VisionCreator-R1: A Reflection-Enhanced Native Visual-Generation Agentic Model

Jinxiang Lai[1*], **Wenzhe Zhao**[2*], **Zexin Lu**[2*], **Hualei Zhang**[1], **Qinyu Yang**[2], **Rongwei Quan**[2], **Zhimin Li**[2], **Shuai Shao**[2], **Song Guo**[1§], **Qinglin Lu**[2]
[1]Hong Kong University of Science and Technology, [2]Tencent Hunyuan
[*]Equal contribution, [§]Corresponding author: songguo@cse.ust.hk

Abstract

Visual content generation has moved from single-image to multi-image workflows, yet existing agents are largely plan-driven and lack a systematic reflection mechanism for correcting mid-trajectory visual errors. To address this gap, we propose **VisionCreator-R1**, a native visual generation agent with explicit reflection, paired with a **Reflection-Plan Co-Optimization (RPCO)** training methodology. Through extensive experiments and trajectory-level analysis, we uncover a **reflection-plan optimization asymmetry** in reinforcement learning (RL): planning can be reliably optimized via plan rewards, while reflection learning is held back by noisy credit assignment. Motivated by this finding, we further abstract a general **Decouple-then-Fuse** paradigm for co-optimizing capabilities with asymmetric reward variance, of which RPCO is the visual-generation instantiation. Our RPCO first trains on the self-constructed **VCR-SFT** dataset, which covers reflection-strong single-image trajectories and planning-strong multi-image trajectories, and then co-optimizes on the **VCR-RL** dataset via RL. This yields our unified VisionCreator-R1 agent, which beats Gemini2.5Pro on existing benchmarks and on our **VCR-Bench** covering single-image and multi-image tasks.

1 Introduction

AI-assisted visual generation has moved rapidly from single-image generation [1–4] to complex multi-image and video workflows [5–9]. Yet workflow-specific agents [5–7] rely on hand-crafted pipelines tailored to narrow domains, and struggle to adapt when task structure or tools change. Workflow-guided agents [8, 9] call external tools through carefully designed system prompts and coordination logic, but this hard-coded control flow rules out end-to-end optimization of understanding, planning, and generation. This inevitably stops the seamless alignment of complex user instructions with precise visual outputs, and prevents the system from learning globally effective policies.

Motivated by the need for a unified and differentiable architecture, a recent native visual generation agent named VisionCreator [10] packs Understanding, Thinking, Planning, and Creation (UTPC) into a single trainable framework, and beats "VLM + tools" systems on multi-step visual generation tasks. Yet these agents are still *plan-driven*: designed system prompts or training signals put the rationality of plans and tool calls first. This exclusive focus on procedural correctness rules out a retrospective mechanism for *structured reflection and self-correction*. As a result, minor deviations in early stages propagate unchecked, and lead to severe error accumulation in long-horizon, multi-image workflows. By *reflection* we mean the agent's ability to (i) check intermediate visual outputs against multi-dimensional checkpoints (e.g., subject, style, attribute, text), and (ii) trigger targeted corrective actions (e.g., re-generation or editing). Each reflection decision can be labeled by a VLM judge, as **Good-Reflection** (correct accept/reject), **Under-Reflection** (accepting a flawed image), or **Over-Reflection** (rejecting an acceptable one). Good-Reflection percentage is the main quality metric. As

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

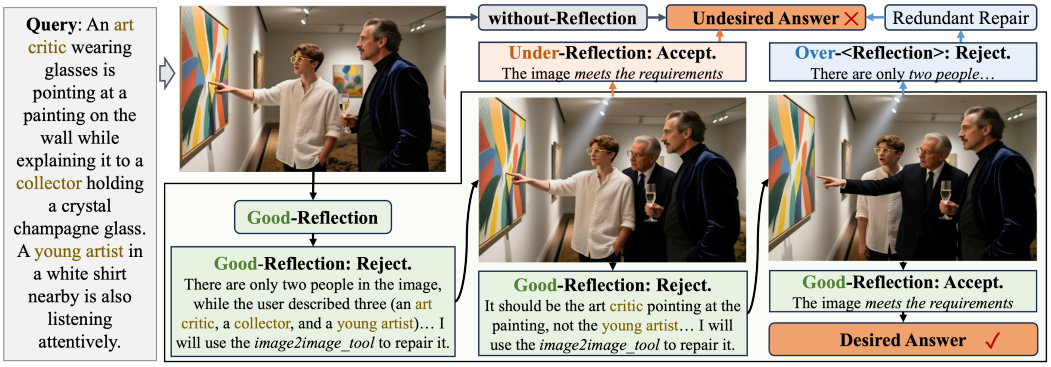


Figure 1: Without-Reflection, with Good-Reflection, Under-Reflection and Over-Reflection.

shown in Fig. 1, Good-Reflection outputs a Desired Answer, while without/Under/Over-Reflection outputs an Undesired Answer, which then causes error accumulation in later trajectories.

