advanced rapidly with diffusion models, instruction-guided editors, and multimodal large language models (MLLMs) [13, 1, 14]. 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 or another object, crop the result, and enhance its resolution. Such requests require generation, localization, segmentation, editing, compositing, OCR, geometric transformation, and enhancement to be coordinated within one coherent workflow.

These workflows differ from standard image generation or single-step editing in three key ways. First, they are long-horizon: later operations depend on visual artifacts produced by earlier tools. Second, they are visually grounded: after each tool call, the agent must inspect the intermediate output instead of assuming success. Third, they are stateful: multiple images, masks, crops, extracted objects, and edited variants may coexist, and the agent must select the correct asset for each subsequent operation.

*Corresponding author.

Code and datasets are available at: `https://github.com/GML-FMGroup/CanvasAgent`
`https://huggingface.co/datasets/GML-FMGroup/CanvasCraftSFT`
`https://huggingface.co/datasets/GML-FMGroup/CanvasCraftRL`.

These properties make complex image creation and editing a trajectory-learning problem rather than a simple prompt-to-image or instruction-to-image task.

Existing multimodal tool-use research addresses parts of this problem but not the full setting. Early tool-augmented systems connect language models with visual foundation models or external experts for multi-step reasoning and editing [26, 29, 18, 6, 22]. Recent agentic MLLMs improve active visual perception, search, and executable reasoning [36, 7, 33, 24, 35], but their tasks mainly target understanding, search, or reasoning. Image-editing systems and datasets, including instruction-guided editing and photo-retouching agents [1, 31, 4, 19, 25, 9, 10], are closer to our target domain, yet typically focus on single-model editing or retouching in specialized environments. They do not provide large-scale executable trajectories that combine heterogeneous tools, intermediate visual observations, and explicit multi-asset state.

To address this gap, we introduce **CanvasCraft**, a large-scale multimodal tool-use dataset for complex image creation and editing workflows. CanvasCraft contains two complementary subsets. **CanvasCraft-SFT** provides fully annotated execution trajectories with user instructions, optional input images, step-level reasoning, tool calls, parameters, outputs, intermediate visual artifacts, and final images. **CanvasCraft-RL** provides task specifications with expected tool sets, enabling reinforcement learning to explore tool ordering, parameterization, recovery, and stopping strategies without imitating a fixed trajectory.

We instantiate **CanvasAgent**, a tool-augmented MLLM trained in two stages. Supervised fine-tuning on CanvasCraft-SFT provides an initialization for valid tool invocation and cross-tool dependencies. Reinforcement learning on CanvasCraft-RL then refines the policy with GRPO and a hybrid trajectory-level reward combining final-output alignment, visual quality, reasoning validity, rule-based executability, and efficiency penalties. During execution, CanvasAgent follows a visual-first protocol and maintains explicit image-asset references, allowing it to revise plans, switch tools, or roll back to earlier outputs.

Our contributions are summarized as follows:

- We introduce **CanvasAgent**, the first tool-augmented multimodal agent designed for complex image creation and editing. CanvasAgent moves beyond passive visual perception by actively orchestrating heterogeneous visual tools for generation, editing, extraction, composition, transformation, and enhancement through multi-turn reasoning.
- We construct **CanvasCraft**, the first large-scale multimodal tool-use dataset for complex image creation and editing. CanvasCraft covers diverse creation scenarios, tool combinations, and multi-turn visual workflows, with **CanvasCraft-SFT** providing fully annotated multi-step execution trajectories and **CanvasCraft-RL** providing diverse and challenging task specifications for reinforcement learning.
- We design a task-specific **hybrid reward** for complex visual creation and integrate it into a two-stage SFT+GRPO training framework. The reward combines LLM-as-judge signals for image-prompt alignment, aesthetic quality, and trajectory validity with rule-based process checks and efficiency penalties, enabling robust optimization of final images and tool-use processes.

