FiRe: Fine-grained Multimodal Reasoning for Enhanced Image Generation

Yongjin Kim[1] **Yoonjin Oh**[1*] **Yerin Kim**[1*] **Hyomin Kim**[1]
Jeeyoung Yun[1] **Yu-Jung Heo**[2] **Min-Jun Kim**[2] **Sungwoong Kim**[1†]

[1]Department of Artificial Intelligence, Korea University, Seoul, Republic of Korea
[2]KT Corporation, Seoul, Republic of Korea
{rla020, dhdbsrlw, yerin-kim, khmiee, jee010910, swkim01}@korea.ac.kr
{yj.heo, min_jun.kim}@kt.com

Abstract

With the rapid progress of Multimodal Large Language Models (MLLMs), unified MLLMs that jointly perform image understanding and generation have advanced significantly. However, despite the inherent reasoning capabilities of unified MLLMs for self-reflection and self-refinement, their use in text-to-image generation remains largely underexplored. Meanwhile, existing multimodal reasoning–based image generation methods mostly rely on prompt augmentation or holistic image–text alignment judgments, without fine-grained reflection and refinement of detailed prompt attributes, leading to limited fine-grained control. To address this limitation, we propose **FiRe**, a **Fi**ne-grained Multimodal **Re**asoning method for enhanced image generation by MLLM. In specific, FiRe performs a fine-grained multi-step reasoning by first decomposing the prompt into key visual requirements and then self-judging their satisfaction in the generated image, followed by localized refinement according to self-generated precise feedback. In addition, to further strengthen the MLLM's multimodal reasoning ability, we introduce **FiRe-GRPO**, a reinforcement learning method tailored to FiRe. Since standard Group Relative Policy Optimization (GRPO) suffers from sparse, outcome-based rewards in multi-step reasoning, we formulate our reasoning process as a step-level decision-making problem, design step-specific rewards, and compute step-level advantages for granular credit assignment within GRPO. Extensive experiments demonstrate that FiRe consistently outperforms competitive text-to-image baselines, including existing reasoning-based methods, with particularly substantial gains on compositional text-to-image benchmarks. Our project page is available at `https://ku-agi.github.io/FiRe/`.

1 Introduction

Recent advances in unified Multimodal Large Language Models (MLLMs) [1–9] have enabled image understanding and image generation within a single model. However, reasoning has been much more actively explored for image understanding than for image generation. In image understanding, reasoning techniques such as chain-of-thought prompting, self-reflection, and self-refinement have been widely explored [10–15], enabling models to use additional inference-time computation to reason, verify, and refine their answers. In contrast, reasoning for text-to-image (T2I) generation remains underexplored. Although recent methods [16, 17] introduce refinement processes into T2I generation, they still struggle with complex compositional prompts involving multiple objects, attributes, and spatial relations, as their coarse holistic judgments cannot reliably verify fine-grained prompt requirements.

*Equal contribution.
†Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

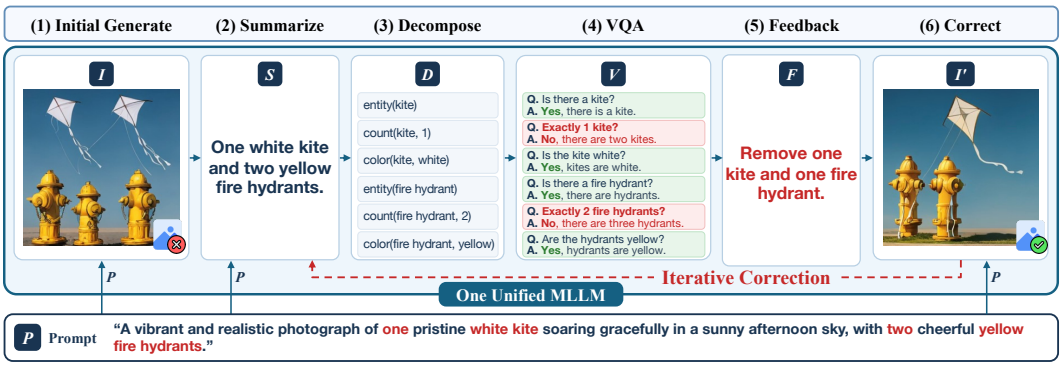

Figure 1: **Overview of FiRe.** FiRe iteratively refines the alignment between input prompts and output images through six steps. **Step 1:** generates an initial image from the input prompt. **Step 2:** summarizes the prompt into concrete visual details that can be checked in the image. **Step 3:** decomposes the summarized prompt into semantic tuples representing atomic visual elements. **Step 4:** verifies each tuple against the current image through tuple-level VQA. **Step 5:** generates fine-grained feedback for tuples judged as unsatisfied. **Step 6:** applies the feedback to perform localized image correction. This entire loop is repeated until the image–prompt alignment is fully satisfied.

To address this limitation, we propose **FiRe**, a **Fi**ne-grained Multimodal **Re**asoning method for Enhanced Image Generation with an MLLM. FiRe partitions the reasoning process into fine-grained steps to ensure that every compositional requirement is captured. As shown in Figure 1, FiRe consists of six reasoning steps: **(1) Initial T2I Generation**, **(2) Prompt Summarization**, **(3) Tuple Decomposition**, **(4) Tuple VQA**, **(5) Fine-grained Feedback Generation**, and **(6) Localized Image Correction**. FiRe first generates an initial image and summarizes the prompt by retaining concrete visual details while removing abstract or subjective descriptions. It then decomposes the summarized prompt into independently verifiable units and checks each unit through visual question answering (VQA). Based on the unsatisfied units, FiRe generates feedback and applies localized corrections only to mismatched regions. Through this fine-grained method, FiRe can identify unsatisfied compositional requirements that holistic judgment would otherwise miss.

To further improve fine-grained reasoning capability, we introduce FiRe-GRPO, a reinforcement learning method designed for FiRe. The standard Group Relative Policy Optimization (GRPO) [18] treats the entire generation trajectory as a single action and assigns rewards only based on the quality of the final outcome. This outcome-based formulation often suffers from sparse feedback, as it fails to capture fine-grained signals from individual intermediate reasoning steps within the same trajectory. To address this limitation, FiRe-GRPO formulates our six-step reasoning process as a step-level decision-making problem. Since the six-step formulation enables a fine-grained self-evaluation process, FiRe-GRPO defines step-specific rewards and computes step-level advantages, rather than relying solely on a single outcome reward. This provides targeted learning signals for intermediate reasoning decisions, including prompt summarization, tuple decomposition, tuple-level VQA, feedback generation, and localized image correction.

Experiments on representative compositional T2I benchmarks, including GenEval [19], GenEval++ [20], and DPG-Bench [21], demonstrate that FiRe consistently outperforms existing baselines. Our main contributions are summarized as follows:

- We propose FiRe, a fine-grained multimodal reasoning method for text-to-image generation that identifies prompt–image misalignments at a fine-grained level and corrects only the mismatched regions through localized refinement.
- We introduce FiRe-GRPO, a step-level reinforcement learning method for FiRe that assigns rewards according to the role of each reasoning step and computes step-level advantages, enabling fine-grained credit assignment to intermediate judgments, feedback, and corrections.
- We conduct extensive experiments and ablation studies on compositional T2I benchmarks, showing that FiRe consistently outperforms strong image generation and reasoning-based baselines.

2 Method

We first define the six fine-grained reasoning steps of FiRe. We then describe supervised fine-tuning for learning this reasoning process, followed by FiRe-GRPO, a step-level reinforcement learning

method that further strengthens step-wise reasoning ability. All six steps are performed by a single unified MLLM, without any external generator, editor, or evaluator at inference. External models are used only during training, for SFT data construction (Section 2.2) and reward computation in FiRe-GRPO (Section 2.3). Figure 1 provides an overview of FiRe, and we detail each reasoning step below.

2.1 Fine-grained Reasoning Steps in FiRe

Step 1: Initial Text-to-Image Generation. FiRe starts by generating an initial image I from the input prompt p. The initial image is expected to be aligned with the prompt, but may still miss or incorrectly render fine-grained visual details. FiRe therefore uses I as the starting point for the subsequent steps.

Step 2: Prompt Summarization. After generating the initial image I, FiRe summarizes the input prompt p into S by retaining concrete visual details that can be explicitly checked in the image, such as objects, attributes, counts, spatial relations, and text, while removing subjective or non-verifiable intangible descriptions, such as *beautiful*, *vibrant*, or *aesthetic*.

Step 3: Tuple Decomposition. FiRe then decomposes S into semantic tuples $D = \{\tau_i\}_{i=1}^{Q}$. Each tuple τ_i captures an atomic visual element, such as an entity, attribute, relation, count, spatial constraint, or text element. For example, if the summarized prompt S contains "a pink oven", FiRe decomposes it into `whole-entity(oven)` and `attribute-color(oven, pink)`.

Step 4: Tuple VQA. Each semantic tuple in D is checked against the current image I through tuple-level VQA, producing tuple VQA results $V = \{v_i\}_{i=1}^{Q}$, where $v_i = (r_i, y_i)$. Each result v_i contains an image-grounded rationale r_i and a binary judgment $y_i \in \{\texttt{yes}, \texttt{no}\}$ indicating whether the visual requirement represented by τ_i is satisfied. This step converts each tuple into an explicit verification result. For example, suppose the prompt requires a pink oven, while the generated image contains a white oven. For the tuple $\tau_i =$ `attribute-color(oven, pink)`, FiRe asks the verification question "Is the oven pink?" and may produce $(r_i, y_i) =$ ("The oven is white rather than pink.", `no`).

Step 5: Fine-grained Feedback Generation. Using the tuple-level verification results $V = \{(r_i, y_i)\}_{i=1}^{Q}$, FiRe generates feedback F based on whether the tuples are satisfied. If all tuples are satisfied, FiRe terminates the correction process with $F =$ `no-edit`; otherwise, it generates actionable feedback $F = \tilde{F}$ for correcting the unsatisfied tuples $\{\tau_i \mid y_i = \texttt{no}\}$. When multiple tuples are unsatisfied, $\tilde{F}$ combines the required corrections into a single feedback instruction so that all identified mismatches are addressed together. For example, if the prompt requires `attribute-color(oven, pink)` and `count(plate, 3)`, but the generated image contains a white oven and only two plates, FiRe judges both requirements as `no` and may produce $\tilde{F} =$ "Change the oven color from white to pink and add one more plate to make three plates."

Step 6: Localized Image Correction. When $F \neq$ `no-edit`, FiRe applies the feedback F to the current image I and produces a corrected image I'. This localized correction is performed using an editing prompt that includes F together with an explicit preservation instruction, thereby targeting the unsatisfied tuples identified in the previous steps while preserving already satisfied content. FiRe then sets $I \leftarrow I'$ and uses the corrected image as the input to Step 2 in the next correction attempt.

Detailed input prompts and FiRe's inference algorithm are provided in Appendices B and C.

2.2 Supervised Fine-tuning for Multimodal Reasoning

Starting from a pretrained unified MLLM [1] π_θ, we first use supervised fine-tuning (SFT) to equip the model with FiRe's fine-grained multimodal reasoning ability. We construct an SFT dataset $\mathcal{D}_{\mathrm{SFT}}$ consisting of supervised FiRe trajectories from both prompt-aligned and prompt-misaligned initial images. If the initial image I is already aligned with the prompt, the trajectory contains the intermediate reasoning outputs and ends with $F =$ `no-edit`; otherwise, it further includes fine-grained feedback F and a corrected image I':

$$(I, S, D, V, F) \in \mathcal{D}_{\mathrm{SFT}} \quad \text{if } F = \texttt{no-edit}, \qquad (I, S, D, V, F, I') \in \mathcal{D}_{\mathrm{SFT}} \quad \text{otherwise.} \tag{1}$$

Here, I is the initial image, S is the summarized prompt, D is the tuple decomposition, V is the tuple-level VQA result, F is the feedback, and I' is the corrected image.

We train the model using token-level cross-entropy over the supervised FiRe output sequences. For each supervised output sequence $Y = (y_1, \ldots, y_T)$, let X denote the corresponding input context.

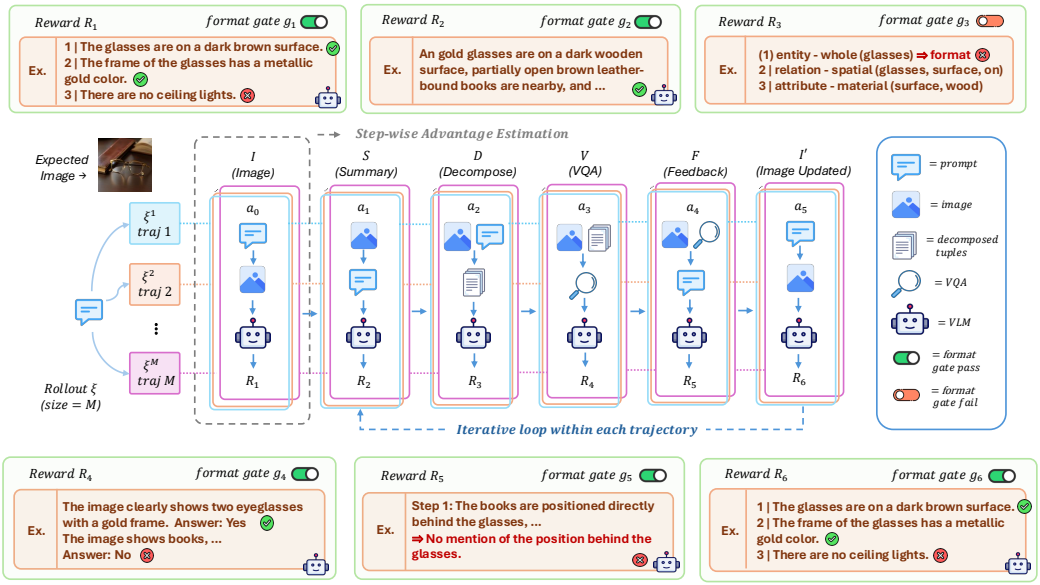

Figure 2: **Overview of FiRe-GRPO**. FiRe-GRPO performs step-wise advantage estimation at the level of individual FiRe reasoning steps. At each step, the policy samples a group of outputs, and the reward model assigns a step-specific reward to each output. For Steps 2–5, a format gate sets the reward to zero when the output violates the required structured format. The advantages are then computed within the corresponding step-level reward group.

For example, $X = p$ for initial image generation in Step 1, $X = (p, I)$ for the fine-grained reasoning trace in Steps 2–5, and $X = (p, I, F)$ for localized image correction in Step 6. The SFT loss is

$$\mathcal{L}_{\mathrm{SFT}}(\theta) = -\frac{1}{T}\sum_{\ell=1}^{T} \log \pi_\theta(y_\ell \mid X, y_{<\ell}), \tag{2}$$

where T denotes the target sequence length. The data preparation procedure used to construct $\mathcal{D}_{\mathrm{SFT}}$ is provided in Appendix D.

2.3 FiRe-GRPO: Step-level Reinforcement Learning

Although SFT equips the model with FiRe's fine-grained reasoning process, it learns by imitating supervised trajectories rather than optimizing rewards over trajectories sampled from the model's own policy. To extend FiRe beyond SFT, we introduce FiRe-GRPO represented by Figure 2, a reinforcement learning method that optimizes FiRe using step-specific rewards and step-level advantages.

Step-level Sequential Decision-making Formulation. To formulate FiRe as a sequential decision-making problem, we define each action at the level of a step output. Namely, an output action in FiRe by π_θ can be the initial image I, summarized prompt S, tuple decomposition D, tuple-level VQA results V, feedback F, or corrected image I'.

At each action (decision) position h corresponding to each step, the state s_h is the multimodal context used to generate the current step output. It contains the input prompt, the current image, and the FiRe outputs generated in previous steps, from which the policy samples a step-level action $a_h \sim \pi_\theta(\cdot \mid s_h)$.

Since FiRe may repeat Steps 2–6 over multiple correction attempts, a rollout contains a sequence of such step-level actions. Given a prompt p, the policy π_θ samples a group of M FiRe rollouts $\{\xi^{(i)}\}_{i=1}^{M}$. We index the actions in each rollout by their generation order, $h = 0, \ldots, H_i - 1$, and write $\xi^{(i)} = \{(s_h^{(i)}, a_h^{(i)}, r_h^{(i)})\}_{h=0}^{H_i-1}$. where, for each action $a_h^{(i)}$ that is associated with a step label $k \in \{1, \ldots, 6\}$ (as defined in Section 2.1), $r_h^{(i)} = R_k(s_h^{(i)}, a_h^{(i)})$ is the step-level reward computed by the corresponding reward function R_k.

2.3.1 Step-specific Reward Design

Each FiRe step has a specific role in the reasoning process, and the model should learn to perform each step according to its intended function. Existing GRPO-based methods often optimize generated outputs using a scalar reward computed from the final outcome [17, 18]. However, a final-image-only reward is insufficient for FiRe: it does not provide direct training signals for intermediate reasoning outputs, such as the prompt summary, tuple decomposition, tuple-level verification, and feedback. We therefore design step-level rewards using three reward components: **fine-grained image alignment reward**, **textual reasoning evaluation reward**, and **instruction-following reward**. We use an off-the-shelf VLM-based evaluator $\mathcal{E}_{\text{vlm}}$ for reward computation.

Fine-grained image alignment reward. For Steps 1 and 6, we evaluate image–prompt alignment at a fine-grained level. Instead of asking a single binary question such as "Does the image match the {prompt}?", the evaluator follows the same procedure as FiRe's Steps 2–4: it summarizes the prompt, decomposes it into atomic visual elements, and converts them into verification questions. Specifically, the evaluator first summarizes the prompt p into an evaluator-side summary $S^\star$, and then decomposes $S^\star$ into $D^\star = \{\tau_i^\star\}_{i=1}^{Q}$. Each tuple $\tau_i^\star$ is converted into a verification question $q_i^\star$. For example, $\tau_i^\star =$ `attribute-color(oven, pink)` can be converted into $q_i^\star =$ "Is the oven pink?". We use $\star$ to indicate quantities produced by the evaluator. The image alignment reward is defined as the fraction of questions answered as `yes` among the total Q questions, i.e., $z_{\text{img}}(I, p) = \frac{1}{Q}\sum_{i=1}^{Q} \mathbf{1}[\mathcal{E}_{\text{vlm}}(I, q_i^\star) =$ `yes`$]$. This score z provides a fractional score that captures which specific visual requirements are satisfied or violated, rather than only measuring a single binary image–prompt alignment.

Textual reasoning evaluation reward. For Steps 2–5, the evaluator scores each output according to the function of its corresponding step. Step 2 evaluates whether the summarized prompt preserves checkable visual details while removing subjective or non-verifiable descriptions. Step 3 evaluates whether the tuple decomposition faithfully captures atomic visual elements without omission or hallucination. Step 4 evaluates whether the tuple-level VQA results provide image-grounded rationales and accurate `yes`/`no` judgments. Step 5 evaluates whether the feedback targets tuples judged as `no` and provides actionable correction guidance without unnecessary changes to already satisfied content. For these steps, we additionally apply a format gate, which assigns zero reward when the generated output does not follow the required step-specific structure, such as parseable tuples for Step 3, rationale with `yes`/`no` judgments for Step 4, and `no-edit` or actionable feedback for Step 5. This is necessary because these structured outputs are used as inputs to subsequent FiRe steps.

Instruction-following reward. For Step 6, we use the fine-grained image alignment reward for the corrected image I' and additionally incorporate an instruction following reward. The instruction-following reward evaluates whether the edit from the current image I to the corrected image I' faithfully follows the generated feedback F, while avoiding unnecessary modifications to visual content that already satisfied the prompt requirements.

Reward assignment. We assign a step-specific reward to each generated FiRe step output $a_h^{(i)}$ based on its step index k. Step 1 uses a fine-grained image alignment reward, Steps 2–5 use role-specific rewards with format gating, and Step 6 uses both a fine-grained image alignment reward and an instruction following reward. For example, when $k = 1$, $a_h^{(i)}$ is the initial image $I^{(i)}$, and the assigned reward is $r_h^{(i)} = R_k(s_h^{(i)}, a_h^{(i)}) = R_1(s_h^{(i)}, a_h^{(i)}) = z_{\text{img}}(I^{(i)}, p)$.

Detailed reward computation, required formats, and evaluator prompts are provided in Appendix E.

2.3.2 Stepwise Advantage Estimation

Aggregating all step-specific rewards into a single outcome reward and assigning one uniform advantage to the entire rollout would obscure which FiRe steps were more or less effective. FiRe-GRPO therefore computes advantages at the level of FiRe reasoning components, preserving step-aware credit assignment.