Recent works such as JarvisEvo [11] and ReasonEdit [12] show that adding reflection into *single-image editing* can improve visual quality. But these settings involve short-horizon tasks with minimal planning requirements. In contrast, native visual agents must handle diverse workloads spanning single-image and multi-image, all of which rely heavily on robust planning. This raises a new challenge: how to *co-optimize planning and reflection* within a native visual generation agent to improve long-horizon behavior. In this paper, we systematically study reflection as a trainable behavior inside a native visual generation agent. Our study is organized as follows:

(i) UTPCR agentic framework with reward design. We extend the plan-driven UTPC into a **UTPCR** loop (Understanding–Thinking–Planning–Creation–**Reflection**, Sec. 3) that treats reflection as an explicit, trainable stage interleaved with generation. To supervise this loop, we design a multi-dimensional reward covering reflection, planning, format, tool use, and result correctness.

(ii) Reflection-Plan variance-asymmetry finding. Inside UTPCR framework, we found that naïvely extending reflection RL from single-image to multi-image tasks fails. Through trajectory-level analysis (Sec. 4), we identify a structural **reflection-plan variance asymmetry**: plan rewards are evaluated deterministically, while reflection rewards depend on stochastic downstream diffusion outcomes, and this collapses the signal-to-noise ratio (SNR).

(iii) RPCO methodology. Guided by the above finding, we propose **Reflection-Plan Co-Optimization (RPCO)** method (Sec. 5), whose core principle is *Decouple-then-Fuse*: (a) *Decouple* reflection learning in a low-noise single-image environment to bring the SNR back to a usable level; (b) *Fuse* reflection-strong and planning-strong trajectories via SFT on our **VCR-SFT** dataset, which transfers the learned capability without noisy RL gradients; (c) *Co-Optimize* planning and reflection jointly via multi-task RL on **VCR-RL** dataset.

In short, our contributions are as follows: (i) **Reflection-Plan variance-asymmetry finding**: We uncover why single-image reflection RL fails to generalize to multi-image tasks, theoretically and empirically attributing it to a structural reward-variance asymmetry between deterministic planning and stochastic reflection. (ii) **RPCO methodology**: We instantiate Decouple-then-Fuse as RPCO to co-optimize reflection and planning. The resulting **VisionCreator-R1** beats Gemini2.5Pro on VCR-Bench, GEdit-Bench [13], and MultiBanana [14]. (iii) **Open-source infrastructure**: We will release VCR-SFT, VCR-RL, VCR-Bench, and the UTPCR framework to support future research on reflection-aware visual generation.

2 Related Works

Modern image generators [15–20, 1–3, 21, 4] offer strong single-step text-to-image and image-to-image capabilities but do not support task decomposition, tool orchestration, or structured reflection; we treat them as callable tools. Visual generation agents generally fall into three categories: **workflow-specific** [5–7], **workflow-guided** [8, 9, 22, 23], and **native** [10]; the former two rely on hand-crafted pipelines or prompt-based control flow that rule out end-to-end optimization. Closest to us, the native