2 Related Work

Tool-Augmented Multimodal Agents Tool use has become a central mechanism for extending language and multimodal models beyond direct generation. ReAct [30] formalizes the reason-action-observation pattern, and early multimodal systems connect language models with visual foundation models or expert modules for planning, execution, and programmatic visual reasoning [26, 29, 18, 6, 22, 11]. Recent agentic MLLMs further develop active visual inspection, search, and executable reasoning through operations such as cropping, zooming, Python execution, and heterogeneous tool use [36, 7, 33, 24, 17, 35, 20, 2, 34]. These works mainly target perception, visual search, reasoning, or general multimodal assistance. Our work instead focuses on complex image creation and editing workflows, where the agent must produce a final visual artifact by coordinating generation, localization, editing, compositing, OCR, and enhancement tools across stateful trajectories.

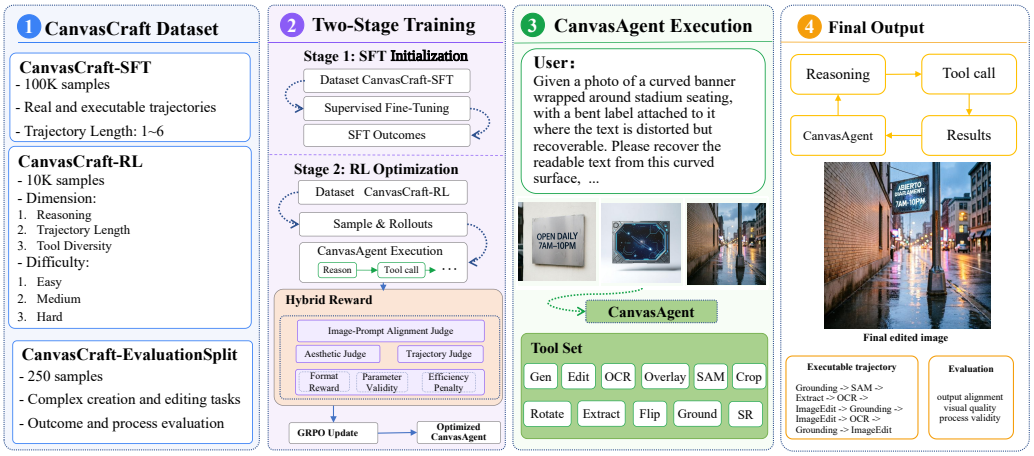

Figure 1: Overview of CanvasCraft and CanvasAgent. CanvasCraft provides supervised tool-use trajectories and RL task specifications for training CanvasAgent, which learns to orchestrate visual tools for complex image creation and editing.

Image Editing and Creation Workflows Image generation and editing models have made substantial progress on individual operations. Latent diffusion models enable high-quality synthesis [13], and instruction-guided editors improve controllable editing, generation, and text rendering [1, 31, 4, 19, 25]. These models are strong building blocks, but they typically execute a single instruction in one model call and do not explicitly manage multi-step tool dependencies or multiple intermediate visual assets.

Photo-retouching agents are closer to our setting because they coordinate editing operations over multiple steps. JarvisArt [9] controls Lightroom operations with an MLLM agent, and JarvisEvo [10] studies a self-evolving edit-evaluate-reflect loop. However, their environment is primarily designed for photo retouching. Our setting targets open-ended image creation and editing workflows that may require generation, grounding, segmentation, object extraction, compositing, cropping, OCR, geometric transformation, and super-resolution within the same trajectory.

Learning Tool-Orchestration Policies Learning effective tool use requires more than exposing a model to tool descriptions. Supervised fine-tuning can teach invocation schemas and reasoning-action formats, but imitation alone may overfit to static demonstrations and fail to discover better long-horizon strategies. Reinforcement learning has therefore been used to optimize tool-use policies, from PPO [15] and GRPO [16] to recent search, visual reasoning, and tool-use systems [8, 27, 21, 32, 3, 5, 23, 28].

Complex image creation and editing pose a different optimization problem. The agent must select tools, set tool parameters, track intermediate assets, and decide whether the current visual result is sufficient. Reward design must therefore evaluate both the final output image and the execution process. We address this with a two-stage SFT+GRPO framework and a hybrid reward that jointly optimizes visual reasoning, image quality, trajectory validity, and robust tool-use behavior.