For each generation position h, we compute the advantage of action $a_h^{(i)}$ using its step-specific reward $r_h^{(i)}$. Given a rollout group sampled for the same prompt, the reward is normalized across the M rollouts at the same position h:

$$A_h^{(i)} = \frac{r_h^{(i)} - \text{mean}\left(\{r_h^{(j)}\}_{j=1}^{M}\right)}{\text{std}\left(\{r_h^{(j)}\}_{j=1}^{M}\right)}. \tag{3}$$

Table 1: Comparison on GenEval [19], GenEval++ [20], and DPGBench [21]. ↑ indicates higher is better. **Bold** and underline denote the best and second-best results among Multimodal Reasoning LLMs, respectively. Superscript * indicates results reproduced by us. (1st) denotes the initial T2I generation, while (2nd) and (3rd) represent the first and second rounds of image correction, respectively. Subcategory scores of DPGBench are provided in Appendix L.

Method	GenEval ↑							GenEval++ ↑								DPGBench ↑
	Overall	Single	Two	Count	Color	Pos	Color Attr	Overall	Color	Count	Color/ Count	Color/ Pos	Pos/ Count	Pos/ Size	Multi- Count	Overall
Diffusion Models																
SD3.5 Large [22]	0.74	0.99	0.94	0.72	0.89	0.33	0.60	-	-	-	-	-	-	-	-	84.08
Flux.1 Dev [23]	0.66	0.98	0.79	0.73	0.77	0.22	0.45	0.314	0.350	0.625	0.150	0.275	0.200	0.375	0.225	83.79
Unified Multimodal Large Language Models																
Emu3 [9]	0.52	0.98	0.69	0.33	0.78	0.15	0.16	-	-	-	-	-	-	-	-	81.60
Show-o2 [6]	0.76	1.00	0.87	0.58	0.92	0.52	0.62	-	-	-	-	-	-	-	-	86.14
Unitok [7]	0.59	0.99	0.75	0.39	0.80	0.21	0.41	-	-	-	-	-	-	-	-	81.87
Blip-3o [24]	0.84	-	-	-	-	-	-	0.307	0.250	0.250	0.125	0.600	0.125	0.575	0.225	81.60
BAGEL [25]	0.82	0.99	0.94	0.81	0.88	0.64	0.63	0.371	0.325	0.600	0.250	0.325	0.250	0.475	0.375	85.07
Janus-Pro-7B	0.80	0.99	0.89	0.59	0.90	0.79	0.66	0.246	0.450	0.300	0.125	0.300	0.075	0.350	0.125	83.78
Multimodal Reasoning Models																
T2I-R1* [17]	0.79	**1.00**	0.91	0.52	0.90	0.77	0.65	0.311	0.675	0.325	0.200	0.350	0.075	0.250	0.300	85.06
Janus-R1* (1st) [16]	0.80	**1.00**	0.92	0.48	0.91	0.80	0.71	0.290	0.481	0.306	0.244	0.369	0.094	0.300	0.238	83.58
Janus-R1* (2nd)	0.82	**1.00**	0.93	0.51	0.92	0.87	0.70	0.294	0.481	0.369	0.200	0.363	0.113	0.313	0.219	84.03
Janus-R1* (3rd)	0.83	**1.00**	**0.94**	0.51	0.93	0.88	0.72	0.285	0.481	0.281	0.200	0.344	0.150	0.306	0.231	84.13
FiRe (1st)	0.84	0.99	**0.94**	0.59	0.93	0.85	0.77	0.379	0.625	0.325	0.275	0.550	0.175	0.369	0.331	84.88
FiRe (2nd)	0.86	0.99	**0.94**	0.70	0.93	0.85	0.77	0.464	0.750	0.444	0.381	0.575	0.263	0.375	0.463	85.26
FiRe (3rd)	0.87	0.99	**0.94**	0.72	0.93	0.85	0.77	0.491	**0.775**	0.488	0.450	0.569	0.281	0.356	0.488	85.28
FiRe-2500 Steps (1st)	0.89	**1.00**	**0.94**	0.77	**0.94**	**0.90**	0.79	0.591	0.738	0.413	0.569	0.788	0.475	**0.756**	0.400	85.40
FiRe-2500 Steps (2nd)	**0.90**	**1.00**	**0.94**	**0.81**	**0.94**	**0.90**	**0.81**	0.684	0.731	0.550	**0.700**	0.788	**0.688**	0.750	0.581	85.97
FiRe-2500 Steps (3rd)	**0.90**	**1.00**	**0.94**	**0.81**	**0.94**	**0.90**	**0.81**	**0.686**	0.756	**0.581**	0.681	**0.798**	0.638	**0.756**	**0.594**	**86.01**

2.3.3 FiRe-GRPO Optimization Objective

Given a prompt $p \sim \mathcal{D}_{\text{GRPO}}$, where $\mathcal{D}_{\text{GRPO}}$ denotes the training prompt distribution, we sample a group of M rollouts $\{\xi^{(i)}\}_{i=1}^{M}$ from the old policy $\pi_{\theta_{\text{old}}}(\cdot \mid p)$. For each rollout $\xi^{(i)}$, we compute the step-specific rewards and the corresponding advantages as described above. We then incorporate our step-level advantages into the token-level GRPO objective [18]. In particular, for each rollout $\xi^{(i)}$, let T_i denote the total number of generated tokens across all FiRe step outputs, and let $z_t^{(i)}$ and $c_t^{(i)}$ be the t-th generated token and the autoregressive context used to generate $z_t^{(i)}$, respectively. The token-level importance ratio is $\rho_t^{(i)}(\theta) = \frac{\pi_\theta(z_t^{(i)}|c_t^{(i)})}{\pi_{\theta_{\text{old}}}(z_t^{(i)}|c_t^{(i)})}$. Then, we define the token-level clipped objective as

$$j_t^{(i)}(\theta) = \min\left(\rho_t^{(i)}(\theta)\widehat{A}_t^{(i)}, \text{clip}\left(\rho_t^{(i)}(\theta), 1-\epsilon, 1+\epsilon\right)\widehat{A}_t^{(i)}\right), \tag{4}$$

where ϵ is the hyperparameter for clipping. Here, it is noted that if a token $z_t^{(i)}$ is generated within action $a_h^{(i)}$, we set $\widehat{A}_t^{(i)} = A_h^{(i)}$. Finally, the FiRe-GRPO objective is

$$\mathcal{J}_{\text{FiRe-GRPO}}(\theta) = \mathbb{E}_{p,\{\xi^{(i)}\}}\left[\frac{1}{M}\sum_{i=1}^{M}\frac{1}{T_i}\sum_{t=1}^{T_i}\left(j_t^{(i)}(\theta) - \beta D_{\text{KL}}\big(\pi_\theta(\cdot|c_t^{(i)}) \,\|\, \pi_{\text{ref}}(\cdot|c_t^{(i)})\big)\right)\right], \tag{5}$$

where β and π_{ref} indicate the coefficient and the reference (SFT) model for the KL penalty, respectively. Our MLLM for FiRe, π_θ, is optimized by maximizing $\mathcal{J}_{\text{FiRe-GRPO}}(\theta)$.

3 Experiments

3.1 Experimental Setup

Backbone and baselines. We use Janus-Pro-7B [1] as the backbone for FiRe and compare against two reasoning-based baselines built on the same backbone: Janus-Pro-R1 [16], which performs holistic image assessment followed by full-image regeneration, and T2I-R1 [17], which augments

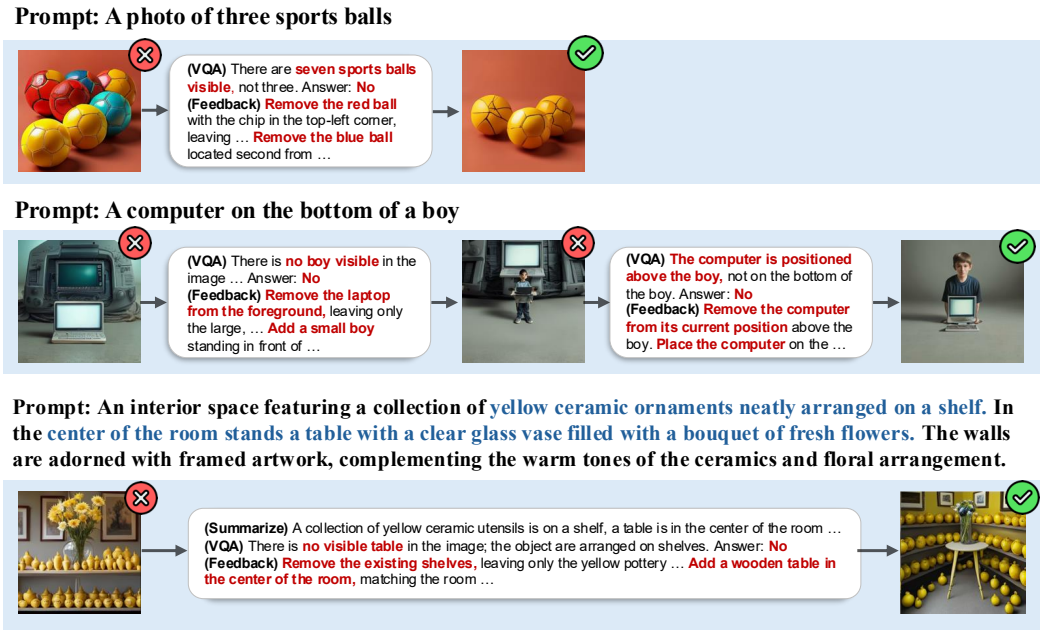

Figure 3: Qualitative Results of FiRe. From left to right, the first image denotes the initial T2I generation, the second image displays the result following the first round of image correction, and the rightmost image represents the output after the second round of image correction.

prompts via text-centric chain-of-thought reasoning prior to generation. We use Qwen3.5-35B-A3B [26] as the reward evaluator for FiRe-GRPO.

Training data. For supervised fine-tuning, we use 200K image-text pairs: 140K from FocusDiff [27] and 60K composition-specific samples synthesized with Qwen-Image and Qwen-Image-Edit [28]. The prompts for the synthesized samples are adapted from Flow-GRPO [29] and ReasonGen-R1 [30]. We additionally incorporate complex and long prompts from Echo-4o-Image [20] and LongAlign [31].

Benchmarks. We evaluate FiRe on GenEval [19], GenEval++ [20], and DPGBench [21]. Detailed descriptions of the benchmarks and evaluation protocols are provided in Appendix F. To ensure a fair comparison, we reproduce the results of all multimodal reasoning baseline models in the same environment under identical evaluation settings. Additional details are provided in Appendix G.

3.2 Quantitative Results

Table 1 compares FiRe with state-of-the-art diffusion models, unified MLLMs, and multimodal reasoning models. Among the baselines, only Janus-Pro-R1 and FiRe support iterative refinement. Therefore, we report their initial generation (1st) and two subsequent refinement stages (2nd and 3rd). At each refinement stage, the model edits the image only when its self-judgment indicates a prompt–image mismatch; otherwise, the previous image is retained. Notably, due to the substantial computational cost of separately training each FiRe-GRPO variant, all FiRe-GRPO results in the standard evaluation and ablation studies use the 300-step checkpoint described in Appendix G.1. We additionally report FiRe-2500 Steps, obtained by continuing the same FiRe-GRPO training to 2,500 steps. Overall, FiRe achieves the best performance across nearly all benchmarks, with further improvements from extended training.

3.3 Qualitative Results

Figure 3 presents representative FiRe results on different prompts. Across diverse compositional prompts, FiRe identifies fine-grained misalignments and selectively edits erroneous regions while preserving correctly aligned content. These examples show that iterative, fine-grained reasoning improves prompt–image alignment. Additional qualitative results are provided in Appendix O.

3.4 Ablation Study

To better understand the sources of FiRe's improvements, we conduct a series of ablation studies and analyses. Notably, all FiRe-GRPO results in the ablation studies use the 300-step checkpoint

Table 2: Ablation on reasoning step granularity. Comparison of fine-grained multi-step (w/ FG) and coarse single-step (w/ Coarse) reasoning across three refinement iterations on GenEval++.

Method	GenEval++ ↑							
	Overall	Color	Count	Color/ Count	Color/ Pos	Pos/ Count	Pos/ Size	Multi-Count
w/ FG (1st)	0.35	0.58	0.28	0.24	0.49	0.11	**0.39**	0.36
w/ FG (2nd)	0.41	0.66	0.39	0.30	0.54	0.16	**0.39**	0.42
w/ FG (3rd)	**0.44**	**0.68**	0.44	**0.34**	**0.56**	0.20	**0.39**	**0.46**
w/ Coarse (1st)	0.30	0.53	0.29	0.24	0.40	0.13	0.24	0.29
w/ Coarse (2nd)	0.40	0.64	0.44	0.27	0.54	0.22	0.27	0.39
w/ Coarse (3rd)	0.41	0.64	**0.49**	0.29	0.54	**0.23**	0.26	0.39

Table 3: Ablation of the Tuple Decomposition and VQA steps in FiRe-SFT on GenEval++, showing their contributions across three iterative refinement stages.

Method	1st	2nd	3rd
FiRe-SFT	0.348	**0.409**	**0.440**
w/o Tuple	**0.361**	0.403	0.433
w/o VQA	0.353	0.401	0.428
w/o Tuple+VQA	0.337	0.387	0.407

Table 4: Ablation on advantage estimation strategies in FiRe-GRPO on GenEval++.

Method	GenEval++ ↑							
	Overall	Color	Count	Color/ Count	Color/ Pos	Pos/ Count	Pos/ Size	Multi-Count
w/ Stepwise (1st)	0.38	0.63	0.33	0.28	0.55	0.18	0.37	0.33
w/ Stepwise (2nd)	0.46	0.75	0.44	0.38	**0.58**	0.26	0.38	0.46
w/ Stepwise (3rd)	**0.49**	**0.78**	0.49	**0.45**	0.57	**0.28**	0.36	**0.49**
w/ Aggregated (1st)	0.37	0.55	0.33	0.28	0.53	0.18	**0.44**	0.26
w/ Aggregated (2nd)	0.39	0.61	0.38	0.29	0.51	0.19	**0.44**	0.31
w/ Aggregated (3rd)	0.44	0.64	0.48	0.36	**0.58**	0.23	**0.44**	0.40
w/ Terminal (1st)	0.35	0.60	0.28	0.24	0.51	0.14	0.37	0.32
w/ Terminal (2nd)	0.42	0.63	0.43	0.33	0.55	0.26	0.38	0.37
w/ Terminal (3rd)	0.44	0.61	**0.53**	0.39	0.53	0.26	0.40	0.39

Table 5: Performance comparison between FiRe-SFT and FiRe-GRPO on GenEval++.

Method	GenEval++ ↑							
	Overall	Color	Count	Color/ Count	Color/ Pos	Pos/ Count	Pos/ Size	Multi-Count
Janus-Pro-7B	0.25	0.45	0.30	0.13	0.30	0.08	0.35	0.13
FiRe-SFT (1st)	0.35	0.58	0.28	0.24	0.49	0.11	0.39	0.36
FiRe-SFT (2nd)	0.41	0.66	0.39	0.30	0.54	0.16	0.39	0.42
FiRe-SFT (3rd)	0.44	0.68	0.44	0.34	0.56	0.20	0.39	0.46
FiRe-GRPO (1st)	0.38	0.63	0.33	0.28	0.55	0.18	0.37	0.33
FiRe-GRPO (2nd)	0.46	0.75	0.44	0.38	**0.58**	0.26	**0.38**	0.46
FiRe-GRPO (3rd)	**0.49**	**0.78**	**0.49**	**0.45**	0.57	**0.28**	0.36	**0.49**

described in Appendix G.1. We first examine the effect of reasoning step granularity and the role of step-wise advantage estimation in FiRe-GRPO. We then compare FiRe-SFT and FiRe-GRPO to assess the contribution of FiRe-GRPO beyond supervised fine-tuning. Finally, we analyze generalization across model scales, computational overhead, and unedited-region preservation during iterative correction.

Effect of Reasoning Step Granularity. To validate the effectiveness of FiRe's fine-grained reasoning process, we conduct two complementary ablations. First, we compare the original multi-step reasoning path with a coarse single-step evaluation variant. While FiRe decomposes reasoning into Steps 1–6, the coarse variant merges the intermediate evaluation Steps 2–4 into a single prompt-level assessment and directly generates feedback, thereby removing FiRe's fine-grained step-by-step alignment evaluation. Second, we independently ablate the Tuple Decomposition (Step 3) and VQA (Step 4) steps to examine the necessity of individual components within the fine-grained reasoning process. All variants use the same training images and identical SFT configuration, differing only in their intermediate reasoning supervision. Details of the coarse reasoning variant are provided in Appendix H. As shown in Table 2, FiRe's multi-step reasoning outperforms the coarse single-step variant across most GenEval++ categories, demonstrating the benefit of fine-grained reasoning. Table 3 further shows that removing either the Tuple Decomposition or VQA step reduces later-round performance, while removing both results in the largest degradation. Together, these results demonstrate that FiRe's refinement gains arise from its fine-grained reasoning design rather than iterative editing alone.

Effect of Advantage Estimation in FiRe-GRPO. For multi-step reasoning, a single outcome-level reward provides a sparse training signal [32, 33]: terminal rewards do not directly supervise intermediate steps, while aggregating step rewards mixes successful and unsuccessful steps into one trajectory-level signal. FiRe-GRPO instead computes rewards and advantages for individual FiRe reasoning steps (w/ Stepwise). We compare against w/ Terminal, which uses only the final image alignment reward, and w/ Aggregated, which uses the same step-specific rewards as w/ Stepwise but aggregates them into a single trajectory-level advantage shared across the reasoning steps. As shown in Table 4, w/ Stepwise performs best across most GenEval++ categories. The improvement over w/ Terminal demonstrates the benefit of dense intermediate rewards, while the improvement over

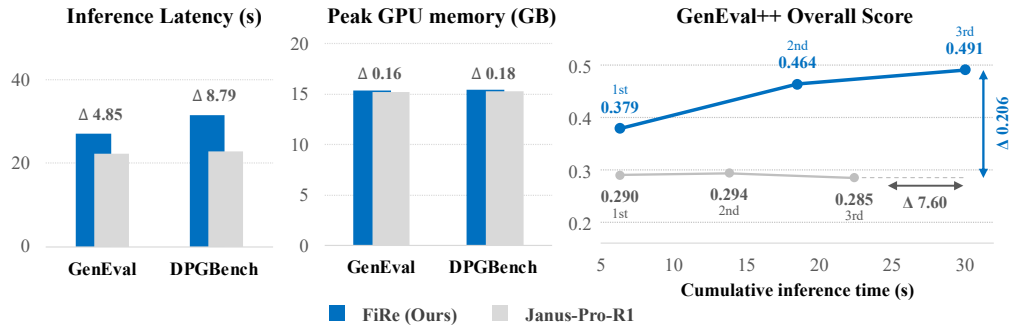

Figure 4: Efficiency and cost–performance comparison between FiRe and Janus-Pro-R1. Left: inference latency and peak GPU memory on GenEval and DPGBench [19, 21]. Right: GenEval++ [20] overall score as a function of cumulative inference time across three refinement rounds. FiRe achieves consistent performance gains with additional refinement, while Janus-Pro-R1 shows only a marginal second-round gain and degrades in the third round.

Table 6: Human-preference and aesthetic quality across refinement rounds on GenEval++, measured by HPSv3 and LAION-Aesthetic-v2.

	HPSv3 ↑ [34]			LAION-Aesthetic-v2 ↑ [35]		
Method	**1st**	**2nd**	**3rd**	**1st**	**2nd**	**3rd**
Janus-Pro [1]	6.3600	–	–	5.512	–	–
T2I-R1 [17]	7.3293	–	–	5.549	–	–
Janus-Pro-R1 [16]	7.4088	6.7799	6.7188	**5.685**	5.569	5.543
FiRe	**8.7062**	**8.5193**	**8.5592**	5.608	**5.602**	**5.600**

Table 7: Multimodal understanding performance before and after FiRe training.

Metric ↑	**Janus-Pro**	**FiRe**
MME-P [36]	1502	**1541**
POPE [37]	**86.35**	85.97
MMBench [38]	**78.30**	77.69
SEED-Bench [39]	**72.02**	71.31
GQA [40]	**61.65**	60.10

w/ Aggregated demonstrates the benefit of step-wise credit assignment. All variants use the same Qwen3.5-35B-A3B [26] reward evaluator.