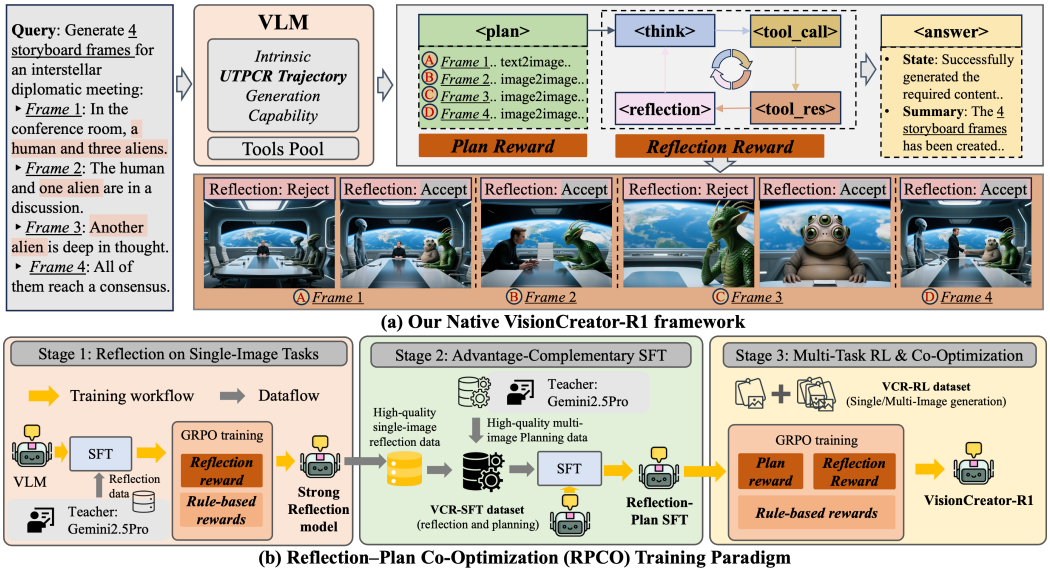


Figure 2: (a) Our Native VisionCreator-R1 framework. (b) RPCO Training Paradigm.

VisionCreator packs UTPC into a single trainable framework but is still *plan-driven* – it supervises only plan rationality and tool validity, with no explicit *reflection* path to check intermediate images and trigger self-correction. Our VisionCreator-R1 builds on this line of work and studies how to *co-optimize* reflection and planning.

3 Agent Framework and Reward Designs

3.1 UTPCR Agentic Framework

As illustrated in Fig. 2 (a), we build the UTPCR framework as a dynamic loop of Act–Reflect–Think–Act. Unlike traditional feed-forward pipelines, our agent adds an explicit *reflection path* to enable self-correction. In this loop, the agent uses its visual understanding ability to: (i) check intermediate visual results for quality and consistency, (ii) spot deviations from the user's instructions and prior context, and (iii) form corrective plans that trigger targeted edits or re-generation, so that the trajectory aligns with the desired outcome.

3.2 Multi-dimensional Reward Design

We define the multi-dimensional rewards used to supervise UTPCR trajectories.

Reflection Reward. We split each user request into K evaluation checkpoints $\{q_1, \ldots, q_K\}$ (e.g., subject, style, attribute, scene, action, text); a VLM judge (Qwen3VL32B) outputs binary decisions $c_k \in \{\text{accept}, \text{refuse}\}$ on the final accepted image set $\mathcal{I}$, and R_{reflect} is the satisfied-checkpoint ratio: $R_{\text{reflect}} = \frac{1}{K}\sum_{k=1}^{K} \mathbb{1}_{(c_k=\text{accept})}$.

Plan Reward. Given a user requirement Q and an agent plan P, an external LLM evaluator $f_{\text{eval}}(Q, P) \in \{1, \ldots, N\}$ scores requirement completeness, logical coherence and tool–goal matching, which gives $R_{\text{plan}} = f_{\text{eval}}(Q, P)/N$. This is the main signal for long-horizon, reference-aware planning in multi-image settings.

Rule-based Rewards. We also use (i) **Format Reward** $R_{\text{format}} \in [0, 1]$, which checks UTPCR tag presence/uniqueness/order/non-emptiness; (ii) **Tool Call Reward** $R_{\text{tool}} \in \{0, 0.1, 0.8, 1\}$, graded by tool-call success and self-correction; and (iii) **Result Reward** $R_{\text{result}} \in \{0, 1\}$, which indicates whether the number of generated images matches the request.

Total Reward. For a trajectory, the overall reward is a weighted average $R_{\text{total}} = \frac{1}{|W|}\sum_{i \in W} w_i R_i$, where W is the set of active reward dimensions (varying by training stage), w_i defaults to 1, and

each $R_i \in [0, 1]$. In Stage 1 of RPCO (Sec. 5), $W_1 = \{\text{reflect, format, tool, result}\}$ since planning is trivial under single-image tasks; in Stage 3, $W_3 = W_1 \cup \{\text{plan}\}$ to drive joint optimization.