3 Method

This section describes the data construction and training framework of CanvasAgent, illustrated in Fig. 1. We first construct CanvasCraft, a large-scale multimodal tool-use dataset for complex image creation and editing. It contains CanvasCraft-SFT, which provides fully annotated multi-step trajectories, and CanvasCraft-RL, which provides diverse task-level specifications for RL training. We then train CanvasAgent with a two-stage SFT+RL framework: supervised fine-tuning bootstraps stable tool-use behavior from expert trajectories, while reinforcement learning with GRPO improves long-horizon planning with a task-specific hybrid reward. CanvasAgent operates with 11 specialized visual tools covering generation, 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, enabling adaptive multi-tool orchestration guided by user instructions and evolving visual feedback.

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. 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.

3.1.1 Dataset Construction

Table 1: 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	`image`, `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`	$4\times$ enhanced image	Real-ESRGAN

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 Table 1. 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 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 [12], and reverse-engineer a natural user instruction from the intended tool-use behavior. The instruction is then executed in the tool environment to produce reasoning traces, JSON-style tool calls, intermediate assets, and final images. We retain only trajectories that pass quality control, including

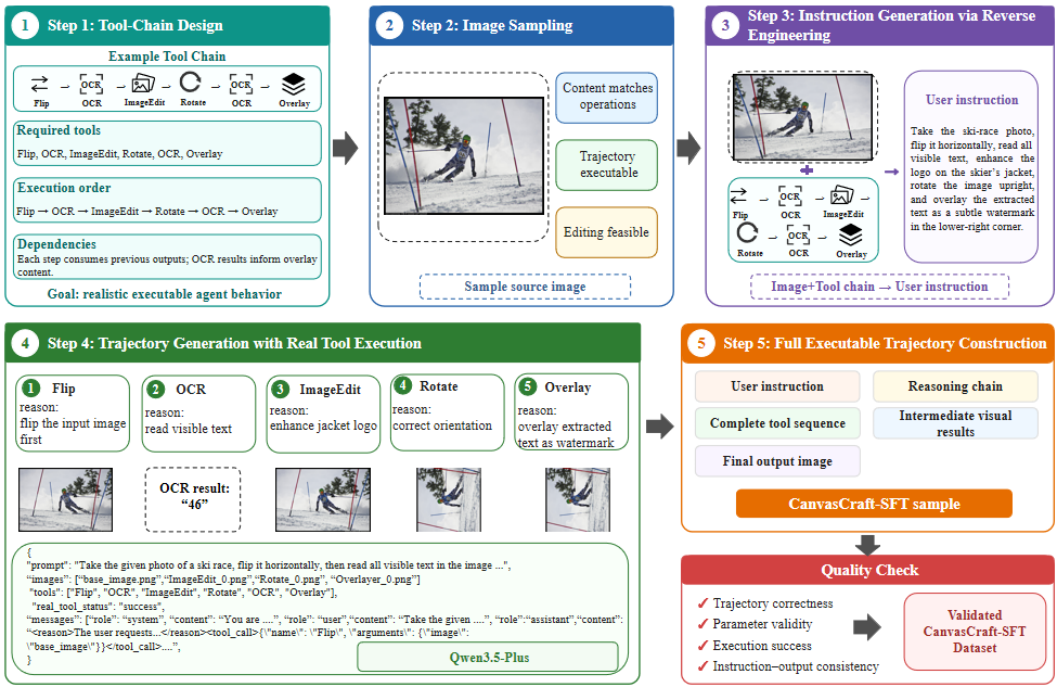

Figure 2: CanvasCraft-SFT data construction pipeline. The pipeline constructs executable tool-use trajectories through tool-chain design, image sampling, instruction generation via reverse engineering, and real tool execution with quality control.

checks for tool-call parsability, parameter validity, asset-reference consistency, execution success, and redundancy.