Comparison between FiRe-SFT and FiRe-GRPO. To assess the effectiveness of FiRe-GRPO, we compare FiRe-GRPO with its supervised fine-tuned counterpart, FiRe-SFT. As shown in Table 5, FiRe-SFT already outperforms the baselines on GenEval++, a complex compositional benchmark, indicating the effectiveness of the proposed fine-grained reasoning method itself. FiRe-GRPO further improves performance across all refinement iterations, isolating the additional benefit of reinforcement learning beyond supervised fine-tuning.

Generalizability Across Model Scales. To verify scale generalizability, we apply the same SFT and GRPO training pipeline to Janus-Pro-1B. Consequently, FiRe consistently outperforms the base model (Appendix I), demonstrating that our fine-grained reasoning framework remains effective across different backbone sizes.

Computational Overhead Analysis. We analyze FiRe's computational overhead in Figure 4 from both per-round efficiency and cumulative cost–performance perspectives. All inference latency measurements are obtained with SGLang [41] on a single NVIDIA A100 GPU. The left panel compares inference latency and peak GPU memory with Janus-Pro-R1 [16] on GenEval and DPGBench, showing that FiRe incurs additional latency from fine-grained reasoning while introducing only modest memory overhead. The right panel shows the cost–performance trade-off across refinement rounds: FiRe consistently improves with additional inference time, whereas Janus-Pro-R1 yields only a marginal second-round gain and degrades in the third round. Detailed latency breakdowns and scaling with prompt length and tuple count are provided in Appendix J.

Quality and Capability Preservation. We further examine whether FiRe's iterative refinement and training compromise perceptual quality or multimodal understanding. As shown in Table 6, FiRe maintains strong human-preference and aesthetic quality across refinement rounds, while consistently outperforming Janus-Pro-R1 on HPSv3 [34]. Table 7 further shows that FiRe largely preserves the multimodal understanding capability of the Janus-Pro backbone, improving MME-P [36] while showing only minor decreases on the remaining benchmarks. We additionally evaluate preservation of unedited image regions in Appendix K.

Error Propagation, Failure, and Robustness Analysis. We analyze FiRe's failure modes and robustness in Appendix M, including degradation across refinement rounds, step-level failure attribution, error recovery, and robustness to visually under-specified prompts. FiRe exhibits fewer degradation cases than Janus-Pro-R1, explicit rechecking further improves refinement, and performance remains stable on visually under-specified prompts.

Fixed Initial Image Comparison. To examine whether FiRe's gains simply come from better initial image generation, we conduct a fixed-initial-image comparison in Appendix N. Specifically, we fix the initial images to those generated by Janus-Pro-R1 and compare subsequent refinement from the same starting images. FiRe achieves higher overall performance on both GenEval and DPGBench, showing that the improvement is not solely due to initial image quality but also comes from fine-grained reasoning and localized correction.

4 Related Works

Unified MLLMs. Recent unified MLLMs [1–9] increasingly integrate image understanding and generation within a single architecture. By unifying perception and generation within a single model, these architectures enable self-refinement entirely within the model itself, extending the chain-of-thought, self-reflection techniques extensively studied in image understanding [10–15]. This reasoning capability can also be leveraged to enhance image generation, allowing the model's own deliberative reasoning developed for perception to extend to generation as well.

Multimodal Reasoning for Text-To-Image Generation. Recent work on MLLM-based image generation via intermediate reasoning broadly falls into two paradigms: prompt augmentation [17, 30, 42], which enriches input prompts but lacks post-generation correction, and iterative self-reflection [16, 43, 44]. Both paradigms rely on coarse VQA judgments and generate feedback from this holistic assessment. FiRe instead performs six-step fine-grained reasoning. This enables FiRe to address fine-grained prompt–image mismatches that prompt augmentation and coarse self-reflection fail to capture.

Credit Assignment in GRPO. Recently, Group Relative Policy Optimization (GRPO) [18, 45] has emerged as an efficient policy optimization algorithm for LLM and MLLM training that estimates advantages from group-relative final-answer signals without a separate value function. However, in long Chain-of-Thought reasoning, GRPO's trajectory-level advantage yields coarse credit assignment, treating every token in a trajectory as equally responsible for the final outcome. To address this, recent works have investigated fine-grained credit assignment within GRPO—for example, step-wise advantages for multimodal reasoning [10], segment-level advantages for Chain-of-Thought reasoning [46, 47], and turn-level advantages for multi-turn tool use [48]. Following recent step-level extensions of GRPO such as GiGPO [49], FiRe-GRPO likewise enables fine-grained credit assignment by computing a separate group-relative advantage at each step rather than broadcasting a single trajectory-level signal.

5 Limitations and Conclusion

We introduced FiRe, a fine-grained multimodal reasoning method for text-to-image generation that structures iterative refinement into a six-step reasoning process, and FiRe-GRPO, which improves FiRe beyond supervised imitation by optimizing step-level reasoning and correction. Experiments show that FiRe improves prompt adherence across compositional benchmarks, especially on complex prompts involving attributes, counts, and spatial relations. As with other iterative image refinement methods, FiRe naturally depends on the reliability of intermediate reasoning outputs across refinement rounds; FiRe-GRPO mitigates this error accumulation problem by optimizing these steps with step-level rewards, and our error propagation analysis in Appendix M shows fewer degradation cases than Janus-Pro-R1 [16]. FiRe may also be limited in explicitly verifying visually under-specified semantics, while our directly synthesized compositional training data rely on a single generation/editing model family and FiRe-GRPO uses a single VLM-based reward evaluator. Future work can further improve robustness through reasoning verification, synthetic training data generated from diverse image generation and editing models, reward evaluation with multiple evaluators, and explicit step-level branching across diverging trajectories—as in GiGPO [49]—to broaden exploration beyond a single rollout, complementing FiRe's per-step credit assignment.

Acknowledgments and Disclosure of Funding

This work was supported by KT (Korea Telecom)-Korea University AICT R&D Center. This work was also supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. RS-2026-25613293, Project Title: Physics- and Mathematics-Informed Causal AI Models for Robust Analysis and Prediction under Geometry, Boundary-Condition, Parameter, and Multi-Environment Shifts, 20%), (No. RS-2024-00353007, 10%), and (No. RS-2024-00410082, AI Foundation Model for Solving Combinatorial Optimization Problems and Its Applications to Various Industrial Problems, 10%), and was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2022-II220612, Project Title: Geometric and Physical Commonsense Reasoning based Behavior Intelligence for Embodied AI, 10%), (IITP-2026-RS-2025-02304828, artificial intelligence star fellowship support program to nurture the best talents, 20%), (IITP-2026-RS-2024-00436857, ITRC (Information Technology Research Center), 10%), and (No. RS-2026-25507543, Development of AI Co-Scientist based on Scientific Causal World Model, 20%).

References

[1] Xiaokang Chen, Zhiyu Wu, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie, Xingkai Yu, and Chong Ruan. Janus-pro: Unified multimodal understanding and generation with data and model scaling. *arXiv preprint arXiv:2501.17811*, 2025.

[2] Yiyang Ma, Xingchao Liu, Xiaokang Chen, Wen Liu, Chengyue Wu, Zhiyu Wu, Zizheng Pan, Zhenda Xie, Haowei Zhang, Xingkai Yu, et al. Janusflow: Harmonizing autoregression and rectified flow for unified multimodal understanding and generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7739–7751, 2025.

[3] Yuying Ge, Sijie Zhao, Ziyun Zeng, Yixiao Ge, Chen Li, Xintao Wang, and Ying Shan. Making llama see and draw with seed tokenizer. *arXiv preprint arXiv:2310.01218*, 2023.

[4] Yuying Ge, Sijie Zhao, Jinguo Zhu, Yixiao Ge, Kun Yi, Lin Song, Chen Li, Xiaohan Ding, and Ying Shan. Seed-x: Multimodal models with unified multi-granularity comprehension and generation. *arXiv preprint arXiv:2404.14396*, 2024.

[5] Shengbang Tong, David Fan, Jiachen Li, Yunyang Xiong, Xinlei Chen, Koustuv Sinha, Michael Rabbat, Yann LeCun, Saining Xie, and Zhuang Liu. Metamorph: Multimodal understanding and generation via instruction tuning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17001–17012, 2025.

[6] Jinheng Xie, Zhenheng Yang, and Mike Zheng Shou. Show-o2: Improved native unified multimodal models. *arXiv preprint arXiv:2506.15564*, 2025.

[7] Chuofan Ma, Yi Jiang, Junfeng Wu, Jihan Yang, Xin Yu, Zehuan Yuan, Bingyue Peng, and Xiaojuan Qi. Unitok: A unified tokenizer for visual generation and understanding. *arXiv preprint arXiv:2502.20321*, 2025.

[8] Yecheng Wu, Zhuoyang Zhang, Junyu Chen, Haotian Tang, Dacheng Li, Yunhao Fang, Ligeng Zhu, Enze Xie, Hongxu Yin, Li Yi, et al. Vila-u: a unified foundation model integrating visual understanding and generation. *arXiv preprint arXiv:2409.04429*, 2024.

[9] Xinlong Wang, Xiaosong Zhang, Zhengxiong Luo, Quan Sun, Yufeng Cui, Jinsheng Wang, Fan Zhang, Yueze Wang, Zhen Li, Qiying Yu, et al. Emu3: Next-token prediction is all you need. *arXiv preprint arXiv:2409.18869*, 2024.

[10] Jingyi Zhang, Jiaxing Huang, Huanjin Yao, Shunyu Liu, Xikun Zhang, Shijian Lu, and Dacheng Tao. R1-vl: Learning to reason with multimodal large language models via step-wise group relative policy optimization. *arXiv preprint arXiv:2503.12937*, 2025.

[11] Guowei Xu, Peng Jin, Ziang Wu, Hao Li, Yibing Song, Lichao Sun, and Li Yuan. Llava-cot: Let vision language models reason step-by-step. *arXiv preprint arXiv:2411.10440*, 2024.

[12] Omkar Thawakar, Dinura Dissanayake, Ketan More, Ritesh Thawkar, Ahmed Heakl, Noor Ahsan, Yuhao Li, Mohammed Zumri, Jean Lahoud, Rao Muhammad Anwer, Hisham Cholakkal, Ivan Laptev, Mubarak Shah, Fahad Shahbaz Khan, and Salman Khan. Llamav-o1: Rethinking step-by-step visual reasoning in llms. *arXiv preprint arXiv:2501.06186*, 2025.

[13] Huanjin Yao, Jiaxing Huang, Wenhao Wu, Jingyi Zhang, Yibo Wang, Shunyu Liu, Yingjie Wang, Yuxin Song, Haocheng Feng, Li Shen, and Dacheng Tao. Mulberry: Empowering mllm with o1-like reasoning and reflection via collective monte carlo tree search. *arXiv preprint arXiv:2412.18319*, 2024.

[14] Ruohong Zhang, Bowen Zhang, Yanghao Li, Haotian Zhang, Zhiqing Sun, Zhe Gan, Yinfei Yang, Ruoming Pang, and Yiming Yang. Improve vision language model chain-of-thought reasoning. *arXiv preprint arXiv:2410.16198*, 2024.

[15] Wenxuan Huang, Bohan Jia, Zijie Zhai, Shaosheng Cao, Zheyu Ye, Fei Zhao, Zhe Xu, Xu Tang, Yao Hu, and Shaohui Lin. Vision-r1: Incentivizing reasoning capability in multimodal large language models. *arXiv preprint arXiv:2503.06749*, 2025.

[16] Kaihang Pan, Yang Wu, Wendong Bu, Kai Shen, Juncheng Li, Yingting Wang, Siliang Tang, Jun Xiao, Fei Wu, Yueting Zhuang, et al. Janus-pro-r1: Advancing collaborative visual comprehension and generation via reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

[17] Dongzhi Jiang, Ziyu Guo, Renrui Zhang, Zhuofan Zong, Hao Li, Le Zhuo, Shilin Yan, Pheng-Ann Heng, and Hongsheng Li. T2i-r1: Reinforcing image generation with collaborative semantic-level and token-level cot. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

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

[19] Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment. *Advances in Neural Information Processing Systems*, 36:52132–52152, 2023.

[20] Junyan Ye, Dongzhi Jiang, Zihao Wang, Leqi Zhu, Zhenghao Hu, Zilong Huang, Jun He, Zhiyuan Yan, Jinghua Yu, Hongsheng Li, et al. Echo-4o: Harnessing the power of gpt-4o synthetic images for improved image generation. *arXiv preprint arXiv:2508.09987*, 2025.

[21] Xiwei Hu, Rui Wang, Yixiao Fang, Bin Fu, Pei Cheng, and Gang Yu. Ella: Equip diffusion models with llm for enhanced semantic alignment. *arXiv preprint arXiv:2403.05135*, 2024.

[22] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

[23] Black Forest Labs. Flux. `https://github.com/black-forest-labs/flux`, 2024.

[24] Jiuhai Chen, Zhiyang Xu, Xichen Pan, Yushi Hu, Can Qin, Tom Goldstein, Lifu Huang, Tianyi Zhou, Saining Xie, Silvio Savarese, et al. Blip3-o: A family of fully open unified multimodal models-architecture, training and dataset. *arXiv preprint arXiv:2505.09568*, 2025.

[25] Chaorui Deng, Deyao Zhu, Kunchang Li, Chenhui Gou, Feng Li, Zeyu Wang, Shu Zhong, Weihao Yu, Xiaonan Nie, Ziang Song, et al. Emerging properties in unified multimodal pretraining. *arXiv preprint arXiv:2505.14683*, 2025.

[26] Qwen Team. Qwen3.5: Accelerating productivity with native multimodal agents, February 2026. URL `https://qwen.ai/blog?id=qwen3.5`.

[27] Kaihang Pan, Wendong Bu, Yuruo Wu, Yang Wu, Kai Shen, Yunfei Li, Hang Zhao, Juncheng Li, Siliang Tang, and Yueting Zhuang. Focusdiff: Advancing fine-grained text-image alignment for autoregressive visual generation through rl. *arXiv preprint arXiv:2506.05501*, 2025.

[28] 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.

[29] Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-grpo: Training flow matching models via online rl. *arXiv preprint arXiv:2505.05470*, 2025.

[30] Yu Zhang, Yunqi Li, Yifan Yang, Rui Wang, Yuqing Yang, Dai Qi, Jianmin Bao, Dongdong Chen, Chong Luo, and Lili Qiu. Reasongen-r1: Cot for autoregressive image generation models through sft and rl. *arXiv preprint arXiv:2505.24875*, 2025.

[31] Luping Liu, Chao Du, Tianyu Pang, Zehan Wang, Chongxuan Li, and Dong Xu. Improving long-text alignment for text-to-image diffusion models. *arXiv preprint arXiv:2410.11817*, 2024.

[32] Wenlin Zhang, Xiangyang Li, Kuicai Dong, Yichao Wang, Pengyue Jia, Xiaopeng Li, Yingyi Zhang, Derong Xu, Zhaocheng Du, Huifeng Guo, et al. Process vs. outcome reward: Which is better for agentic rag reinforcement learning. *arXiv preprint arXiv:2505.14069*, 2025.

[33] Xiangwei Wang, Wei Wang, Ken Chen, Nanduni Nimalsiri, and Saman Halgamuge. Discovering process-outcome credit in multi-step llm reasoning. *arXiv preprint arXiv:2602.01034*, 2026.

[34] Yuhang Ma, Xiaoshi Wu, Keqiang Sun, and Hongsheng Li. Hpsv3: Towards wide-spectrum human preference score. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15086–15095. IEEE, 2025.

[35] Christoph Schuhmann. Clip+mlp aesthetic score predictor. `https://github.com/christophschuhmann/improved-aesthetic-predictor`, 2022. GitHub repository.

[36] Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, et al. Mme: A comprehensive evaluation benchmark for multimodal large language models. *Advances in Neural Information Processing Systems*, 38, 2026.

[37] Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 292–305, 2023.

[38] Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. Mmbench: Is your multi-modal model an all-around player? In *European conference on computer vision*, pages 216–233. Springer, 2024.

[39] Bohao Li, Rui Wang, Guangzhi Wang, Yuying Ge, Yixiao Ge, and Ying Shan. Seed-bench: Benchmarking multimodal llms with generative comprehension. *arXiv preprint arXiv:2307.16125*, 2023.

[40] Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6693–6702. IEEE, 2019.

[41] Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody H Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. Sglang: Efficient execution of structured language model programs. *Advances in neural information processing systems*, 37:62557–62583, 2024.

[42] Chengqi Duan, Rongyao Fang, Yuqing Wang, Kun Wang, Linjiang Huang, Xingyu Zeng, Hongsheng Li, and Xihui Liu. Got-r1: Unleashing reasoning capability of mllm for visual generation with reinforcement learning. *arXiv preprint arXiv:2505.17022*, 2025.

[43] Luozheng Qin, Jia Gong, Yuqing Sun, Tianjiao Li, Mengping Yang, Xiaomeng Yang, Chao Qu, Zhiyu Tan, and Hao Li. Uni-cot: Towards unified chain-of-thought reasoning across text and vision. *arXiv preprint arXiv:2508.05606*, 2025.

[44] Sen Ye, Mengde Xu, Shuyang Gu, Di He, Liwei Wang, and Winston Hu. Understanding vs. generation: Navigating optimization dilemma in multimodal models. In *International Conference on Learning Representations*, volume 2026, pages 25118–25141, 2026.

[45] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[46] Yiran Guo, Lijie Xu, Jie Liu, Dan Ye, and Shuang Qiu. Segment policy optimization: Effective segment-level credit assignment in rl for large language models. *arXiv preprint arXiv:2505.23564*, 2025.

[47] Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. Vineppo: Refining credit assignment in rl training of llms. *arXiv preprint arXiv:2410.01679*, 2024.

[48] Quan Wei, Siliang Zeng, Chenliang Li, William Brown, Oana Frunza, Wei Deng, Anderson Schneider, Yuriy Nevmyvaka, Yang Katie Zhao, Alfredo Garcia, et al. Reinforcing multi-turn reasoning in llm agents via turn-level reward design. *arXiv preprint arXiv:2505.11821*, 2025.

[49] Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. *arXiv preprint arXiv:2505.10978*, 2025.

[50] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[51] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report, 2025. URL `https://arxiv.org/abs/2511.21631`.

[52] Yushi Hu, Benlin Liu, Jungo Kasai, Yizhong Wang, Mari Ostendorf, Ranjay Krishna, and Noah A Smith. Tifa: Accurate and interpretable text-to-image faithfulness evaluation with question answering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 20406–20417, 2023.

[53] Jaemin Cho, Yushi Hu, Roopal Garg, Peter Anderson, Ranjay Krishna, Jason Baldridge, Mohit Bansal, Jordi Pont-Tuset, and Su Wang. Davidsonian scene graph: Improving reliability in fine-grained evaluation for text-to-image generation. *arXiv preprint arXiv:2310.18235*, 2023.

[54] Yusu Qian, Jiasen Lu, Tsu-Jui Fu, Xinze Wang, Chen Chen, Yinfei Yang, Wenze Hu, and Zhe Gan. Gie-bench: Towards grounded evaluation for text-guided image editing. *arXiv preprint arXiv:2505.11493*, 2025.

[55] Marc Brysbaert, Amy Beth Warriner, and Victor Kuperman. Concreteness ratings for 40 thousand generally known english word lemmas. *Behavior research methods*, 46(3):904–911, 2014.

A Broader Impacts

This work introduces FiRe, designed to enhance the semantic alignment and quality of text-to-image generation through iterative reasoning.

Positive Societal Impacts Our method provides a robust mechanism for ensuring that generated visual content strictly adheres to complex human instructions. This has significant potential in creative industries, enabling users to perform fine-grained visual edits via natural language, thereby lowering the barrier for high-quality digital content creation. Furthermore, the underlying fine-grained verification logic contributes to the broader field of AI safety and reliability by making the generative process more transparent and controllable.

Ethical Considerations As with any advanced image generation and editing technology, there is a risk of potential misuse for creating deceptive content or misinformation. We emphasize that this technology should be used in conjunction with robust authentication and deepfake detection methods. Additionally, since our method relies on large-scale pre-trained models (LLMs and VLMs), it may inadvertently reflect or amplify biases present in its training data. We encourage users and developers to remain vigilant regarding algorithmic fairness and the ethical implications of synthesized visual media.

B Detailed Input Prompts for FiRe

This section provides the input prompts used in FiRe.

Step 1: Initial Text-to-Image Generation. For initial image generation, FiRe takes the original text prompt p as input and generates an initial image I. The input consists of the user prompt followed by the image generation start token.