4 Empirical and Analytical Foundation

This section presents the empirical and analytical findings that motivate our methodology. We show that reflection is learnable in single-image tasks (Sec. 4.1), that direct transfer to multi-image fails (Sec. 4.2), and give a theoretical explanation via a variance decomposition together with an empirically verified structural-asymmetry claim (Sec. 4.3).

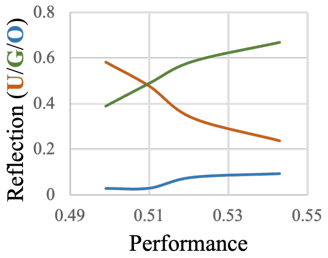


Figure 3: Performance-Reflection.

4.1 Reflection is Learnable on Single-Image Tasks

To cleanly study reflection separate from complex planning, we start with single-image tasks. First, we construct UTPCR trajectories by querying Gemini2.5Pro. Second, we perform SFT on this corpus to endow the agent with explicit reflection structures and basic reflection behaviors in the UTPCR trajectory. Finally, we add RL driven by the visual reflection reward R_{reflect} defined in Sec. 3.2 to further improve how reflection helps final visual quality. Through this two-stage SFT+RL training, we obtain a *Strong-Reflection model* that beats Gemini2.5Pro on single-image tasks, which gives clear evidence that our reflection mechanism and reward design are effective.

Single-Image Reflection Experiments. Fig. 3 shows that raising the **Good-Reflection** percentage leads to better final visual generation. And Tab. 1 lists the detailed experiments on the single-image task in VCR-Bench. Several observations follow: (i) Merely adding a reflection structure is not enough to guarantee performance gains. Gemini2.5Pro and Qwen3VL32B show only marginal improvements over the baseline. (ii) Reflection learned purely via SFT (SFT-Reflection model) leads to degraded performance and a strong bias toward under-reflection, which means the model becomes overly conservative. (iii) After adding RL with the proposed R_{reflect}, both the single-img performance and the share of good-reflection rise steadily from Reflection-s30 to Strong-Reflection (i.e., Reflection-s180), where the suffix "s#" denotes the number of GRPO training steps. Meanwhile, under-reflection drops a lot, and over-reflection stays at a moderate level. This quantitatively confirms that for single-image tasks, the visual reflection reward successfully drives grounded reflection, and directly leads to tangible gains in visual fidelity. This shows that reflection is learnable when trained in isolation. We next ask whether this capability transfers to multi-image settings.

Table 1: Reflection experiment on Single-Img task in VCR-Bench. Ref. Quality (U/G/O): reflection quality with under-/good-/over-reflection percentages. And all models' plan score on the single-image task is larger than 99.5%.

Model	Single-Img	Ref. Quality (U/G/O)
Qwen-Image-Fast	0.505	–
Gemini2.5Pro	0.515	32.0%/55.4%/12.6%
Qwen3VL32B	0.500	41.3%/49.7%/9.0%
SFT-Reflection	0.499	58.2%/38.9%/2.9%
Reflection-s30	0.510	47.7%/48.9%/3.3%
Reflection-s130	0.521	33.7%/58.5%/7.8%
Strong-Reflection	**0.543**	23.7%/**66.9%**/9.4%

4.2 Direct Transfer Fails on Multi-Image Tasks

Direct transfer from single-image reflection to multi-image via RL. We first check whether a reflection-optimized model trained on single-image tasks can be directly extended to multi-image settings. Starting from *Strong-Reflection* model, which gets the highest single-image performance, we further run RL training on the multi-image dataset, and obtain the *Reflection-Plan Conflict* model. Multi-image tasks ask the agent to reason over longer horizons, build reference images, keep cross-image consistency, and follow structured narratives. However, as shown in Tab. 2, direct

RL transfer yields good improvements in multi-image performance (0.617), but still underperforms strong planners such as Gemini2.5Pro (0.649).

Trajectory-level analysis of reflection and planning. Compared with Strong-Reflection model, the Reflection-Plan Conflict model shows: (i) only marginal improvement in plan score (0.8431 vs. 0.8285), and (ii) a further drop in reflection quality, with the share of high-quality reflections dropping from 21.6% to 16.5%. This shows that single-image reflection capabilities cannot be directly transferred to multi-image workflows via RL. While this approach improves planning performance, it leads to a drop in reflection quality. To explain why direct transfer fails, we next provide a theoretical analysis with empirical verification (Sec. 4.3).