Construction of CanvasCraft-RL. Unlike CanvasCraft-SFT, CanvasCraft-RL only contains the expected tool set serves 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 Fig. 3, 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 expand each seed into a natural user instruction with an expected tool set $\mathcal{T}^*$. After task generation, each sample is characterized along the R/L/D dimensions and assigned difficulty levels according to the criteria in Table 2. The generated tasks are then filtered by DeepSeek-V4-Flash and further verified by human annotators to remove ambiguous, overly simple, or R/L/D-inconsistent instances. For visually grounded tasks, we generate one or more initial images aligned with the instruction.

Together, CanvasCraft-SFT and CanvasCraft-RL provide complementary supervision: the former teaches executable tool-use patterns through complete trajectories, while the latter enables reinforcement learning to optimize flexible planning and dynamic multi-tool orchestration under task-level weak supervision.

3.1.2 Dataset Composition

CanvasCraft contains **140K** SFT trajectories, **10K** RL task specifications, and a manually curated **250**-sample evaluation benchmark. A representative data instance is shown in Fig. 4. The dataset covers diverse visual tools, tool-chain structures, and task difficulties, supporting both supervised tool-use bootstrapping and RL-based policy optimization.

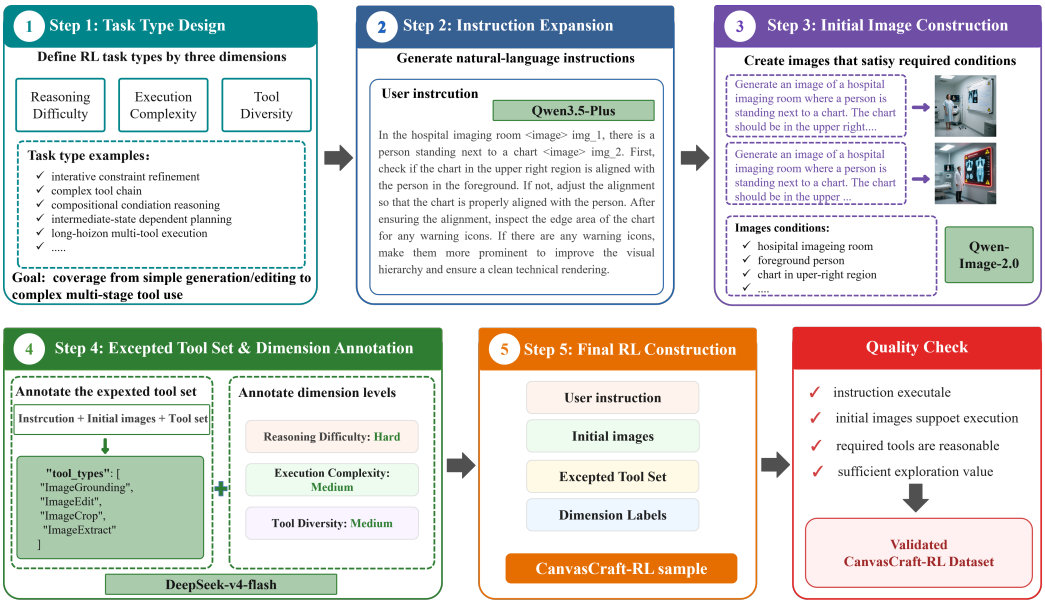

Figure 3: CanvasCraft-RL data construction pipeline. The pipeline generates difficulty-aware tasks by expanding task seeds into user instructions, constructing aligned initial images when needed, annotating expected tool sets, and filtering ambiguous or low-quality instances for RL training.

Table 2: 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

CanvasCraft-SFT providing 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.

Table 3: Difficulty distribution of the 10K CanvasCraft-RL tasks across the R/L/D dimensions.