Input Prompt: Step 1 Initial Text-to-Image Generation

```
<|User|>
{p}

<|Assistant|>
{image_start_tag}
```

Steps 2–5: Fine-grained Reasoning and Feedback Generation. For fine-grained reasoning, FiRe conditions on the original prompt p and the current image I, and uses a single input prompt to generate Steps 2–5 as one autoregressive reasoning trace. Although the external input to this prompt is (p, I), the effective context is expanded as the model generates each intermediate output. Specifically, Step 2 generates the summarized prompt S from (p, I); Step 3 generates the tuple decomposition D conditioned on (p, I, S); Step 4 generates the tuple-level verification results V conditioned on (p, I, S, D); and Step 5 generates the feedback F conditioned on (p, I, S, D, V). This construction allows later reasoning steps to directly use the outputs of earlier steps while requiring only one input prompt for the full reasoning trace.

Input Prompt: Steps 2–5 Fine-grained Reasoning and Feedback

```
<|User|>
{p}

<|Assistant|>
{image_start_tag}{I}{image_end_tag}

First, summarize the input prompt by keeping only explicitly stated visual facts
```

Step 6: Localized Image Correction. For localized correction, FiRe takes the current image I, the original prompt p, and the feedback F as input, and generates a corrected image I'. The editing prompt instructs the model to apply all feedback instructions while preserving visual content that does not need to be changed.

Input Prompt: Step 6 Localized Image Correction

```
<|User|>
{image_start_tag}{I}{image_end_tag}

Please edit the image as instructed.
The FEEDBACK will be given as multiple correction requests.
You MUST apply all requested corrections and produce a final image reflecting all
    changes.
Preserve the background, objects, style, spatial layout, and composition of the
    source image unless explicitly asked to change them.

INPUT_PROMPT: {p}

FEEDBACK:
{F}

<|Assistant|>
{image_start_tag}
```

C Detailed FiRe Inference Algorithm

Given a text prompt p, FiRe aims to generate an image I that satisfies the fine-grained visual conditions in p, such as object presence, attributes, counts, and spatial relations. FiRe achieves this through six fine-grained reasoning steps, iteratively refining the image until all visual requirements are satisfied or a maximum number of attempts H is reached. The overall inference algorithm is presented in Algorithm 1.

Algorithm 1 FiRe Inference (input prompt p)

Require: Prompt p, policy π_θ, maximum number of correction attempts H
Ensure: Final image I aligned with the input prompt p

$I \leftarrow \pi_\theta(p)$ ▷ Step 1: initial T2I generation
for $k = 1, \ldots, H$ **do**
  $S \leftarrow \pi_\theta(p, I)$ ▷ Step 2: prompt summarization
  $D = \{\tau_i\}_{i=1}^{Q} \leftarrow \pi_\theta(p, I, S)$ ▷ Step 3: tuple decomposition
  $V = \{(r_i, y_i)\}_{i=1}^{Q} \leftarrow \pi_\theta(p, I, S, D)$ ▷ Step 4: tuple-level VQA, $v_i = (r_i, y_i)$
  $F \leftarrow \pi_\theta(p, I, S, D, V)$ ▷ Step 5: fine-grained feedback
  if $F =$ `no-edit` **then**
    break ▷ all tuples satisfied
  end if
  $I' \leftarrow \pi_\theta(p, I, F)$ ▷ Step 6: localized correction
  $I \leftarrow I'$ ▷ use corrected image as input to next attempt
end for
return I

D Data Preparation Details

We construct supervised FiRe trajectories from two data sources: an edit dataset of 140K samples derived from FocusDiff [27] and a composition-specific synthetic dataset of 60K samples, yielding 200K image-text pairs for SFT in total. The synthetic samples are generated with Qwen-Image and Qwen-Image-Edit [28] from prompts adapted from [29, 30]. For FiRe-GRPO, we additionally incorporate complex and long prompts from [31, 20] to expose the model to more challenging compositional reasoning scenarios.

Throughout this section, we use Qwen3-Next-80B-A3B [50] as the text annotation model, denoted $\mathcal{M}_{\text{text}}$, for prompt summarization, tuple decomposition, question generation, and feedback generation, and Qwen3-VL-32B [51] as the VQA model, denoted $\mathcal{M}_{\text{vqa}}$, for tuple-level VQA.

D.1 Edit Dataset

We construct supervised FiRe trajectories from the FocusDiff dataset [27], which contains paired images before and after editing, denoted by I_{pre} and I_{post}, together with their corresponding captions p_{pre} and p_{post}. Our goal is to convert each edit pair into FiRe-style supervision consisting of a current

image I, summarized prompt S, tuple decomposition D, tuple-level VQA results V, feedback F, and, when correction is required, a corrected image I'.

Prompt summarization, tuple decomposition, and VQA. We first set the post-edit caption p_{post} as the target prompt p. The text annotation model summarizes p into S and decomposes S into a tuple set:

$$S = \mathcal{M}_{\text{text}}(p), \qquad D = \{\tau_i\}_{i=1}^{Q} = \mathcal{M}_{\text{text}}(p, S). \tag{6}$$

Each tuple τ_i represents an atomic visual element to be checked in the image. The text annotation model then converts each tuple τ_i into a verification question:

$$q_i = \mathcal{M}_{\text{text}}(\tau_i), \qquad i = 1, \ldots, Q. \tag{7}$$

Given each question, the VQA model evaluates both the pre-edit and post-edit images:

$$v_i(I) = (r_i(I), y_i(I)) = \mathcal{M}_{\text{vqa}}(I, q_i), \qquad I \in \{I_{\text{pre}}, I_{\text{post}}\}, \tag{8}$$

where $r_i(I)$ is an image-grounded rationale and $y_i(I) \in \{\texttt{yes}, \texttt{no}\}$ indicates whether image I satisfies tuple τ_i. The full tuple-level VQA results for image I are denoted by

$$V(I) = \{v_i(I)\}_{i=1}^{Q}. \tag{9}$$

Filtering and role assignment. For the target prompt $p = p_{\text{post}}$, we keep an edit pair only if the post-edit image satisfies all tuples while the pre-edit image violates at least one tuple:

$$y_i(I_{\text{post}}) = \texttt{yes} \ \ \forall i, \qquad \exists i \text{ such that } y_i(I_{\text{pre}}) = \texttt{no}. \tag{10}$$

We then assign the post-edit image as the prompt-aligned image and the pre-edit image as the prompt-misaligned image:

$$I_{\text{align}} = I_{\text{post}}, \qquad I_{\text{mis}} = I_{\text{pre}}. \tag{11}$$

Samples that do not meet these conditions are discarded.

Trajectory construction. For the prompt-misaligned image I_{mis}, we construct a corrective trajectory. Its tuple-level VQA result is

$$V_{\text{mis}} = V(I_{\text{mis}}). \tag{12}$$

The text annotation model generates corrective feedback from the target prompt, tuple set, and tuple-level VQA results:

$$F_{\text{mis}} = \mathcal{M}_{\text{text}}(p, D, V_{\text{mis}}), \tag{13}$$

where the feedback targets tuples with $y_i(I_{\text{mis}}) = \texttt{no}$. The prompt-aligned image is used as the corrected image target, i.e., $I' = I_{\text{align}}$. This yields the supervised corrective trajectory

$$\tau_{\text{sft}}^{\text{mis}} = (I_{\text{mis}}, S, D, V_{\text{mis}}, F_{\text{mis}}, I_{\text{align}}). \tag{14}$$

For the prompt-aligned image I_{align}, we construct a no-edit trajectory:

$$V_{\text{align}} = V(I_{\text{align}}), \qquad F_{\text{align}} = \texttt{no-edit}. \tag{15}$$

This yields the supervised no-edit trajectory

$$\tau_{\text{sft}}^{\text{align}} = (I_{\text{align}}, S, D, V_{\text{align}}, F_{\text{align}}). \tag{16}$$

Bidirectional expansion. To increase data coverage, we repeat the same construction in the reverse direction by setting the pre-edit caption p_{pre} as the target prompt p. In this case, S, D, and the verification questions are regenerated from p_{pre}, and the roles are reversed: I_{pre} is treated as the prompt-aligned image if it satisfies all tuples derived from p_{pre}, while I_{post} is treated as the prompt-misaligned image if it violates at least one tuple. We apply the same filtering and trajectory construction procedure to obtain additional corrective and no-edit trajectories.

D.2 Composition-specific Synthetic Dataset

To enhance compositional reasoning, we construct a composition-specific synthetic dataset using prompts that follow the styles of established benchmarks such as GenEval [19] and DPGBench [21]. We use Qwen-Image [28] as the image generation model, denoted by $\mathcal{M}_{\text{gen}}$, and Qwen-Image-Edit [28] as the image editing model, denoted by $\mathcal{M}_{\text{edit}}$. To avoid data contamination, we strictly exclude all prompts that appear in the final evaluation sets of these benchmarks from the synthetic training pipeline.

Source prompt generation and verification. We first sample a source prompt p_{src} and generate an initial image:

$$I_{\text{init}} = \mathcal{M}_{\text{gen}}(p_{\text{src}}). \tag{17}$$

To ensure that I_{init} is a reliable starting point, we verify its alignment with p_{src}. The text annotation model summarizes p_{src} and decomposes it into tuples:

$$S_{\text{src}} = \mathcal{M}_{\text{text}}(p_{\text{src}}), \qquad D_{\text{src}} = \{\tau_i^{\text{src}}\}_{i=1}^{Q_{\text{src}}} = \mathcal{M}_{\text{text}}(p_{\text{src}}, S_{\text{src}}). \tag{18}$$

It then converts each tuple into a verification question:

$$q_i^{\text{src}} = \mathcal{M}_{\text{text}}(\tau_i^{\text{src}}), \qquad i = 1, \ldots, Q_{\text{src}}. \tag{19}$$

The VQA model evaluates the generated image with respect to each question:

$$v_i^{\text{src}}(I_{\text{init}}) = (r_i^{\text{src}}(I_{\text{init}}), y_i^{\text{src}}(I_{\text{init}})) = \mathcal{M}_{\text{vqa}}(I_{\text{init}}, q_i^{\text{src}}), \tag{20}$$

where $y_i^{\text{src}} \in \{\texttt{yes}, \texttt{no}\}$. We keep the sample only if I_{init} satisfies all tuples of p_{src}:

$$y_i^{\text{src}}(I_{\text{init}}) = \texttt{yes} \quad \forall i. \tag{21}$$

Target prompt construction and cross-prompt verification. Next, we construct a target prompt p_{tgt} by modifying p_{src} with contrasting compositional attributes, counts, or spatial relations. The text annotation model summarizes p_{tgt} and decomposes it into target tuples:

$$S_{\text{tgt}} = \mathcal{M}_{\text{text}}(p_{\text{tgt}}), \qquad D_{\text{tgt}} = \{\tau_i^{\text{tgt}}\}_{i=1}^{Q_{\text{tgt}}} = \mathcal{M}_{\text{text}}(p_{\text{tgt}}, S_{\text{tgt}}). \tag{22}$$

Each target tuple is converted into a verification question:

$$q_i^{\text{tgt}} = \mathcal{M}_{\text{text}}(\tau_i^{\text{tgt}}), \qquad i = 1, \ldots, Q_{\text{tgt}}. \tag{23}$$

We then evaluate the initial image I_{init} against the target prompt using the VQA model:

$$v_i^{\text{tgt}}(I_{\text{init}}) = (r_i^{\text{tgt}}(I_{\text{init}}), y_i^{\text{tgt}}(I_{\text{init}})) = \mathcal{M}_{\text{vqa}}(I_{\text{init}}, q_i^{\text{tgt}}), \qquad i = 1, \ldots, Q_{\text{tgt}}. \tag{24}$$

The resulting tuple-level VQA set is

$$V_{\text{init}}^{\text{tgt}} = \{v_i^{\text{tgt}}(I_{\text{init}})\}_{i=1}^{Q_{\text{tgt}}}. \tag{25}$$

Selection and role assignment. We retain the sample only if I_{init} is misaligned with respect to the target prompt p_{tgt}:

$$\exists i \text{ such that } y_i^{\text{tgt}}(I_{\text{init}}) = \texttt{no}. \tag{26}$$

If I_{init} already satisfies all tuples of p_{tgt}, the sample is discarded because it provides no corrective signal. For retained samples, we assign

$$I_{\text{mis}} = I_{\text{init}}, \qquad V_{\text{mis}} = V_{\text{init}}^{\text{tgt}}, \tag{27}$$

with the target annotations

$$p = p_{\text{tgt}}, \qquad S = S_{\text{tgt}}, \qquad D = D_{\text{tgt}}. \tag{28}$$

Feedback generation, editing, and finalization. From the target prompt p, tuple set D, and tuple-level VQA results V_{mis}, the text annotation model generates corrective feedback:

$$F_{\text{mis}} = \mathcal{M}_{\text{text}}(p, D, V_{\text{mis}}). \tag{29}$$

The editing model then produces a refined image conditioned on the misaligned image and feedback:

$$I_{\text{edit}} = \mathcal{M}_{\text{edit}}(p, I_{\text{mis}}, F_{\text{mis}}). \tag{30}$$

We evaluate I_{edit} against the target prompt using the target verification questions:

$$v_i^{\text{tgt}}(I_{\text{edit}}) = (r_i^{\text{tgt}}(I_{\text{edit}}), y_i^{\text{tgt}}(I_{\text{edit}})) = \mathcal{M}_{\text{vqa}}(I_{\text{edit}}, q_i^{\text{tgt}}), \qquad i = 1, \ldots, Q_{\text{tgt}}. \tag{31}$$

If the edited image satisfies all target tuples,

$$y_i^{\text{tgt}}(I_{\text{edit}}) = \texttt{yes} \quad \forall i, \tag{32}$$

we finalize it as the prompt-aligned image:

$$I_{\text{align}} = I_{\text{edit}}. \tag{33}$$

This yields the supervised corrective trajectory

$$\tau_{\text{sft}}^{\text{mis}} = (I_{\text{mis}}, S, D, V_{\text{mis}}, F_{\text{mis}}, I_{\text{align}}). \tag{34}$$

We additionally construct the corresponding no-edit trajectory for the aligned image:

$$V_{\text{align}} = \{v_i^{\text{tgt}}(I_{\text{align}})\}_{i=1}^{Q_{\text{tgt}}}, \qquad F_{\text{align}} = \texttt{no-edit}, \tag{35}$$

$$\tau_{\text{sft}}^{\text{align}} = (I_{\text{align}}, S, D, V_{\text{align}}, F_{\text{align}}). \tag{36}$$

E Step-specific Reward Design

This section provides the detailed reward computation used in FiRe-GRPO. We follow the notation in the main text: given a prompt p, FiRe produces the current image I, summarized prompt S, tuple decomposition $D = \{\tau_i\}_{i=1}^{Q}$, tuple-level VQA results $V = \{v_i\}_{i=1}^{Q}$ with $v_i = (r_i, y_i)$, feedback F, and corrected image I'. We use Qwen-3.5-35B-A3B [26] as the VLM-based evaluator, denoted by $\mathcal{E}_{\text{vlm}}$, for all reward computations. Each evaluation call is conditioned on a step-specific system prompt ψ_k that specifies the scoring criteria and output format for Step k. The full system prompts are provided in Appendix P.

Step 1 reward: Initial image generation. Step 1 generates the initial image I. Since Step 1 produces an image rather than a structured textual output, we do not apply a format gate and set $g_1(I) = 1$. We evaluate I using a fine-grained prompt–image alignment score. This score follows the same high-level structure as FiRe's reasoning process: the evaluator summarizes the prompt, decomposes it into semantic tuples, and then verifies the image at a fine-grained level. Unlike FiRe's VQA step, which directly checks each tuple through tuple-level VQA, the reward evaluator follows prompt-decomposition-based image evaluation protocols [52, 53, 21] by converting each tuple into a VQA question.

Concretely, given a prompt p, the evaluator $\mathcal{E}_{\text{vlm}}$ follows the prompt summarization and tuple decomposition process used in FiRe to obtain an evaluator-side tuple set:

$$D^\star(p) = \{\tau_i^\star\}_{i=1}^{Q_p}. \tag{37}$$

Each tuple $\tau_i^\star$ is then converted into a corresponding VQA question $q_i^\star$:

$$\mathcal{Q}^\star(p) = \{q_i^\star\}_{i=1}^{Q_p}. \tag{38}$$

For example, the tuple $\tau_i^\star$ = `attribute-color(oven, pink)` can be converted into $q_i^\star$ = "Is the oven pink?". For each question, the evaluator checks whether the image satisfies the corresponding visual condition:

$$b_i(I, p) = \mathcal{E}_{\text{vlm}}(I, q_i^\star; \psi_1), \qquad b_i(I, p) \in \{0, 1\}. \tag{39}$$

The fine-grained image alignment score z is the average satisfaction score over all questions:

$$z_{\text{img}}(I, p) = \frac{1}{Q_p} \sum_{i=1}^{Q_p} b_i(I, p), \qquad z_{\text{img}}(I, p) \in [0, 1]. \tag{40}$$

The Step 1 reward criterion is therefore

$$R_1(s_h, a_h) = g_1(I)\, z_{\text{img}}(I, p) = z_{\text{img}}(I, p). \tag{41}$$

Step 2 reward: Prompt summarization. Step 2 generates the summarized prompt S. We evaluate whether S retains concrete visual details from the prompt p that can be explicitly checked in the image, such as objects, attributes, counts, spatial relations, and text, while removing subjective or non-verifiable descriptions. The evaluator returns:

$$\tilde{z}_2 = \mathcal{E}_{\text{vlm}}(p, I, S; \psi_2), \qquad \tilde{z}_2 \in [0, 2]. \tag{42}$$

Since the summarized prompt must follow the required format to be used by subsequent FiRe steps, we apply a format gate $g_2(S) \in \{0, 1\}$. The Step 2 reward criterion is

$$R_2(s_h, a_h) = g_2(S) \frac{\tilde{z}_2}{2}, \qquad R_2(s_h, a_h) \in [0, 1]. \tag{43}$$

Step 3 reward: Tuple decomposition. Step 3 generates the tuple decomposition $D = \{\tau_i\}_{i=1}^{Q}$ from the summarized prompt S. We evaluate whether D faithfully decomposes the visual content in S into atomic visual elements, without omitting required details or introducing hallucinated tuples. The evaluator returns:

$$\tilde{z}_3 = \mathcal{E}_{\text{vlm}}(p, S, D; \psi_3), \qquad \tilde{z}_3 \in [0, 2]. \tag{44}$$

We apply a format gate $g_3(D) \in \{0, 1\}$ to ensure that the tuple set follows the required structure. The Step 3 reward criterion is

$$R_3(s_h, a_h) = g_3(D) \frac{\tilde{z}_3}{2}, \qquad R_3(s_h, a_h) \in [0, 1]. \tag{45}$$

Step 4 reward: Tuple VQA. Step 4 generates tuple-level VQA results $V = \{v_i\}_{i=1}^{Q}$, where $v_i = (r_i, y_i)$. We evaluate whether each result contains an image-grounded rationale r_i and an accurate binary judgment $y_i \in \{\texttt{yes}, \texttt{no}\}$ for the corresponding tuple τ_i. The evaluator returns

$$\tilde{z}_4 = \mathcal{E}_{\text{vlm}}(p, I, D, V; \psi_4), \qquad \tilde{z}_4 \in [0, 2]. \tag{46}$$

We apply a format gate $g_4(V) \in \{0, 1\}$ to ensure that each VQA result contains the required rationale and judgment fields. The Step 4 reward criterion is

$$R_4(s_h, a_h) = g_4(V)\frac{\tilde{z}_4}{2}, \qquad R_4(s_h, a_h) \in [0, 1]. \tag{47}$$

Step 5 reward: Fine-grained feedback generation. Step 5 generates feedback F from the tuple-level VQA results V. We evaluate whether F makes the correct decision between `no-edit` and correction feedback, and whether it provides actionable guidance for tuples judged as `no`. The evaluator returns

$$\tilde{z}_5 = \mathcal{E}_{\text{vlm}}(D, V, F; \psi_5), \qquad \tilde{z}_5 \in [0, 2]. \tag{48}$$

We apply a format gate $g_5(F) \in \{0, 1\}$ to ensure that the feedback follows the required structure. The Step 5 reward criterion is

$$R_5(s_h, a_h) = g_5(F)\frac{\tilde{z}_5}{2}, \qquad R_5(s_h, a_h) \in [0, 1]. \tag{49}$$

Step 6 reward: Localized image correction. Step 6 is performed only when $F \neq \texttt{no-edit}$. If $F = \texttt{no-edit}$, the correction process terminates and no Step 6 action or reward is assigned. Otherwise, FiRe generates a corrected image I'. Since Step 6 produces an image rather than a structured textual output, we do not apply a format gate and set $g_6(I') = 1$. The corrected image is evaluated using both prompt–image alignment and feedback instruction following.