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

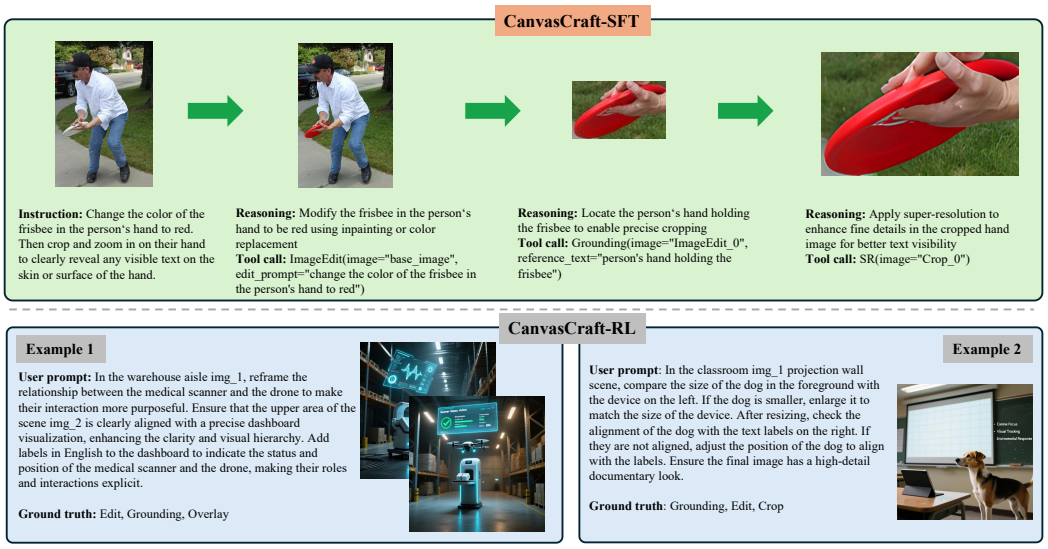

Figure 4: CanvasCraft data example. CanvasCraft-SFT provides the complete execution chain, including step-level reasoning and tool calls. CanvasCraft-RL provides task specifications without complete trajectories, requiring agents to explore tool ordering, parameterization, and intermediate operations during rollout.

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 Table 3. Unlike CanvasCraft-SFT, this subset only provides task-level supervision rather than fully annotated trajectories, encouraging the agent to learn task decomposition, state-dependent planning, and heterogeneous tool orchestration during reinforcement learning.

3.2 Two-Stage Training Framework

Based on the two subsets of CanvasCraft, we train CanvasAgent with a two-stage SFT+GRPO framework as shown in Fig. 1. SFT bootstraps stable tool-use behavior from complete executable trajectories, while GRPO optimizes rollout-level multi-tool planning under weak supervision. This design combines the reliability of trajectory imitation with the flexibility of reinforcement learning-based policy exploration.

3.2.1 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.

3.2.2 RL for Image Planning and Manipulation

After SFT initialization, we further optimize CanvasAgent on CanvasCraft-RL with group relative policy optimization (GRPO) [16]. 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

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

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 $\mathcal{A}_i = \mathcal{A}_{i-1}$. 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, and instead encourage concise, purposeful, and task-relevant tool orchestration.

Together, the hybrid reward balances **semantic alignment, visual aesthetics, reasoning validity, rule adherence, and execution efficiency**. By combining semantic, perceptual, procedural, and symbolic constraints, it reduces reward hacking and provides rich supervision for robust multi-step visual creation.

4 Experiments

4.1 Experiment Settings

Evaluation set. We evaluate all models on the CanvasCraft-RL evaluation split, which contains 250 samples. Each sample includes a user instruction, an optional input image, and a reference tool set 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, we include **Qwen-Image-2.0**, **Wan2.7-Image**, and **GPT-Image-2** as image-only reference models. Since they directly generate or edit the final image without producing reasoning–action–observation trajectories, their results serve only as outcome-level references. We therefore report only alignment and aesthetic scores for these models. Finally, we compare two CanvasCraft-trained variants. **CanvasAgent (SFT)** is trained only on CanvasCraft-SFT, while **CanvasAgent (SFT+RL)** is our full model trained with supervised fine-tuning followed by GRPO-based reinforcement learning on CanvasCraft-RL.

Training setup. All experiments are conducted on 8 NVIDIA A800 GPUs over 7 days. 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.