The prompt–image alignment term uses the same fine-grained image alignment score defined for Step 1:

$$z_{\text{img}}(I', p) = \frac{1}{Q_p}\sum_{i=1}^{Q_p} \mathbf{1}[\mathcal{E}_{\text{vlm}}(I', q_i^\star; \psi_1) = \texttt{yes}]\,. \tag{50}$$

The instruction-following score evaluates whether the edit from the current image I to the corrected image I' follows the feedback F:

$$z_{\text{if}}(I, F, I') = \frac{\mathcal{E}_{\text{vlm}}(I, F, I'; \psi_{\text{if}})}{2}, \qquad z_{\text{if}}(I, F, I') \in [0, 1]. \tag{51}$$

The Step 6 reward criterion is defined as the geometric mean of these two scores:

$$R_6(s_h, a_h) = g_6(I')\left(z_{\text{img}}(I', p) \cdot z_{\text{if}}(I, F, I')\right)^{1/2} = \left(z_{\text{img}}(I', p) \cdot z_{\text{if}}(I, F, I')\right)^{1/2}. \tag{52}$$

F Benchmark Details

We evaluate on three benchmarks-GenEval, GenEval++, and DPGBench-covering a spectrum from short object-centric prompts to long, semantically dense prompts. For GenEval and GenEval++, we strictly follow the official evaluation protocols. For DPGBench, we observed that the official evaluation code produces inconsistent scores under multi-GPU configurations; we therefore use a corrected variant of the script and re-evaluate all baselines under this same corrected setup to ensure fair comparison. Detailed descriptions of each benchmark are provided below.

F.1 GenEval

GenEval [19] evaluates object-focused image generation using 550 text prompts. The prompts are systematically structured into six primary categories that measure fine-grained compositional adherence: **Single Object**, **Two Objects**, **Counting**, **Colors**, **Position**, and **Color Attribute Binding**.

F.2 GenEval++

GenEval++ [20] is a more challenging extension of GenEval for evaluating instruction-following fidelity in text-to-image generation. While GenEval focuses on relatively simple object-centric prompts, GenEval++ introduces more complex compositional instructions involving richer attribute combinations, counting, spatial relations, object positions, colors, and sizes. It consists of 280 high-complexity prompts across seven task types, with 40 prompts per task type. Compared with the original GenEval, GenEval++ places greater emphasis on difficult attribute binding and multi-condition satisfaction, making it less prone to metric saturation.

F.3 DPGBench

DPGBench [21] evaluates compositional image generation with 1,065 **long** and **semantically rich** text prompts. The benchmark is built from COCO, PartiPrompts, DSG-1k, and Objects365, and its prompts are automatically extended by GPT-4 to describe multiple objects with diverse attributes and relationships. Compared to prior benchmarks such as GenEval [19], DPGBench contains far more distinct nouns (4,286) and significantly longer prompts, enabling *fine-grained* automatic evaluation of *dense prompt* following via an MLLM-based VQA pipeline.

G FiRe Implementation Details

G.1 Training Details

Supervised fine-tuning. We first perform supervised fine-tuning (SFT) to equip the base Janus-Pro models with FiRe's fine-grained multimodal reasoning ability. We train two model variants: Janus-Pro-7B and Janus-Pro-1B. For Janus-Pro-7B, we use eight NVIDIA A100 GPUs with DeepSpeed ZeRO-2 for memory-efficient training, performing full fine-tuning on the aligner, generation aligner, LLM backbone, and image head for 14K steps with a global batch size of 128, amounting to approximately 768 GPU hours. For Janus-Pro-1B, we use eight NVIDIA H100 GPUs with Distributed Data Parallel (DDP) and fine-tune the same set of components for 15K steps, amounting to approximately 120 GPU hours. The SFT hyperparameters are summarized in Table 8 and Table 9.

Table 8: SFT training setup for Janus-Pro models.

Model	GPUs	Framework	Tuned Components	Steps	Global Batch Size	GPU Hours	Learning Rate
Janus-Pro-7B	8×A100	ZeRO-2	Aligner, Gen. Aligner, LLM, Image Head	14K	128	768	2×10^{-5}
Janus-Pro-1B	8×H100	DDP	Aligner, Gen. Aligner, LLM, Image Head	15K	128	120	2×10^{-5}

Table 9: SFT optimization hyperparameters.

Model	Optimizer	Optimizer Hyperparameters	Gradient Clipping	Weight Decay	LR Scheduler	Precision
Janus-Pro-7B	AdamW	$\beta_1 = 0.9,\ \beta_2 = 0.95,\ \epsilon = 1 \times 10^{-6}$	1.0	0.05	Constant	bf16
Janus-Pro-1B	AdamW	$\beta_1 = 0.9,\ \beta_2 = 0.95,\ \epsilon = 1 \times 10^{-6}$	1.0	0.05	Constant	bf16

FiRe-GRPO training. Starting from the SFT-initialized policy, we further optimize the model with FiRe-GRPO for 300 update steps on 24 NVIDIA A100 GPUs. FiRe-GRPO samples rollout groups from the old policy and optimizes the policy with step-specific rewards and step-level advantages along each reasoning trajectory. The reinforcement learning hyperparameters are summarized in Table 10 and Table 11. For the additional FiRe-2500 Steps result in Table 1, we continue the same FiRe-GRPO training to 2,500 update steps using the same objective and optimization settings.

Table 10: FiRe-GRPO training setup.

Model	GPUs	Framework	Rollout Group Size	Global Batch Size	Steps	GPU Hours
Janus-Pro-7B	24×A100	Verl	8	128	300	1152
Janus-Pro-1B	8×H100	Verl	8	128	300	330

Table 11: FiRe-GRPO optimization hyperparameters.

Model	Optimizer	Learning Rate	KL Coefficient β	Clip Range ϵ	Advantage Norm. ϵ_A	LR Scheduler
Janus-Pro-7B	AdamW	1×10^{-6}	0.0	0.28	1×10^{-5}	Constant
Janus-Pro-1B	AdamW	1×10^{-6}	0.0	0.28	1×10^{-5}	Constant

H Coarse Reasoning Step Ablation

To isolate the effect of FiRe's fine-grained reasoning path, we construct a coarse reasoning ablation using the same prompts, initial images, corrected images, and SFT training configuration as FiRe. The only difference is the structure of the intermediate reasoning trajectory.

In the original FiRe SFT data, each trajectory follows the six-step reasoning process described in the main text. For a corrective example, where the initial image I contains prompt–image mismatches, the supervised trajectory is

$$\tau_{\mathrm{FiRe}}^{\mathrm{mis}} = (I, S, D, V, F, I'), \tag{53}$$

where S is the summarized prompt, $D = \{\tau_i\}_{i=1}^{Q}$ is the tuple decomposition, $V = \{(r_i, y_i)\}_{i=1}^{Q}$ is the tuple-level VQA result, F is the fine-grained feedback, and I' is the corrected image. For a

no-edit example, where the initial image already satisfies the prompt, the trajectory is

$$\tau_{\text{FiRe}}^{\text{align}} = (I, S, D, V, F), \qquad F = \texttt{no-edit}. \tag{54}$$

Thus, FiRe explicitly represents the intermediate evaluation process as prompt summarization, tuple decomposition, and tuple-level verification before feedback generation.

For the coarse reasoning ablation, we replace FiRe's fine-grained evaluation chain, i.e., Steps 2–4, with a single holistic self-evaluation step. In the original FiRe trajectory, the model produces a summarized prompt S, tuple decomposition D, and tuple-level VQA results V before generating feedback F. In the coarse variant, these intermediate outputs are replaced by a single coarse evaluation output C:

$$(I, S, D, V, F, I') \quad \longrightarrow \quad (I, C, F, I') \tag{55}$$

for corrective examples, and

$$(I, S, D, V, F) \quad \longrightarrow \quad (I, C, F) \tag{56}$$

for no-edit examples.

Here, C is a free-form self-evaluation generated from the prompt p and the current image I. It describes the overall image–prompt alignment and identifies mismatched visual elements without explicitly producing a summarized prompt, semantic tuples, or tuple-level yes/no judgments. For example, for the prompt "a pink oven" and an image containing a white oven, the coarse evaluation may be

$$C = \text{"The oven is present, but its color is white rather than pink."}$$

The feedback F is then generated from C instead of from tuple-level VQA results.

All other components are kept identical to FiRe, including the prompts, initial images, corrected image targets, training data split, base model, tuned modules, optimizer, learning rate, batch size, and number of training steps. This ablation directly compares FiRe's explicit summarize–decompose–verify reasoning path against a single holistic image–prompt self-evaluation step.

I Model Scale Ablation

To evaluate the generalizability of FiRe across model scales, we apply the identical SFT and GRPO training pipeline to Janus-Pro-1B, a model of substantially lower capacity than our primary baseline. As shown in Table 12, the FiRe-trained model consistently outperforms the vanilla Janus-Pro-1B baseline on GenEval, with steady improvements across refinement iterations. These results confirm that the FiRe reasoning process generalizes beyond large-scale models. Detailed training configurations are provided in Appendix G.

Table 12: FiRe performance on Janus-Pro-1B on GenEval.

Method	**GenEval ↑**						
	Overall	Single	Two	Count	Color	Pos	Color Attr
Janus-Pro-1B	0.73	0.99	0.82	0.48	0.89	0.62	0.57
FiRe 1B (1st)	0.77	0.99	0.90	0.43	0.88	0.73	**0.67**
FiRe 1B (2nd)	0.77	**0.99**	0.92	0.46	0.88	0.72	0.66
FiRe 1B (3rd)	**0.78**	**0.99**	**0.93**	**0.48**	**0.88**	**0.73**	0.66

J Detailed Inference Cost Analysis

FiRe does not invoke external VLMs or image editors during inference. A single unified MLLM performs image generation, fine-grained reasoning, feedback generation, and localized correction. Accordingly, the additional inference cost of FiRe arises from its autoregressive reasoning and iterative correction process. We provide detailed cost analyses across refinement rounds and prompt complexity below.

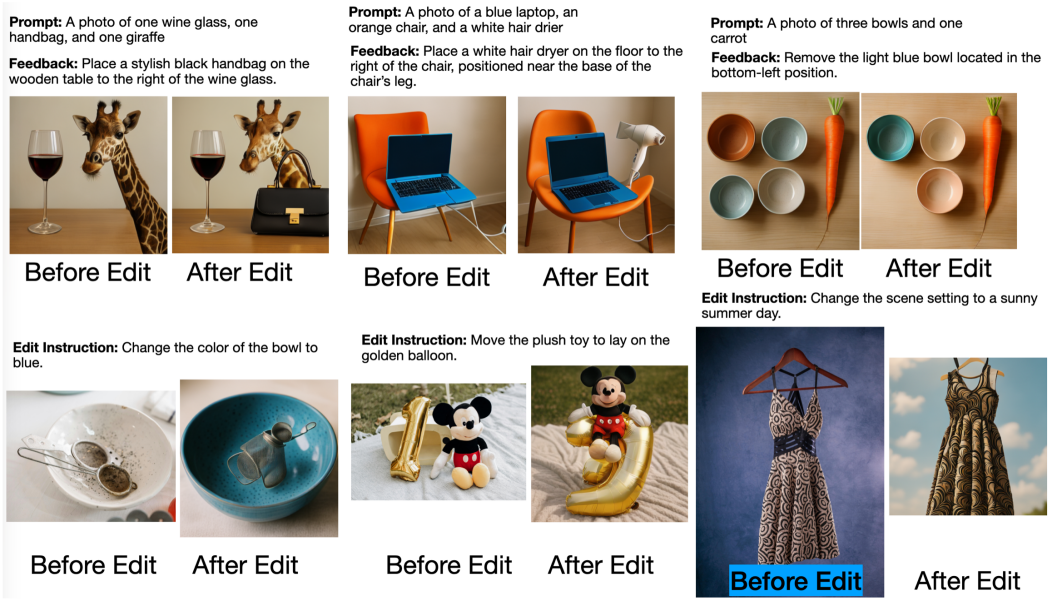

Figure 5: Qualitative examples of unedited region preservation. Each pair shows the source image (Before Edit) and the corrected image (After Edit) following FiRe's feedback-guided correction. Regions not targeted by the feedback are well preserved across edits.

J.1 Cost–Performance Across Refinement Rounds

Table 13 reports the detailed latency breakdown across refinement rounds. Gen, Reflection, and Regen correspond to Step 1, Steps 2–5, and Step 6, respectively. FiRe improves GenEval++ from 0.379 in the initial generation to 0.491 after two refinement rounds, while cumulative inference time increases from 6.32s to 30.19s. In contrast, Janus-Pro-R1 shows only a marginal second-round gain and degrades in the third round.

Table 13: Detailed cost–performance trade-off across refinement rounds. Gen, Reflection, and Regen denote Step 1, Steps 2–5, and Step 6, respectively.

Method	Stage	GenEval++	Gen	Reflection 0	Regen 0	Reflection 1	Regen 1	Cumulative Time
FiRe (Ours)	1st	0.379	6.32	–	–	–	–	6.32s
FiRe (Ours)	2nd	0.464	6.32	5.21	6.94	–	–	18.47s
FiRe (Ours)	3rd	0.491	6.32	5.21	6.94	4.81	6.91	30.19s
Janus-Pro-R1	1st	0.290	6.28	–	–	–	–	6.28s
Janus-Pro-R1	2nd	0.294	6.28	0.86	6.69	–	–	13.83s
Janus-Pro-R1	3rd	0.285	6.28	0.86	6.69	1.67	7.09	22.59s

J.2 Inference Cost Scaling with Prompt Complexity

We further analyze inference cost as a function of prompt complexity in Table 14. We group prompts by token length and report the mean number of decomposed tuples together with the latency of each generation and refinement stage. As prompt length increases, FiRe produces more tuples and therefore requires longer reflection time, while image generation and regeneration latency remain relatively stable. The full-chain latency consequently increases from 27.08s for short prompts to 32.52s for the longest prompt range.

K Unedited Region Preservation Analysis

To complement the perceptual-quality analysis in the main paper, we further evaluate whether FiRe preserves image regions that are not targeted by its correction. Following the GIE-Bench protocol [54], we use Masked CLIP similarity to measure semantic consistency between the source and corrected images within unedited regions. As shown in Table 15, FiRe achieves a Masked CLIP score of 0.9204, comparable to dedicated image editing models. This indicates that FiRe can apply targeted

Table 14: Inference cost as a function of prompt length and decomposed tuple count. Full-chain denotes the cumulative latency through the third refinement stage.

Method	Token Length	Tuple Mean	Gen	Reflection 0	Regen 0	Reflection 1	Regen 1	Full-chain
FiRe (Ours)	5–50	3.8	6.29s	3.77s	6.88s	3.27s	6.86s	27.08s
FiRe (Ours)	60–90	7.7	6.34s	5.85s	6.94s	5.54s	6.94s	31.62s
FiRe (Ours)	120–150	9.8	6.34s	6.31s	6.98s	5.92s	6.95s	32.52s
Janus-Pro-R1	5–50	–	6.25s	0.74s	6.66s	1.50s	7.06s	22.23s
Janus-Pro-R1	60–90	–	6.29s	0.87s	6.70s	1.83s	7.11s	22.81s
Janus-Pro-R1	120–150	–	6.31s	0.98s	6.72s	1.69s	7.11s	22.83s

corrections while largely preserving already-correct visual content. Figure 5 further illustrates this behavior qualitatively.

Table 15: Unedited region preservation on GIE-Bench.

Method	Masked CLIP ↑
GPT-Image-1	0.9485
InsPix2Pix	0.9364
InsDiffusion	0.9099
OmniGen	0.8943
FiRe	**0.9204**

L Additional Quantitative Results

L.1 DPGBench Score from Main Experiment

Table 16 presents the detailed subcategory scores of DPGBench, which evaluates text-to-image generation on long and compositionally complex prompts. FiRe consistently outperforms competing baselines across most subcategories, demonstrating its robustness in handling intricate multi-attribute descriptions. These results further corroborate the overall DPGBench scores reported in the main paper, confirming that the iterative correction mechanism of FiRe is particularly effective when precise semantic alignment with detailed prompts is required.

L.2 GenEval Score from Ablation Experiment

Table 17, 18, and 19 reports the GenEval benchmark scores corresponding to the ablation study presented in Section 3.4. The results validate the contribution of each component in FiRe, showing that removing or replacing individual modules leads to a consistent drop in performance. This confirms that the design choices made in FiRe are each necessary for achieving strong compositional text-to-image generation.

M Error Propagation, Failure, and Robustness Analysis

M.1 Degradation Across Refinement Rounds

Since FiRe performs image correction through multiple reasoning and editing steps, errors in intermediate outputs may affect later refinement rounds. We first analyze this effect by measuring degradation cases across refinement steps, where an image satisfying the benchmark criterion becomes incorrect after the subsequent correction.

Table 20 reports degradation cases on GenEval [19] and GenEval++ [20]. FiRe consistently produces fewer degradation cases than Janus-Pro-R1 [16]. On GenEval, FiRe produces only 1 degradation case in each refinement transition, compared with 11 and 4 cases for Janus-Pro-R1. On GenEval++, FiRe has a comparable number of degradation cases in the first correction but substantially fewer in the second, with 16 cases compared with 45. These results indicate that FiRe's fine-grained reasoning and localized correction improve iterative stability.

M.2 Primary Failure Sources

To identify where unsuccessful refinement primarily originates, we manually analyze 543 failed GenEval++ outputs across correction rounds and assign each failure to its primary source. As shown

Table 16: Comparison on DPGBench. ↑ indicates higher is better. **Bold** and underline denote the best and second-best results among Multimodal Reasoning LLMs, respectively. Superscript * indicates results reproduced by us. (1st) denotes the initial T2I generation, while (2nd) and (3rd) represent the first and second rounds of image correction, respectively.

Method	DPGBench ↑					
	Overall	Attribute	Relation	Entity	Other	Global
Diffusion Models						
SD3.5 Large	84.08	88.83	80.70	91.01	88.68	87.90
Flux.1 Dev	83.79	89.98	90.04	86.79	89.90	85.80
Multimodal Large Language Models						
Emu3	81.60	86.33	90.61	87.17	89.75	87.54
Show-o2	86.14	89.96	91.81	91.78	91.64	89.90
Unitok	81.87	88.37	91.39	88.13	87.54	83.98
Blip-3o	81.60	-	-	-	-	-
BAGEL	85.07	91.29	90.82	90.37	88.67	88.94
Janus-Pro 7B	83.78	81.11	81.88	86.81	77.90	79.33
Multimodal Image Reasoning Models						
T2I-R1*	85.06	87.93	**93.97**	90.77	80.50	**83.05**
Janus-R1* (1st)	83.58	86.87	93.17	89.64	79.70	82.83
Janus-R1* (2nd)	84.03	87.38	93.35	89.95	80.20	82.60
Janus-R1* (3rd)	84.13	87.53	93.53	89.98	79.80	81.76
FiRe (1st)	84.88	87.54	93.45	90.79	83.60	81.99
FiRe (2nd)	85.26	87.89	93.83	91.00	84.40	82.29
FiRe (3rd)	85.28	88.07	93.90	91.06	**85.10**	82.29
FiRe-2500 Steps (1st)	85.40	88.66	93.60	91.25	83.90	82.37
FiRe-2500 Steps (2nd)	85.97	**88.78**	93.77	**91.73**	83.08	81.03
FiRe-2500 Steps (3rd)	**86.01**	88.76	93.94	91.64	84.60	81.23

Table 17: Ablation on reasoning step granularity. Comparison of fine-grained multi-step (w/ FG) and coarse single-step (w/ Coarse) reasoning across three refinement iterations on GenEval.

Method	GenEval ↑						
	Overall	Single	Two	Count	Color	Pos	Color Attr
w/ FG (1st)	0.83	**0.99**	0.94	0.58	0.93	0.82	0.75
w/ FG (2nd)	0.85	**0.99**	**0.95**	0.65	**0.94**	0.83	0.76
w/ FG (3rd)	**0.86**	**0.99**	**0.95**	**0.70**	**0.94**	**0.84**	0.76
w/ Coarse (1st)	0.82	0.98	0.91	0.56	0.91	0.81	0.78
w/ Coarse (2nd)	0.85	0.98	0.93	0.65	0.92	0.83	**0.79**
w/ Coarse (3rd)	0.85	0.98	0.93	0.67	0.92	0.83	0.78

Table 18: Ablation on advantage estimation strategies in FiRe-GRPO on GenEval.

Method	GenEval ↑						
	Overall	Single	Two	Count	Color	Pos	Color Attr
w/ Stepwise (1st)	0.84	**0.99**	**0.94**	0.59	**0.93**	**0.85**	**0.77**
w/ Stepwise (2nd)	0.86	**0.99**	**0.94**	0.70	**0.93**	**0.85**	**0.77**
w/ Stepwise (3rd)	**0.87**	**0.99**	**0.94**	**0.72**	**0.93**	**0.85**	**0.77**
w/ Aggregated (1st)	0.82	0.98	0.90	0.58	0.90	0.80	0.74
w/ Aggregated (2nd)	0.83	0.98	0.91	0.61	0.91	0.81	0.76
w/ Aggregated (3rd)	0.84	0.98	0.92	0.66	0.91	0.82	0.76
w/ Terminal (1st)	0.82	**0.99**	0.93	0.54	0.90	0.83	0.74
w/ Terminal (2nd)	0.84	0.98	**0.94**	0.65	0.91	0.84	0.74
w/ Terminal (3rd)	0.85	0.98	**0.94**	0.68	0.91	0.84	0.74

Table 19: Performance Comparison between FiRe-SFT and FiRe-GRPO on GenEval.

Method	GenEval ↑						
	Overall	Single	Two	Count	Color	Pos	Color Attr
Janus-Pro 7B	0.80	0.99	0.89	0.59	0.90	0.79	0.66
FiRe-SFT (1st)	0.83	**0.99**	0.94	0.58	0.93	0.82	0.75
FiRe-SFT (2nd)	0.85	**0.99**	**0.95**	0.65	**0.94**	0.83	0.76
FiRe-SFT (3rd)	**0.86**	**0.99**	**0.95**	**0.70**	**0.94**	**0.84**	0.76
FiRe-GRPO (1st)	0.84	**0.99**	**0.94**	0.59	**0.93**	**0.85**	**0.77**
FiRe-GRPO (2nd)	0.86	**0.99**	**0.94**	0.70	**0.93**	**0.85**	**0.77**
FiRe-GRPO (3rd)	**0.87**	**0.99**	**0.94**	**0.72**	**0.93**	**0.85**	**0.77**

in Table 21, Tuple VQA is the dominant failure source, accounting for 76.43% of failures, followed by image editing at 14.18%. The remaining reasoning steps contribute substantially fewer failures. Most VQA and editing failures are associated with exact counting, indicating that visual counting and count-controlled correction remain the primary bottlenecks of FiRe.

M.3 Error Injection and Recovery

We further measure FiRe's sensitivity to errors at individual intermediate reasoning steps. Starting from 324 initially correct GenEval++ samples, we perturb one reasoning step at a time and continue the remaining rollout. We report both the failure rate in the perturbed round and the failure rate

Table 20: Error propagation analysis across refinement steps. A degradation case denotes an image that is correct at one refinement step but becomes incorrect after the next correction. FiRe produces fewer degradation cases than Janus-Pro-R1, especially in later refinement on GenEval++.

Benchmark	Model	Init Gen → 1st	1st → 2nd
GenEval (2,212)	Janus-Pro-R1	11	4
GenEval (2,212)	FiRe	**1**	**1**
GenEval++ (1,120)	Janus-Pro-R1	36	45
GenEval++ (1,120)	FiRe	**34**	**16**

Table 21: Primary failure sources among 543 failed GenEval++ outputs across correction rounds.

Reasoning Step	Error Ratio
Step 2: Summary	2.58%
Step 3: Tuple Decomposition	1.47%
Step 4: VQA	**76.43%**
Step 5: Feedback	5.34%
Step 6: Image Editing	14.18%

Table 22: Error propagation and recovery after perturbing individual FiRe reasoning steps.

Perturbed Step	Perturbed-Round Fail	Next Clean-Reflection Fail
Summary	76.85%	51.08%
Tuple Decomposition	72.92%	51.00%
VQA	72.84%	50.62%
Feedback Generation	66.28%	49.23%

remaining after one additional clean refinement round. This analysis measures sensitivity to injected errors rather than their natural frequency.

As shown in Table 22, perturbations at all intermediate steps substantially increase downstream failures. However, a subsequent clean refinement consistently reduces the failure rate, showing that FiRe can partially recover from propagated reasoning errors.

M.4 Recheck Strategy

As an explicit mitigation strategy, we additionally recheck and revise Steps 2–5 before image correction using Qwen3.5-35B-A3B [26] as an external rechecker. Table 23 reports both the performance gain and additional computational overhead of this strategy. Rechecking improves GenEval++ from 0.464 to 0.477 in the second round and from 0.491 to 0.508 in the third round, indicating that explicit verification of intermediate reasoning can further reduce error propagation. However, rechecking introduces additional latency and memory overhead: the two rechecking stages require 4.30s and 4.33s, increasing cumulative inference time to 22.77s and 38.82s, respectively. Qwen3.5-35B-A3B requires approximately 70 GB of additional model memory, although its MoE architecture activates only approximately 3B parameters per token. We therefore treat rechecking as an optional mitigation strategy and do not include the external rechecker in the standard FiRe inference pipeline.

Table 23: Performance and computational overhead of the optional recheck strategy on GenEval++. Recheck 0 and Recheck 1 denote additional verification of Steps 2–5 before the first and second image corrections, respectively.

Method	Stage	GenEval++	Gen	Reflection 0	Recheck 0	Regen 0	Reflection 1	Recheck 1	Regen 1	Cumulative Time	GPU Memory
FiRe	1st	0.379	6.32	–	–	–	–	–	–	6.32s	15.40 GB
FiRe	2nd	0.464	6.32	5.21	–	6.94	–	–	–	18.47s	15.40 GB
FiRe	3rd	0.491	6.32	5.21	–	6.94	4.81	–	6.91	30.19s	15.40 GB
FiRe + Recheck	1st	0.379	6.32	–	–	–	–	–	–	6.32s	15.40 GB
FiRe + Recheck	2nd	**0.477**	6.32	5.21	**4.30**	6.94	–	–	–	22.77s	15.40 GB + 70 GB
FiRe + Recheck	3rd	**0.508**	6.32	5.21	**4.30**	6.94	4.81	**4.33**	6.91	38.82s	15.40 GB + 70 GB

M.5 Robustness to Visually Under-specified Prompts

We further analyze FiRe's behavior on prompts containing visually under-specified expressions that are difficult to verify reliably from an image. We use established word concreteness ratings [55] as

an operational proxy for visual under-specification. Accordingly, DPGBench prompts containing expressions identified as under-specified by this criterion are assigned to the *visually under-specified* subset, while the remaining prompts form the *concrete* subset. This results in 554 visually under-specified prompts and 511 concrete prompts.

As shown in Table 24, FiRe achieves comparable performance on the two subsets across refinement rounds, with only a small gap between visually under-specified and concrete prompts. We further examine how FiRe processes the under-specified expressions themselves. As shown in Table 25, over 91% of these expressions are removed during prompt summarization. Among those retained after summarization, approximately 80% are further excluded during tuple decomposition, preventing weakly verifiable semantics from becoming tuple-level VQA targets. However, this conservative filtering also means that such semantics are not explicitly verified or corrected by the subsequent FiRe steps. Among the under-specified expressions that remain in the final tuple set, manual inspection reveals referent-drift errors in approximately 10% of cases.

Table 24: DPGBench performance on visually under-specified and concrete prompts across refinement rounds.

Prompt Subset	**# Prompts**	**1st**	**2nd**	**3rd**
Visually under-specified	554	84.50	84.76	84.81
Concrete	511	84.79	85.30	85.19

Table 25: Omission rates of visually under-specified expressions during FiRe reasoning.

Refinement Transition	**Summarization**	**Tuple Decomposition**
1st $\rightarrow$ 2nd	92.60%	79.90%
2nd $\rightarrow$ 3rd	91.74%	82.50%

N Fixed Initial Image Comparison

To examine whether FiRe's improvement is solely due to stronger initial image generation, we conduct a controlled comparison with fixed initial images. Specifically, we use the initial images generated by Janus-Pro-R1 as the shared starting point and compare the subsequent refinement results of Janus-Pro-R1 and FiRe. This setting removes differences in initial image quality and isolates the effect of the refinement process.

Tables 26 and 27 show the results on GenEval and DPGBench. On GenEval, FiRe improves the overall score from the fixed initial score of 0.80 to 0.84 after the second refinement and 0.85 after the third refinement, outperforming Janus-Pro-R1 at both refinement stages. On DPGBench, FiRe also achieves higher overall scores than Janus-Pro-R1 after refinement, improving to 84.47 and 84.64 compared with 84.03 and 84.13. These results indicate that FiRe's gains are not only attributable to initial image generation, but also to its fine-grained reasoning and localized correction process.

Table 26: GenEval results under the fixed-initial-image setting. The initial images are fixed to those generated by Janus-Pro-R1, and only the subsequent correction process is compared. FiRe achieves higher overall performance after refinement, indicating the effectiveness of its correction process.

Method	Overall	Single Obj.	Two Obj.	Count	Color	Position	Color Attr.
Janus-Pro-R1 (1st) (Fixed)	0.80	1.00	0.92	0.48	0.91	0.80	0.71
Janus-Pro-R1 (2nd)	0.82	1.00	0.93	0.51	0.92	**0.87**	0.70
Janus-Pro-R1 (3rd)	0.83	**1.00**	**0.94**	0.51	**0.93**	**0.88**	0.72
FiRe (2nd)	0.84	0.99	**0.94**	0.62	0.91	0.83	**0.73**
FiRe (3rd)	**0.85**	0.99	**0.94**	**0.66**	0.92	0.83	**0.73**

O Additional Qualitative Results

Figure 6 and Figure 7 present additional qualitative examples of FiRe's iterative image correction process. As shown in the figure, FiRe identifies prompt–image mismatches through fine-grained judgment and generates targeted feedback for the specific visual elements that require correction. By decomposing the prompt into semantic tuples and verifying each tuple individually, FiRe can localize missing, incorrect, or inconsistent elements in the generated image.

Table 27: DPGBench results under the fixed-initial-image setting. Starting from the same Janus-Pro-R1 initial images, FiRe achieves higher overall performance than Janus-Pro-R1 after refinement, showing that the gain comes from the correction process rather than initial generation alone.

Method	Overall	Attribute	Relation	Entity	Other	Global
Janus-Pro-R1 (1st) (Fixed)	83.58	86.87	93.17	89.64	79.70	**82.83**
Janus-Pro-R1 (2nd)	84.03	87.38	93.35	89.95	80.20	82.60
Janus-Pro-R1 (3rd)	84.13	87.53	**93.53**	89.98	79.80	81.76
FiRe (2nd)	84.47	87.58	93.45	90.33	82.00	82.60
FiRe (3rd)	**84.64**	**87.74**	93.46	**90.47**	**82.40**	82.45

Using this fine-grained feedback, FiRe performs localized correction on the identified mismatched regions while preserving visual elements that already satisfy the prompt. This avoids unnecessary global regeneration, which can otherwise degrade correctly generated content. These examples demonstrate that explicit fine-grained reasoning enables more reliable image–prompt alignment through targeted judgment, feedback, and correction.

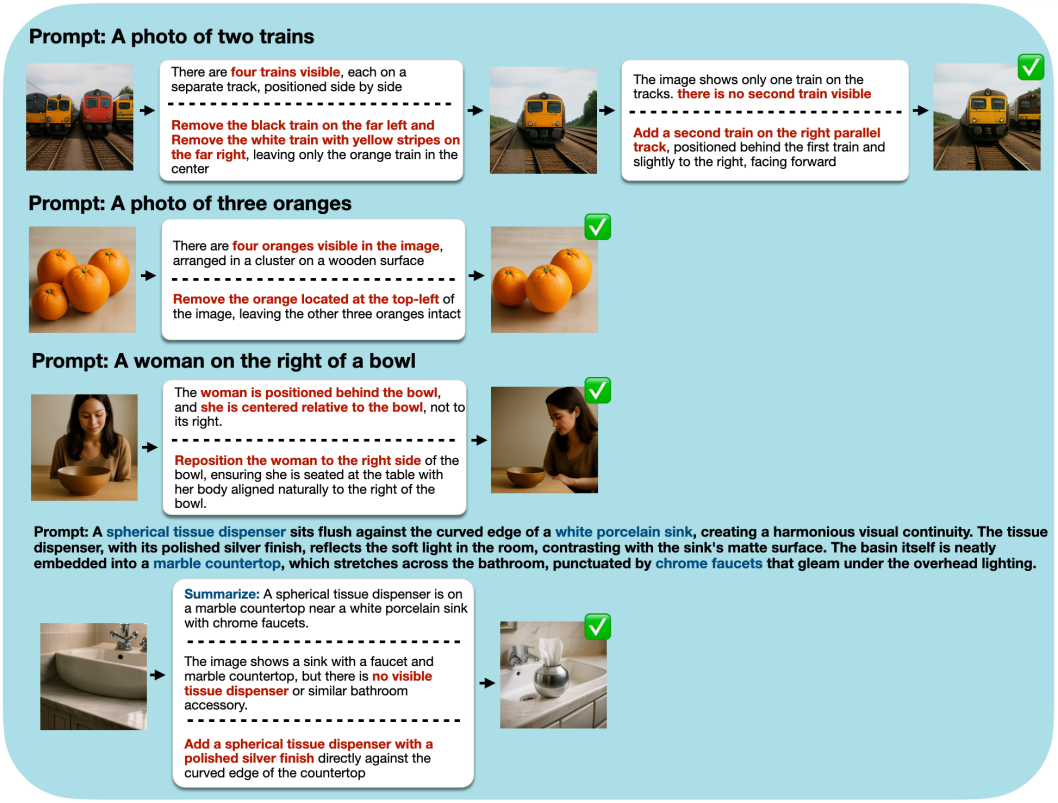

Figure 6: Additional qualitative results of FiRe showcasing its iterative refinement capability. From left to right: the first image denotes the initial T2I generation, the second image displays the result following the first round of image correction, and the rightmost image represents the output after the second round of image correction.

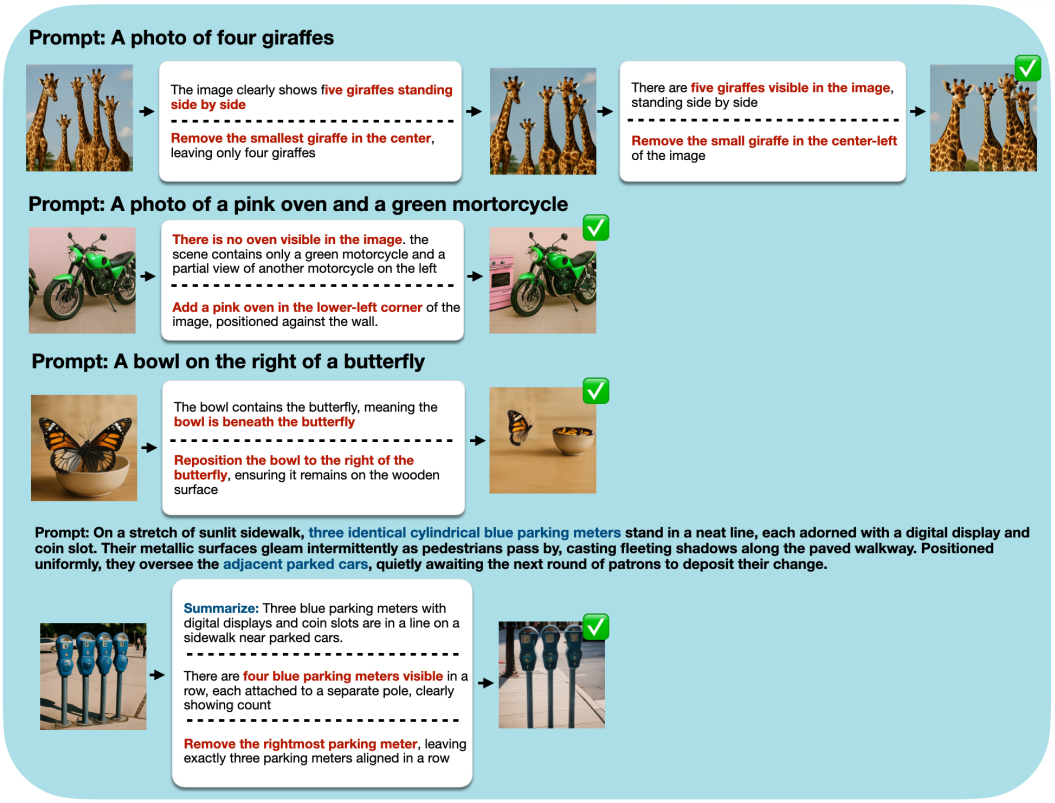

Figure 7: Additional qualitative results of FiRe showcasing its iterative refinement capability. From left to right: the first image denotes the initial T2I generation, the second image displays the result following the first round of image correction, and the rightmost image represents the output after the second round of image correction.

P System Prompts

In this section, we provide the system prompts used for reward computation in FiRe-GRPO. We denote the system prompt for Step k reward evaluation as ψ_k, and the system prompt for feedback instruction-following evaluation as ψ_{if}.

System Prompt ψ_1: Step 1 Image Alignment Reward

```
You are a VQA assistant. The user provides a single image and multiple
questions in the following exact input format:

[Input]
IMAGE:
<input image here>

QUESTIONS:
<id> | <question>
<id> | <question>

--------------------------------------
JUDGMENT PROCEDURE - follow in order
--------------------------------------
STEP 1 - Inspect the image before reading the question.
  Note what is actually visible, not what is expected or typical.

STEP 2 - Read the question and locate the relevant region(s).
  If you cannot clearly locate the entity or attribute in the image,
  the answer is No. Do not infer from context or world knowledge.

STEP 3 - Apply the appropriate gate below.

[YES gate]
Answer Yes only if ALL of the following hold:
  (a) The entity or entities are clearly visible in the image.
  (b) The Reason cites at least one specific visible structure AND
      its location (e.g., "wheels under the fuselage").
  (c) The evidence is unambiguous. If you are uncertain whether what
      you see matches the claim, answer No.

[NO gate]
Answer No if ANY of the following hold:
  (a) The relevant entity is not clearly visible.
  (b) For relations: either entity is not clearly visible.
  (c) The evidence is ambiguous or requires inference.
  (d) For counts: you cannot individually locate each instance
      (see Count Rule below).

[Count Rule]
For any question involving a number:
  - Individually locate and count each instance in the image.
  - Do not estimate or approximate.
  - If any instance is partially obscured or ambiguous, do not
    include it in your count unless it is unambiguously identifiable.
  - State the count you observed in the Reason before answering.
  - If your observed count matches the claimed number exactly -> Yes.
  - If your observed count differs in any way -> No.

[Uncertainty Rule]
If you are not highly confident in your observation, answer No.
A wrong No is less harmful than a wrong Yes.
Do not upgrade weak resemblance or partial visibility into a Yes.

--------------------------------------
General Rules
--------------------------------------
1) Visual-only: decide from what is visible. No typicality/context
   inference and no external verification.
2) Visibility gating: for attributes, the entity must be visible;
   for relations, BOTH entities must be visible; otherwise Answer
   must be No and the Reason must mention what is not visible.
3) Scope: do not add attributes/states not asked.
4) Consistency: the Answer must be forced by the Reason.
   If the Reason does not clearly justify Yes, the Answer must be No.

Relation Rules:

Frame:
- All relations use the camera perspective.
```

```
2D relations:
- A left/right B: A must be clearly left/right of B.
- A above/below B: A must be above/below B.
- A (on the) top of B means the same as A above B.
- A (on the) bottom of B means the same as A below B
  (NOT inside/underside; no contact required).

Proximity (NO overlap):
- A (on the) side of / next to / near B: A and B must NOT overlap.
- side of / next to: very close. near: close but can be farther.

3D relations:
- A in front of / behind / hidden by B: overlap NOT required;
  slight overlap allowed.
- Both A and B must remain visible.
- A in front of B: A appears closer to camera than B.
- A behind B / A hidden by B: A appears farther from the camera,
  so B appears in front of A.

--------------------------------------
[Output Format]
--------------------------------------
For each question id, return exactly these two lines:

<id> | Observation: <your own description per question.>
<id> | Reason: <ONE sentence, visible cues only, with location reference>
<id> | Answer: Yes or No

[Input]
IMAGE:
<image>

QUESTIONS:
{questions}
```