Metrics. We report six metrics. **Overall Reward** is the final hybrid reward for trajectory-level evaluation. **Alignment Score** measures whether the generated image satisfies the user instruction. **Aesthetic Score** measures visual quality and appeal. **Trajectory Score** evaluates the reasoning and tool-use trajectory. **Rule-based Score** measures format validity, parameter validity, and efficiency. **Trajectory Length** reports the average number of executed tool calls.

Table 4: Overall evaluation on the CanvasCraft-RL evaluation split.

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

4.2 Experimental Results and Analysis

Table 4 presents the overall evaluation results on the full evaluation set. We report the hybrid reward, three LLM-as-judge scores for alignment, aesthetics, and trajectory quality, the rule-based score, and the average trajectory length.

Compared with Qwen3-VL-8B-Instruct, CanvasAgent (SFT) improves the overall reward from 0.426 to 0.557 and raises the trajectory judge 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, producing only 1.320 tool calls on average compared with the expected 3.592 calls. Supervised learning therefore establishes initial tool-use capability but remains limited in active multi-tool planning.

CanvasAgent (SFT+RL) further improves all major metrics. Compared with CanvasAgent (SFT), it increases the overall reward from 0.557 to 0.821, image–prompt alignment from 0.613 to 0.869, trajectory quality from 0.576 to 0.849, and the rule-based score from 0.467 to 0.785. The average number of tool calls also increases from 1.320 to 5.436, indicating that RL encourages richer multi-step visual manipulation. These results show that CanvasCraft-RL and the hybrid reward help CanvasAgent learn dynamic task decomposition, tool planning, and adaptive decision-making from intermediate visual observations.

During RL training, the number of tool calls first increases and then gradually stabilizes or decreases, suggesting a transition from active exploration to more efficient tool orchestration under the guidance of process and efficiency rewards.

We include **Qwen-Image-2.0**, **Wan2.7-Image**, and **GPT-Image-2** as image-only reference models. Since these models directly generate or edit the final image without producing reasoning–action–observation trajectories, we report only outcome-level metrics, namely alignment and aesthetic scores. Their alignment scores are lower than CanvasAgent (SFT+RL), suggesting that strong image-only models still struggle with complex image editing tasks that require multi-step tool orchestration.

4.3 Ablation Studies

Table 5 ablates the training strategy and reward design. SFT-only training establishes executable reasoning–action patterns, reaching an overall reward of 0.557 and a trajectory score of 0.576. RL from scratch improves the overall reward to 0.604, but lowers alignment and aesthetics to 0.472 and 0.666, suggesting unstable exploration without an SFT initialization. SFT+RL performs best across all metrics, reaching 0.821 overall reward, 0.869 alignment, 0.762 aesthetics, and 0.849 trajectory score.

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 5: Ablation study of training strategy and hybrid reward design.