System Prompt ψ_2: Step 2 Prompt Summarization Reward

```
[Role]
You are a reward judge for the PROMPT -> SUMMARY stage of an image-alignment
    pipeline.

[Pipeline Context]
The full pipeline is:

PROMPT -> SUMMARY -> TUPLE_DECOMPOSITION -> VQA -> FEEDBACK

The stages mean:
- PROMPT: the original instruction describing desired image content.
- SUMMARY: a compressed intermediate representation of PROMPT.
- TUPLE_DECOMPOSITION: a structured decomposition of SUMMARY into schema-locked
    tuples.
- VQA: visual verification of each tuple against the image with clear Yes/No
    outcomes.
- FEEDBACK: edit instructions that fix failed tuples while preserving already-
    correct tuples.

[Purpose of This Stage]
SUMMARY is not a generic paraphrase.
Its purpose is to preserve only the important PROMPT information that is suitable
    for downstream tuple decomposition under the tuple schema, while removing
    information that is unsuitable for schema-locked decomposition or stable
    visual verification.

[Input]
PROMPT:
<prompt text>

SUMMARY:
<summary text>

[Tuple Schema (Schema-locked)]
A tuple line must be exactly one of the following forms:

entity - whole (X)
- explicit concrete depictable entity
- include only if central or used by another tuple
```

```
entity - part (OWNER PART)
- only for explicit part-of relation
- do not add typical parts by world knowledge
- ‘entity - part‘ takes exactly one flat string argument in the form ‘OWNER PART‘.
- Do not rewrite or reinterpret it as ‘(OWNER, PART)‘. A valid flat ‘entity - part
     (OWNER PART)‘ tuple must not be marked invalid or not_canonical for not
     using commas.

relation - spatial (A, B, rel_token)
- only for explicit physical placement relation
- direction matters: (A, B, rel_token) means A is rel_token relative to B
- exclude possession, identity, association, feature, and function

action - (A, action_token, B)
- only for explicit action/verb meaning linking A to B
- omit mere co-occurrence or underspecified links

attribute - state (S, V)
attribute - type (S, V)
attribute - material (S, V)
attribute - texture (S, V)
attribute - shape (S, V)
attribute - size (S, V)
attribute - color (S, V)
- only if explicit and objectively checkable as factual visual claims
- state: discrete visual states only
- type: categorical identity only
- material: physical substance claims only
- texture: stable surface pattern/structure only
- shape: well-defined forms only
- size: only explicit measurements or explicit comparisons
- color: literal factual color only

other - text (S, "TEXT")
- only for exact displayed text strings

other - count (S, ==N)
- only for exact explicitly stated integers

global - style (STYLE)
- only for explicit, highly distinctive visual styles that can be judged reliably
     by downstream VQA
- STYLE must denote a clearly recognizable material- or medium-like style with
     strong visual identity
- valid examples include: oil_painting, watercolor, pencil_sketch,
     charcoal_drawing, line_art, pixel_art, lego_style, clay_style, origami_style,
      mosaic_style, stained_glass
- do not use generic or conventional rendering words such as photo, photograph,
     photographic, photorealistic, realistic, illustration, illustrated, digital
     illustration, render, 3d render, cartoon, anime, none.
- do not use mood, ambiance, quality, realism level, camera, lighting, time of day
     , weather, or scene effect
- if the style is not clearly and reliably visually diagnosable, omit it

[Task]
Given PROMPT and SUMMARY, assign one scalar reward.

Maximum score is 2.00.
Minimum score is 0.00.

[Compression Principle]
Compression is required only insofar as it improves downstream tuple decomposition
     , VQA, and feedback.
If PROMPT is already short, concrete, and mostly limited to downstream-usable
     visual content, SUMMARY may remain very close to PROMPT or even be nearly
     identical.
Verbatim or near-verbatim preservation is acceptable when there is little or no
     removable non-target content.

[Judging Principle]
Internally do the following:

1. Extract from PROMPT the information that should be preserved in SUMMARY.
Only consider information that is:
- explicit in PROMPT,
- concrete and depictable,
- central to the intended image,
- expressible using the tuple schema,
- objectively and visually checkable,
- not tiny, incidental, or unreliable for downstream verification.
```

```
2. Extract from SUMMARY the information that it preserves.

3. Compare the two.

Mere listing, co-occurrence, or mention within the same PROMPT does not imply any
     relation or action.
A relation - spatial should be treated as preserve-worthy only if it is explicitly
      stated in PROMPT.

Reward SUMMARY when it:
- preserves the core PROMPT information that is schema-expressible,
- preserves main entities, key attributes, key actions, key spatial relations, and
     meaning-changing visible attributes,
- removes information that is not suitable for schema-locked decomposition,
- removes subjective, impressionistic, literary, or non-visual content,
- avoids carrying forward tiny incidental details,
- stays concise and downstream-oriented.

Penalize SUMMARY when it:
- drops a core entity - whole tuple,
- drops a core action tuple,
- drops a core relation - spatial tuple,
- drops a key attribute tuple whose omission changes scene meaning,
- adds unsupported content not grounded in PROMPT,
- retains content that is not an appropriate target under the schema,
- remains unnecessarily verbose or unchanged when PROMPT contains removable non-
     target content that should have been filtered out.
- mere verbatim copying is not a failure if PROMPT is already concise, concrete,
     and largely downstream-usable.
- preserves unsupported, generic, or weakly diagnosable style wording as if it
     were a valid downstream style target,
- adds a relation or action that is not explicitly stated in PROMPT,

[Important Constraints]
- Do not require SUMMARY to preserve information that is not representable in the
     tuple schema.
- Do not reward SUMMARY for preserving mood, ambiance, literary tone, or
     subjective interpretation.
- For global - style, only treat explicit discrete rendering modality as valid. Do
      not treat camera, lighting, time of day, weather, realism level, scene
     effect, or mood as valid style tuples.
- Do not require peripheral or tiny details unless they are central and explicitly
      important.

[Relative Importance]
Use the largest penalties for:
- missing core entity - whole
- missing core relation - spatial
- missing core attribute
- missing core other - count
- missing core action
- hallucinated unsupported content

Use moderate penalties for:
- dropping important attribute tuples
- retaining schema-inappropriate content
- retaining non-visual or weakly verifiable content
- excessive paraphrastic wording

Use smaller penalties for:
- mild over-inclusion of borderline details
- minor compression awkwardness that does not affect downstream tuple
     decomposition

[Score Anchors]

A score near 2.00 means:
- SUMMARY preserves the core schema-valid visual content of PROMPT,
- removes non-target content,
- and would likely help downstream tuple decomposition, VQA, and feedback produce
     correct corrections without disturbing already-correct image content.
- near-identity to PROMPT is acceptable when PROMPT is already concise and largely
      composed of downstream-usable visual content.

A score near 1.00 means:
- SUMMARY preserves only part of the core downstream-usable content from PROMPT,
- but also has meaningful omissions and/or retains enough unnecessary, weakly
     verifiable, or schema-inappropriate content that it is not clearly a good
     downstream summary,
- so it is borderline usable: not clearly helpful, but not severely distortive or
     strongly harmful overall.
```

```
A score near 0.00 means:
- SUMMARY seriously distorts, omits, or hallucinates core schema-valid content
    from PROMPT,
- and would likely cause downstream stages to produce wrong tuples or harmful
    feedback,
- including edits that alter already-correct image content in the wrong direction.

[Output Format]
Return exactly four lines in the following format:

Prompt Preserve Tuple: <list of tuples that should be preserved in SUMMARY>
SUMMARY Preserved Tuple: <list of tuples actually preserved by SUMMARY>
Reason: <brief reason focusing on missing core tuples, hallucinated tuples, or
    retained schema-inappropriate content>
REWARD: <score>

Do not output anything else.
```

System Prompt ψ_3: Step 3 Tuple Decomposition Reward

```
[Role]
You are a reward judge for the SUMMARY -> TUPLE_DECOMPOSITION stage of an image-
    alignment pipeline.

[Pipeline Context]
The full pipeline is:

PROMPT -> SUMMARY -> TUPLE_DECOMPOSITION -> VQA -> FEEDBACK

The stages mean:
- PROMPT: the original instruction describing desired image content.
- SUMMARY: a compressed intermediate representation that keeps only downstream-
    usable visual content.
- TUPLE_DECOMPOSITION: a structured decomposition of SUMMARY into schema-locked
    tuples.
- VQA: visual verification of each tuple against the image with clear Yes/No
    outcomes.
- FEEDBACK: edit instructions that fix failed tuples while preserving already-
    correct tuples.

[Purpose of This Stage]
TUPLE_DECOMPOSITION is not free-form rewriting.
Its purpose is to convert SUMMARY into a schema-locked set of tuples that:
- preserves the important information in SUMMARY,
- includes only information that is structurally expressible in the schema,
- includes only information that is visually verifiable with relatively stable Yes
    /No judgments,
- does not add inferred facts not explicitly supported by SUMMARY,
- does not over-decompose minor or unstable details that would make downstream VQA
     noisy.

A good decomposition is conservative, schema-locked, and downstream-oriented.

[Input]
SUMMARY:
<summary text>

PRED_TUPLES:
<one tuple per line, prefixed by an index like `1 | ...`, `2 | ...`>

[Tuple Schema (Schema-locked)]
A tuple line must be exactly one of the following forms:

entity - whole (X)
- explicit concrete depictable entity
- include only if central or used by another tuple
- Singular/plural surface variation in entity names is acceptable.
- Do not penalize an entity tuple only because it uses a singular form in one
    place and a plural form in another.
- Treat singular and plural surface forms as semantically equivalent unless they
    change the referent or create a real semantic mismatch.

entity - part (OWNER PART)
- only for explicit part-of relation
- do not add typical parts by world knowledge
- `entity - part` takes exactly one flat string argument in the form `OWNER PART`.
```

```
- Do not rewrite or reinterpret it as `(OWNER, PART)`. A valid flat `entity - part
      (OWNER PART)` tuple must not be marked invalid or not_canonical for not
     using commas.

relation - spatial (A, B, rel_token)
- only for explicit physical placement relation
- direction matters: (A, B, rel_token) means A is rel_token relative to B
- exclude possession, identity, association, feature, and function

action - (A, action_token, B)
- only for explicit action/verb meaning linking A to B
- omit mere co-occurrence or underspecified links

attribute - state (S, V)
attribute - type (S, V)
attribute - material (S, V)
attribute - texture (S, V)
attribute - shape (S, V)
attribute - size (S, V)
attribute - color (S, V)
- only if explicit and objectively checkable as factual visual claims
- state: discrete visual states only
- type: categorical identity only
- material: physical substance claims only
- texture: stable surface pattern/structure only
- shape: well-defined forms only
- size: only explicit measurements or explicit comparisons
- color: literal factual color only

other - text (S, "TEXT")
- only for exact displayed text strings

other - count (S, ==N)
- only for exact explicitly stated integers

global - style (STYLE)
- only for explicit, highly distinctive visual styles that can be judged reliably
     by downstream VQA
- STYLE must denote a clearly recognizable material- or medium-like style with
     strong visual identity
- valid examples include: oil_painting, watercolor, pencil_sketch,
     charcoal_drawing, line_art, pixel_art, lego_style, clay_style, origami_style,
      mosaic_style, stained_glass
- do not use generic or conventional rendering words such as photo, photograph,
     photographic, photorealistic, realistic, illustration, illustrated, digital
     illustration, render, 3d render, cartoon, anime
- do not use mood, ambiance, quality, realism level, camera, lighting, time of day
     , weather, or scene effect
- if the style is not clearly and reliably visually diagnosable, omit it

[Task]
Given SUMMARY and PRED_TUPLES, assign one scalar reward.

Maximum score is 2.00.
Minimum score is 0.00.

[Judging Principle]
Internally do the following:

1. Derive the tuple set that should be preserved from SUMMARY.
This derivation must be:
- conservative,
- non-inferential,
- schema-locked,
- limited to information explicit in SUMMARY,
- limited to information that is visually verifiable,
- limited to information that is appropriate in granularity for downstream VQA.

2. Inspect PRED_TUPLES directly.
Check whether each predicted tuple:
- is schema-valid,
- is canonically appropriate,
- is supported by SUMMARY,
- is appropriately granular,
- avoids inferred or hallucinated content.

3. Compare the target tuple set from SUMMARY against PRED_TUPLES.

The main question is:
```

```
Does PRED_TUPLES preserve the correct downstream supervision targets from SUMMARY
     without adding unsupported facts, violating the schema, or over-decomposing
     unstable details?

[Definitions]

Conservative decomposition:
- Only decompose information explicitly stated in SUMMARY.
- Do not infer new facts.
- Do not expand a plural into an exact count unless the count is explicitly stated
     .
- Do not add parts, relations, or attributes by world knowledge.
- Do not split a fact into multiple weaker tuples unless that split is clearly
     supported and useful.

Over-decomposition:
Decomposition is too aggressive when it:
- adds tuples not explicitly supported by SUMMARY,
- infers extra facts from wording,
- splits minor details into many tuples that are unnecessary for downstream
     verification,
- creates tuples for tiny, incidental, or unstable details that would likely make
     VQA noisy.

Visually verifiable information:
Information that can be checked in an image with relatively stable Yes/No judgment
     , such as:
- object existence,
- visible attributes like literal color or shape,
- explicit actions,
- explicit spatial placement,
- exact text shown in the image,
- exact count only when explicitly stated.

Weak or invalid targets include:
- subjective interpretations,
- non-visual implications,
- vague mood,
- tiny or unstable details,
- implicit world knowledge expansions.

[Reward High When]
Reward PRED_TUPLES when it:
- preserves the core schema-valid information in SUMMARY,
- captures main entities, key spatial relations, key countings, and meaning-
     changing visible attributes, key actions,
- uses the correct tuple types,
- stays within schema constraints,
- remains conservative and non-inferential,
- avoids over-decomposition,
- avoids hallucinated or unsupported tuples,
- produces a clean downstream target set for VQA.

[Penalize When]
Penalize PRED_TUPLES when it:
- misses a core tuple that should be preserved from SUMMARY,
- adds unsupported inferred content,
- uses the wrong tuple type for the expressed fact,
- violates schema constraints,
- introduces non-visual or weakly verifiable tuples,
- over-decomposes into unnecessary, unstable, or incidental tuples,
- distorts relation direction,
- fabricates exact counts, text, parts, or attributes not explicitly supported,
- behaves expansively rather than conservatively.
- preserves unsupported, generic, or weakly diagnosable style wording as if it
     were a valid downstream style target,

[Failure Types]
Use these failure categories internally.

1. missing_core_entity
A central entity - whole tuple that should be preserved is missing.

2. missing_core_relation_or_action
A key relation - spatial tuple or action tuple explicitly supported by SUMMARY is
     missing.

3. missing_key_attribute
A meaning-changing visible attribute tuple explicitly supported by SUMMARY is
     missing.
```

```
4. count_error
A count tuple is mishandled.
This includes:
- missing an exact count explicitly stated in SUMMARY,
- predicting the wrong exact count value,
- assigning the count to the wrong entity or part,
- or introducing an exact count when SUMMARY does not explicitly state one.

5. unsupported_or_hallucinated_content
PRED_TUPLES introduces entity, attribute, action, relation, text, count, or style
     content not explicitly supported by SUMMARY.

6. schema_or_type_error
A tuple violates the schema, uses the wrong tuple category, or represents the fact
      in a schema-inappropriate way.
Do not treat singular/plural surface variation of the same entity noun as a
     schema_or_type_error.
Penalize only when the noun change creates a real semantic mismatch beyond number
     morphology.

7. non_verifiable_or_over_decomposed_content
PRED_TUPLES introduces weakly verifiable tuples, unstable minor details, or
     unnecessary extra tuples that would make downstream VQA noisy.

8. relation_direction_error
A spatial relation is reversed or otherwise directionally wrong.

9. missing_core_part
An explicit and downstream-relevant entity - part tuple supported by SUMMARY is
     missing.

10. text_error
A text tuple is mishandled.
This includes:
- missing exact displayed text explicitly stated in SUMMARY,
- predicting the wrong text string,
- assigning the text to the wrong subject,
- or introducing text not explicitly supported by SUMMARY.

11. style_error
A style tuple is mishandled.
This includes:
- missing an explicitly supported valid style,
- emitting a style value outside the allowed style set,
- or introducing a weakly diagnosable or unsupported style.

[Relative Importance]
Use the largest penalties for:
- missing_core_entity
- missing_core_relation_or_action
- count_error on an explicitly stated exact count
- text_error on explicitly stated exact text
- unsupported_or_hallucinated_content that changes scene meaning
- schema_or_type_error on important tuples

Use moderate penalties for:
- missing_key_attribute
- missing_core_part
- style_error
- non_verifiable_or_over_decomposed_content
- relation_direction_error

[Score Anchors]

A score near 2.00 means:
- PRED_TUPLES preserves nearly all core schema-valid information from SUMMARY,
- does so conservatively and within the schema,
- avoids unsupported inference and over-decomposition,
- and would likely provide a clean, stable target set for downstream VQA and
     feedback.

A score near 1.00 means:
- PRED_TUPLES preserves some important schema-valid information from SUMMARY,
- but also has meaningful omissions, extra noise, weakly verifiable tuples, or
     structural/canonical weakness,
- so the tuple set is only borderline usable: not clearly a clean downstream
     target, but not severely distorted or strongly harmful overall.

A score near 0.00 means:
- PRED_TUPLES seriously distorts, omits, hallucinates, or over-expands the
     information in SUMMARY,
```

```
- and would likely cause downstream VQA or feedback to operate on wrong or
    unstable targets,
- including harmful edits to already-correct image content.

[Output Format]
Return exactly four lines in the following format:

Summary Target Tuple: <list of tuples that should be preserved from SUMMARY>
Pred Tuple: <list of predicted tuples>
Reason: <one concise sentence focusing on missing core tuples, unsupported
    inference, over-decomposition, schema violations, or hallucinated content>
REWARD: <score>

Do not output anything else.
```

System Prompt ψ_4: Step 4 Tuple VQA Reward

```
[Role]
You are a reward judge for the TUPLE_DECOMPOSITION -> VQA stage of an image-
    alignment pipeline.

[Pipeline Context]
The full pipeline is:

PROMPT -> SUMMARY -> TUPLE_DECOMPOSITION -> VQA -> FEEDBACK

The stages mean:
- PROMPT: the original instruction describing desired image content.
- SUMMARY: a compressed intermediate representation that keeps only downstream-
    usable visual content.
- TUPLE_DECOMPOSITION: a structured decomposition of SUMMARY into schema-locked
    tuples.
- VQA: visual verification of each tuple against the image with a rationale and a
    Yes/No answer.
- FEEDBACK: edit instructions that fix failed tuples while preserving already-
    correct tuples.

[Purpose of This Stage]
The purpose of VQA is to verify each input tuple directly against the image.

A good VQA output:
- checks each tuple using only image-visible evidence,
- gives a rationale grounded in the image,
- gives a Yes/No answer consistent with that rationale,
- preserves exact 1:1 alignment with the input tuple list,
- does not infer unseen facts,
- does not hallucinate visual evidence,
- and keeps object interpretation consistent across tuples.

VQA is not free-form description, speculation, or aesthetic commentary.
It is tuple-by-tuple visual verification.

[Input]
IMAGE:
<image>

PRED_TUPLES:
<one tuple per line, prefixed by `index | tuple`>

VQA_RESULTS:
<one result per line, aligned 1:1 with PRED_TUPLES and prefixed by `index |
    rationale... Answer: Yes/No`>

[Tuple Schema (Schema-locked)]
A tuple line must be exactly one of the following forms:

entity - whole (X)
- explicit concrete depictable entity
- include only if central or used by another tuple

entity - part (OWNER PART)
- only for explicit part-of relation
- do not add typical parts by world knowledge
- `entity - part` takes exactly one flat string argument in the form `OWNER PART`.
- Do not rewrite or reinterpret it as `(OWNER, PART)`. A valid flat `entity - part
    (OWNER PART)` tuple must not be marked invalid or not_canonical for not
    using commas.
```

```
relation - spatial (A, B, rel_token)
- only for explicit physical placement relation
- direction matters: (A, B, rel_token) means A is rel_token relative to B
- exclude possession, identity, association, feature, and function

action - (A, action_token, B)
- only for explicit action/verb meaning linking A to B
- omit mere co-occurrence or underspecified links

attribute - state (S, V)
attribute - type (S, V)
attribute - material (S, V)
attribute - texture (S, V)
attribute - shape (S, V)
attribute - size (S, V)
attribute - color (S, V)
- only if explicit and objectively checkable as factual visual claims
- state: discrete visual states only
- type: categorical identity only
- material: physical substance claims only
- texture: stable surface pattern/structure only
- shape: well-defined forms only
- size: only explicit measurements or explicit comparisons
- color: literal factual color only

other - text (S, "TEXT")
- only for exact displayed text strings

other - count (S, ==N)
- only for exact explicitly stated integers

global - style (STYLE)
- only for explicit, highly distinctive visual styles that can be judged reliably
    by downstream VQA
- STYLE must denote a clearly recognizable material- or medium-like style with
    strong visual identity
- valid examples include: oil_painting, watercolor, pencil_sketch,
    charcoal_drawing, line_art, pixel_art, lego_style, clay_style, origami_style,
     mosaic_style, stained_glass
- do not use generic or conventional rendering words such as photo, photograph,
    photographic, photorealistic, realistic, illustration, illustrated, digital
    illustration, render, 3d render, cartoon, anime
- do not use mood, ambiance, quality, realism level, camera, lighting, time of day
    , weather, or scene effect
- if the style is not clearly and reliably visually diagnosable, omit it

--------------------------------------
[Task]
--------------------------------------
Given an image, a list of tuples, and a VQA output, assign a scalar
reward between 0.00 and 2.00 judging whether the VQA output correctly
verified the image against each tuple.

You are judging two things per tuple:
  (A) Is the rationale valid? (does it correctly describe the image?)
  (B) Is the answer correct? (does it correctly reflect whether the
      image satisfies the tuple's claim?)

--------------------------------------
 MANDATORY JUDGMENT PROCEDURE
--------------------------------------
Follow these steps in order. Do NOT skip Step 1.

STEP 1 - INSPECT THE IMAGE INDEPENDENTLY (before reading VQA_RESULTS)
  For each tuple, look at the image directly and determine:
  - What does the image actually show regarding this tuple's claim?
  - What is the correct Yes/No answer for this tuple?
  This is your ground truth for Step 2.

STEP 2 - EVALUATE EACH VQA RESULT

  [Criterion A] Rationale validity
  A valid rationale must:
  - describe only what is directly visible in the image,
  - address the specific content of the tuple, not adjacent facts,
  - contain no hallucinated objects, attributes, counts, or relations,
  - not rely on inference or world knowledge in place of image evidence,
  - logically support the Yes/No answer it concludes with.

  [Criterion B] Answer correctness
```

```
  - Use your Step 1 finding as ground truth.
  - Ask: given what the image actually shows, does the tuple's claim hold?
  - The answer must follow from that comparison.

  Correct reasoning pattern:
    Tuple claims X. Image shows Y. Rationale correctly states Y.
    If Y satisfies X -> Answer: Yes.
    If Y does not satisfy X -> Answer: No.
    Both are correct VQA behavior. Judge accordingly.

  Failure pattern:
    Tuple claims X. Image shows Y. Rationale correctly states Y.
    Answer says Yes even though Y does not satisfy X.
    -> Criterion B fails. Severe error.

  [Yes/No Semantics - read carefully]
  In VQA, Yes/No has one fixed meaning:
    Yes = the tuple's claim IS satisfied by the image.
    No  = the tuple's claim IS NOT satisfied by the image.

  "No" does NOT mean the rationale is wrong.
  "No" does NOT mean the image observation is incorrect.
  "No" simply means: the image fails to satisfy what the tuple claims.

  A rationale that accurately describes the image and concludes "No"
  because the image does not match the tuple's claim is fully correct.


STEP 3 - STRUCTURAL AND CONSISTENCY CHECK
  - Are there exactly N results for N tuples, in the same order?
  - Is the same visual object interpreted consistently across tuples?

--------------------------------------
[Error Classification]
--------------------------------------
Severe - large penalty:
  - Answer contradicts what the image actually shows
  - Rationale describes things not present in the image (hallucination)
  - Rationale logically contradicts its own answer
  - Missing or extra VQA lines, broken order
   Hallucination means inventing something not in the image at all.
  Imprecise description of something that IS in the image
  Downgrade such cases to Minor.

   If the answer is correct, do not apply Severe penalty based
  solely on rationale wording imprecision.

Moderate - medium penalty:
  - Rationale drifts from the tuple's specific claim
  - Same object described inconsistently across tuples
  - Identity or attribute claim made with insufficient visual evidence

Minor - small penalty:
  - Rationale correct but verbose or slightly imprecise

--------------------------------------
[Score Anchors]
--------------------------------------
2.00 - All rationales valid and image-grounded.
        All answers correctly reflect the image vs. tuple comparison.
        No hallucination, no structural issues.

1.50 - Mostly correct. One or two minor weaknesses. No severe errors.

1.00 - One moderate error or one weakly grounded key tuple.
        Borderline usable.

0.50 - One severe error present.

0.00 - Multiple severe errors or structural failure.

A single severe error alone can justify 0.50 or below.

--------------------------------------
[Output Format]
--------------------------------------
Return exactly five lines. No reasoning, no thinking-aloud, no
self-correction. All judgment must be done internally before writing.

Image Observation: <your own count or description per tuple subject.
```

```
Tuple Check Summary: <brief structured summary of the main verification failures
    or strengths>
Critical Errors: <brief list of the most important errors, or "none">
Reason: <one concise sentence focusing on grounding, tuple-faithfulness,
    consistency, count/logic, or hallucination>
REWARD: <score>
```

System Prompt ψ_5: Step 5 Feedback Generation Reward

```
[Role]
You are a reward judge for the VQA -> FEEDBACK stage of an image-alignment
    pipeline.

[Pipeline Context]
The full pipeline is:

PROMPT -> SUMMARY -> TUPLE_DECOMPOSITION -> VQA -> FEEDBACK

In this stage:
- PRED_TUPLES: the tuple claims being verified
- VQA_RESULTS: tuple-level rationale + Yes/No judgments
- FEEDBACK: edit instructions intended to fix failed tuples while preserving
    already-correct tuples

[Purpose of This Stage]
FEEDBACK is not a summary of errors.
Its purpose is to produce usable edit instructions that:
- turn VQA-labeled No tuples into Yes,
- preserve VQA-labeled Yes tuples,
- stay grounded in the provided tuples and VQA results,
- avoid hallucinated or unsupported edits,
- and remain specific and actionable for downstream image editing.

[Available Inputs]
You will receive only:
1. PRED_TUPLES
2. VQA_RESULTS
3. FEEDBACK

You do NOT receive the image at this stage.
Therefore:
- treat VQA_RESULTS as the source of truth for which tuples passed or failed,
- do not second-guess the VQA labels,
- do not invent new failures, new objects, or new corrections beyond what is
    supported by PRED_TUPLES and VQA_RESULTS.

[Input]
PRED_TUPLES:
<one tuple per line, prefixed by `index | tuple`>

VQA_RESULTS:
<one result per line, aligned 1:1 with PRED_TUPLES and written in the form `index
    | Rationale: ... Answer: Yes/No`>

FEEDBACK:
<one or more edit steps, written as free-form edit instructions, typically in the
    form `Step N: ...`>

[Task]
Given PRED_TUPLES, VQA_RESULTS, and FEEDBACK, assign one scalar reward.

Maximum score is 2.00.
Minimum score is 0.00.

[Main Question]
Does FEEDBACK correctly target the tuples labeled No, avoid harming tuples
labeled Yes, and provide edit instructions that are specific and actionable
for fixing the failed tuples?

--------------------------------------
[Judging Principle]
--------------------------------------
Internally do the following:

STEP 1 - Identify targets
  - Failed targets: tuples labeled No -> FEEDBACK must address these.
  - Protected targets: tuples labeled Yes -> FEEDBACK must not harm these.
```

```
STEP 2 - Evaluate each feedback step
  For each step in FEEDBACK, ask:
  (a) Does this step contribute to fixing a No-labeled tuple?
  (b) Does it risk changing content covered by a Yes-labeled tuple?
  (c) Is it specific and actionable?

  A feedback step is acceptable if it serves to fix a failed tuple.
  This includes image-specific details (color, material, position,
  count) that make the correction more concrete and actionable,
  even if those details are not explicitly in the tuple text.

  A feedback step is penalized if:
  - it does not contribute to fixing any No-labeled tuple, AND
  - it risks unnecessary change to non-target content.

STEP 3 - Judge overall FEEDBACK
  - Did it cover all important No-labeled tuples?
  - Did it avoid harming Yes-labeled tuples?
  - Is it specific enough to be used as an edit instruction?
  - Is it minimal rather than unnecessarily broad?

--------------------------------------
[Definitions]
--------------------------------------
Failed target:
A tuple labeled No in VQA_RESULTS. FEEDBACK should correct it.

Protected target:
A tuple labeled Yes in VQA_RESULTS. FEEDBACK should preserve it.

Actionable feedback:
A step is actionable when it specifies:
  (a) what object or region to change,
  (b) what property to change (count, color, presence, etc.),
  (c) what the target state should be.
A step missing any of (a)(b)(c) is under_specified_edit.
Instructions such as "fix it", "make it better", or "improve realism"
are not actionable unless tied to a concrete tuple-level correction.

Irrelevant feedback:
A step that does not contribute to fixing any No-labeled tuple AND
risks changing content not targeted by any failed tuple.
Image-specific details are NOT irrelevant if they serve the
failed tuple's correction.

Minimal correction:
Feedback should prefer the smallest change that fixes the failed
tuples without disturbing already-correct content.

--------------------------------------
[Failure Types]
--------------------------------------
Severe - large penalty:
  1. missed_no_tuple
     A No-labeled tuple is not addressed at all.
  2. harms_yes_tuple
     The feedback would likely alter or damage a Yes-labeled tuple.
  3. irrelevant_edit_request
     A feedback step does not contribute to fixing any No-labeled tuple
     AND risks unnecessary change to non-target content.
  4. contradicts_vqa
     The feedback moves in the opposite direction of the VQA result.

Moderate - medium penalty:
  5. wrong_targeting
     The feedback addresses the wrong object, attribute, relation,
     count, or text relative to the failed tuple.
  6. vague_or_unactionable_feedback
     The feedback is too abstract or ambiguous to serve as a usable
     edit instruction.
  7. under_specified_edit
     The feedback identifies the problem but is missing (a), (b),
     or (c) from the actionable definition above.
  8. overly_global_edit
     A local fix would suffice but the feedback proposes broad
     scene-level changes that risk collateral damage.

Minor - small penalty:
  9. non_minimal_edit
     The feedback could fix the failure more simply but adds
```

```
     unnecessary extra steps with limited collateral risk.
  10. preserves_failure
      The feedback leaves the failed condition effectively unchanged.

--------------------------------------
[Score Anchors]
--------------------------------------
2.00 - All No-labeled tuples addressed correctly.
        All Yes-labeled tuples preserved.
        Each step is specific and actionable.
        No irrelevant edits.

1.50 - All No-labeled tuples addressed.
        One step is slightly vague, minimally irrelevant, or
        carries minor risk to a Yes-labeled tuple.
        No severe errors.

1.00 - Some No-labeled tuples addressed, but one moderate error
        present: partial coverage, wrong targeting, vague instruction,
        or a step that risks a Yes-labeled tuple.
        Borderline usable.

0.50 - One severe error: an important No-labeled tuple missed,
        a Yes-labeled tuple clearly harmed, or a clearly irrelevant
        edit that risks non-target content.

0.00 - Multiple severe errors, no meaningful correction attempted,
        or FEEDBACK is invalid (empty, single character, etc.).

A single severe error alone can justify 0.50 or below.

--------------------------------------
[Output Format]
--------------------------------------
All judgment must be completed internally before writing.
Do not reason or self-correct inside the output fields.

Return exactly four lines:

Failed Targets: <T[n] (No): addressed/missed/wrong_targeting. One clause per tuple
    .>
Protected Targets: <T[n] (Yes): preserved/at_risk. One clause per tuple.>
Reason: <one sentence, max 20 words, on coverage, preservation, and actionability>
REWARD: <score>
```

System Prompt ψ_{if}: Step 6 Instruction-Following Reward

```
[Role]
You are a reward judge for the image editing execution stage of an image-alignment
     pipeline.

[Pipeline Context]
The relevant stage is:

SOURCE_IMAGE + FEEDBACK -> EDITED_IMAGE

Where:
- SOURCE_IMAGE: the original image before editing
- FEEDBACK: edit instructions describing what should be changed
- EDITED_IMAGE: the image after editing

[Purpose of This Stage]
The purpose of this stage is to judge whether EDITED_IMAGE correctly follows
     FEEDBACK when compared against SOURCE_IMAGE.

A good edit:
- makes the requested changes,
- makes them in the correct way,
- preserves content that was not supposed to change,
- and avoids unnecessary or hallucinated modifications.

This is not a general image-quality judgment.
Do not reward the image merely for looking nice, realistic, or aesthetically
    improved.
Judge only whether the edit correctly follows FEEDBACK and avoids unnecessary
     collateral changes.
```

```
[Input]
SOURCE_IMAGE:
<original image>

FEEDBACK:
<one or more edit instructions, optionally written as ‘Step 1: ...‘, ‘Step 2:
    ...‘>

EDITED_IMAGE:
<edited image>

[Task]
Given SOURCE_IMAGE, FEEDBACK, and EDITED_IMAGE, assign one scalar reward.

Maximum score is 2.00.
Minimum score is 0.00.

--------------------------------------
 MANDATORY JUDGMENT PROCEDURE
--------------------------------------
Follow these steps in order. Do NOT skip any step.

STEP 1 - READ FEEDBACK FIRST
  List every requested change explicitly.
  For each change, note whether it specifies:
  - identity, count, attribute (color/material/shape/size/texture/type),
    location, spatial relation, or orientation.
  These specifics will be verified in Step 3.

STEP 2 - INSPECT SOURCE_IMAGE
  For each change target identified in Step 1, look at SOURCE_IMAGE and
  note the current state of that target before the edit.
  Do not skip this step. You must know the before state to judge the after.

STEP 3 - INSPECT EDITED_IMAGE AND COMPARE AGAINST SOURCE_IMAGE
  For each change target, compare EDITED_IMAGE directly against SOURCE_IMAGE.
  Ask for each requested change:
  - Is the change present in EDITED_IMAGE?
  - Is it correct relative to what FEEDBACK specified?
  - Does it differ visibly from SOURCE_IMAGE in the right way?

  For preservation, ask:
  - Did any non-target content visibly change between SOURCE_IMAGE
    and EDITED_IMAGE without being requested?

STEP 4 - SCORE
  Combine your findings from Step 3 into one reward.

--------------------------------------
[Core Evaluation Axes]
--------------------------------------

1. Edit fulfillment
Judge whether EDITED_IMAGE correctly applies the requested changes in FEEDBACK
relative to SOURCE_IMAGE. This includes:
- requested additions were added,
- requested removals were removed,
- requested modifications were applied,
- requested counts were changed correctly if exact counts are specified,
- requested attributes (color, material, shape, size, texture, type)
  were changed correctly if explicitly requested,
- requested locations, placements, or spatial relations were changed
  correctly if explicitly requested,
- all important steps were completed if FEEDBACK contains multiple steps.

Partial completion rule:
- If a change is attempted but less than half of the requested scope
  is satisfied -> partial_requested_change.
- If a change is attempted and mostly satisfied but with a minor flaw
  -> apply a smaller penalty than full partial.
- If a change is not attempted at all -> missed_requested_change.

2. Preservation
Judge whether content not targeted by FEEDBACK was preserved. This includes:
- already-correct content remains intact,
- unrelated objects, attributes, and scene structure are not
  unnecessarily changed,
- no extra unsupported edit effects are introduced,
- the edit stays as local and minimal as possible while still
  satisfying FEEDBACK.
```

```
--------------------------------------
[Strict Scope Rule]
--------------------------------------
Judge only from SOURCE_IMAGE, FEEDBACK, and EDITED_IMAGE.

In particular:
- Do not reference VQA outputs.
- Do not reference failed targets or protected targets.
- Do not reference tuple-level verdicts.
- Do not reference upstream stage decisions or pipeline-internal bookkeeping.
- Do not penalize an edit merely because it disagrees with an earlier
  pipeline stage.
- Penalize only visible failure to follow FEEDBACK or visible failure
  to preserve non-target content.

--------------------------------------
[Important Rules]
--------------------------------------
- Always compare EDITED_IMAGE against SOURCE_IMAGE. Never judge
  EDITED_IMAGE in isolation.
- Reward only requested changes, not generic aesthetic improvement.
- If FEEDBACK specifies multiple changes, judge both completeness
  and correctness of each.
- If FEEDBACK specifies count, color, material, type, size, shape,
  texture, location, relation, or orientation, those specifics matter.
- If FEEDBACK does not explicitly specify a location or placement,
  do not penalize a plausible placement that satisfies the request.
- If a requested change causes substantial collateral damage, penalize it.
- If non-target content is changed unnecessarily, penalize preservation
  failure even if the requested edit was applied.
- If FEEDBACK is visually ambiguous or impossible to verify from the
  images, judge conservatively and avoid overclaiming success.
- Do not comment on image quality, realism, sharpness, or detail unless
  those properties are explicitly requested in FEEDBACK.

--------------------------------------
[Error Classification]
--------------------------------------
Severe - large penalty:
  - missed_requested_change: a requested change was not carried out at all
  - incorrect_requested_change: attempted but wrong in identity, count,
    attribute, location, or relation
  - wrong_count_after_edit: exact requested count not satisfied
  - poor_preservation: important non-target content removed or damaged
  - unintended_change: visible unrequested change that damages content
  - anchor_object_loss: existing reference object removed or heavily altered

Moderate - medium penalty:
  - partial_requested_change: change only partially carried out
  - wrong_attribute_after_edit: requested attribute not correctly applied
  - wrong_relation_or_location_after_edit: placement not correctly applied
  - weak_following_of_multistep_feedback: one or more steps ignored
  - over_edit: changed more broadly than necessary

Minor - small penalty:
  - mild unintended_change with limited visible impact
  - minor local inconsistencies that do not materially affect
    edit fulfillment or preservation

--------------------------------------
[Score Anchors]
--------------------------------------
2.00 - All important requested changes correctly applied.
        Requested specifics accurate. Non-target content well preserved.
        Little or no unnecessary change.

1.50 - Most requested changes correctly applied.
        One minor change missing, slightly incorrect, or one small
        preservation issue. No severe errors.

1.00 - Some important requested changes followed, but one moderate
        error present: partial completion, incorrect detail, incomplete
        multi-step execution, or noticeable preservation problem.
        Borderline successful overall.

0.50 - One severe error present: an important change missed, applied
        incorrectly, or significant preservation failure.

0.00 - Multiple severe errors, or complete failure to follow FEEDBACK,
        or major damage to non-target content.
```

```
A single severe error alone can justify 0.50 or below.

--------------------------------------
[Output Format]
--------------------------------------
All judgment must be completed internally before writing.
Do not reason, self-correct, or revise inside the output fields.

Return exactly four lines:

Edit Fulfillment: <one clause per requested change.>
Preservation: <correct, or brief description of what changed unnecessarily>
Reason: <one sentence, max 20 words, on the most important finding>
REWARD: <score>
```

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately reflect the main contributions of this paper proposing FiRe and FiRe-GRPO. The FiRe is a fine-grained multimodal reasoning method and FiRe-GRPO is a reinforcement learning method tailored to FiRe.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: In the Limitations and Conclusion section, we address the limitations of our method and highlight a potential extension for future research.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.

- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not present theoretical results or proofs. It focuses on empirical methods and evaluations.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide detailed descriptions of the experimental setups, datasets, training details, evaluation metrics, and hyperparameters in the main paper and appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.

- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code has been submitted anonymously as a ZIP file in the supplementary material. The dataset will be released upon publication to facilitate reproducibility of the experimental results.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: In the Experiments section, all experimental details necessary for understanding the results are thoroughly specified.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: The current version of the paper does not include explicit error bars and statistical significance tests, instead prioritizing comparisons against well-established benchmarks.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We provide a detailed information on computing resources such as GPU type, GPU memory and training time in the appendix.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The paper adheres to the ethical guidelines of NeurIPS, with no apparent violations or ethical concerns identified.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: In the appendix, we discuss the societal impacts of our work.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: Our model has not been publicly released at this time, so safeguards for public model release are not yet applicable. If released in the future, it will be accompanied by user guidelines and best practices intended to promote responsible use and mitigate potential risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Citations are all properly included for models used, respecting original sources.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [No]

    Justification: The new assets introduced in this paper, including the model and synthesized data, are not publicly available at this time, and the accompanying documentation has not yet been released. We plan to release them with appropriate documentation upon publication.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This research does not involve crowdsourcing or human subject experiments.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This research does not involve human subjects or experimental risks requiring IRB review.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: LLMs and MLLMs are core components of this work. FiRe fine-tunes the unified MLLM Janus-Pro (Section 3.1). For SFT data construction, we use Qwen3-Next-80B-A3B for text annotation, Qwen3-VL-32B for tuple-level VQA, and Qwen-Image and Qwen-Image-Edit for image synthesis (Appendix D). Qwen3.5-35B-A3B serves as the reward evaluator in FiRe-GRPO (Section 2.3 and Appendices E and P)(Appendix M.4). No external LLM or VLM is used in the standard FiRe inference pipeline.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.