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

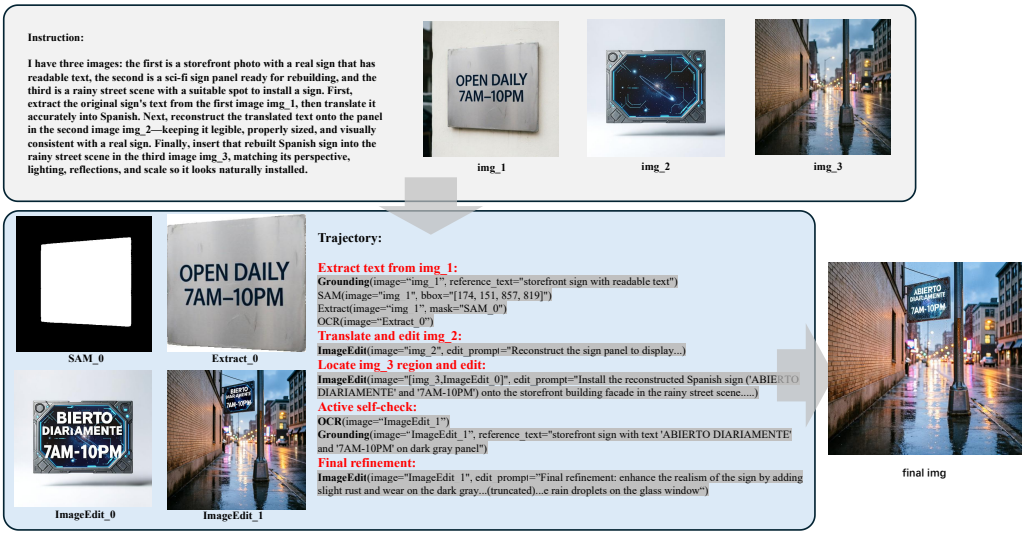

Figure 5: Qualitative case study of CanvasAgent on a complex multi-image editing task.

4.4 Case Study

Figure 5 presents a representative multi-image editing task. The user provides three input images: a storefront sign containing readable English text, a blank sign panel for reconstructing the translated sign, and a rainy street scene into which the final sign should be inserted. Solving this task requires substantially more than direct image generation or editing. The agent must localize the original sign, extract and recognize its text, reconstruct the translated sign on the blank panel, and finally composite the result into the target scene while preserving perspective, lighting, scale, and reflections.

CanvasAgent completes this task through a multi-turn reasoning and tool-use trajectory involving perception, extraction, editing, OCR, and compositing tools. During execution, the agent generates and manages multiple intermediate visual assets, including segmentation masks, extracted sign regions, reconstructed sign panels, and edited scene images. These assets are explicitly referenced by subsequent tool calls, enabling the agent to coordinate multiple input images and intermediate results within a unified trajectory.

This case also demonstrates closed-loop visual tool orchestration. After performing transformation or editing operations, CanvasAgent invokes perception tools such as grounding and OCR to inspect intermediate results and verify whether the generated content satisfies the task requirements. The agent then uses these observations to guide subsequent editing and compositing decisions. This behavior demonstrates that CanvasAgent does not merely generate a final image in a single step; instead, it actively manipulates visual states through iterative reasoning, tool execution, observation, and refinement.

Table 6: Human evaluation on 12 CanvasCraft evaluation samples.

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 (Ours)	**3.97**	**3.90**	**4.06**

4.5 Human Evaluation

We conduct a user study on 12 CanvasCraft evaluation samples to assess whether our judge-based evaluation aligns with human preferences.. Annotators score each output from 1 to 5 on Task Alignment, Key Details Alignment, and Aesthetic Quality.

As shown in Table 6, CanvasAgent achieves the best scores across all dimensions, indicating better instruction satisfaction, key information preservation, and visual quality. The three dimensions also align with our judge design for alignment, semantic/text correctness, and aesthetics.

5 Conclusion

We present CanvasAgent, a tool-augmented multimodal agent for complex image creation and editing. Rather than treating image generation as a single black-box call, CanvasAgent decomposes open-ended visual requests into executable multi-step tool trajectories. CanvasCraft provides the supervision needed for this setting, with 140K SFT trajectories and 10K curated RL task specifications spanning diverse editing, composition, perception, and verification behaviors. 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. Across automatic and human evaluations, CanvasAgent shows consistent gains from supervised trajectory learning, reinforcement learning, and the combination of outcome- and process-level feedback.

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. Future work will study dynamic tool discovery, learned or self-evaluation rewards, more efficient rollout strategies, user feedback, self-improvement, and extensions to video creation and editing.

References

[1] 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)*, pages 18392–18402, 2023.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

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

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

[8] 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.

[9] 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*, 2025.

[10] 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*, 2025.

[11] 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.

[12] 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.

[13] 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)*, pages 10684–10695, 2022.

[14] Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J. Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 45(4):4713–4726, 2022.

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

[16] 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.

[17] 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, 2025.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

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

[23] 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*, 2025.

[24] 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*, 2025.

[25] 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*, 2025.

[26] 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.

[27] 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*, 2025.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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*, 2025.

[33] 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*, 2025.

[34] 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*, 2025.

[35] 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.

[36] 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}
```

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**

A.4 CanvasCraft-SFT Distribution

A.5 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.