Table 2: Transfer from single-image reflection to multi-image tasks. Model: SR is Strong-Reflection model, RPC is Reflection-Plan Conflict model, GPro is Gemini2.5Pro.

Model	Multi-Img	Ref. Quality (U/G/O)	Plan Score
SR	0.565	68.6%/21.6%/9.8%	0.8285
RPC	0.617	73.9%/16.5%/9.7%	0.8431
GPro	0.649	61.6%/31.8%/6.6%	0.9667

4.3 Theoretical Analysis

We now formalize the structural variance gap that explains the failure in Sec. 4.2. We first state a general gradient-variance decomposition (Theorem 4.1), then specialize it to contrast planning and reflection (Proposition 4.2).

Theorem 4.1 (Gradient Variance Decomposition in GRPO) *Consider the stochastic gradient estimator $\hat{g}$ of GRPO at a single token step, with state $s = (q, o_{i,<t})$, sampled action $a = o_{i,t}$, normalized trajectory-level advantage $\hat{A}_{i,t}$, current policy π_θ, reference policy π_{ref}, and KL coefficient β (full form and proof in Appendix C). Its total variance admits the decomposition*

$$\Sigma := \mathrm{Var}(\hat{g}) = \underbrace{\mathbb{E}_{s,a}\left[\mathrm{Var}_{\tau|s,a}(\hat{g})\right]}_{\Sigma_\tau} + \underbrace{\mathbb{E}_s\left[\mathrm{Var}_{a|s}\left(\mathbb{E}_{\tau|s,a}[\hat{g}]\right)\right]}_{\Sigma_a}, \tag{1}$$

where τ denotes the stochastic trajectory, Σ_τ captures the variance induced by environmental dynamics and reward stochasticity (conditioned on state and action), and Σ_a captures the variance induced by stochastic action sampling (the useful learning signal).

Proposition 4.2 (Structural Variance Asymmetry between Planning and Reflection) *Under the decomposition of Theorem 4.1, a fundamental **structural variance asymmetry** holds between planning and reflection objectives in multi-image GRPO. (i) **Planning**: R_{plan} is evaluated deterministically given the action (plan text), so $\mathrm{Var}_{\tau|s,a}(\hat{A}_{i,t}) \approx 0$ and hence $\Sigma_\tau \approx 0$; the gradient variance is dominated by the useful signal Σ_a, which gives a stable optimization regime. (ii) **Reflection**: R_{reflect} in multi-image generation depends on stochastic downstream visual outcomes, so $\mathrm{Var}_{\tau|s,a}(\hat{A}_{i,t}) \gg 0$ and $\Sigma_\tau \gg \Sigma_a$; the optimization suffers a collapsed signal-to-noise ratio (SNR). We empirically verify this with the Reflection-Plan Conflict model's GRPO rollouts (G=4 per prompt): $\mathrm{Var}(R_{\textit{reflect}})/\mathrm{Var}(R_{\textit{plan}})$ grows from $4.8\times$ at training step 10 to $15.5\times$ at step 50, which confirms that the asymmetry is a structural property that* worsens *as training deepens (full protocol and per-step table in Appendix D).*

Implication for RPCO. The asymmetry makes direct RL transfer of reflection from single- to multi-image infeasible, and motivates the Decouple-then-Fuse paradigm in Sec. 5.1: reflection must be learned in a low-noise environment (single-image) and then transferred via SFT rather than via noisy RL gradients. A detailed intuitive derivation is deferred to Appendix E.

5 RPCO Methodology

5.1 The Decouple-then-Fuse Paradigm

Motivated by Proposition 4.2, we abstract a general training paradigm for co-optimizing capabilities with asymmetric variance. Consider an agent that must acquire M capabilities $\{C_1, \ldots, C_M\}$, each with reward R_m. Under RL, the gradient variance for C_m decomposes (by Theorem 4.1) as

$\Sigma_m = \Sigma_{\tau,m} + \Sigma_{a,m}$. The problem shows up when capabilities have *heterogeneous noise levels*: for some C_m, $\Sigma_{\tau,m} \gg \Sigma_{a,m}$ (reward dominated by environmental randomness), while another capability $C_{m'}$ has nearly clean rewards ($\Sigma_{\tau,m'} \approx 0$). Naïve joint RL then causes *capability imbalance*: $C_{m'}$ improves while C_m stagnates or degrades.

Decouple-then-Fuse formulation. Given a base model π_0, a full environment E, and for each high-noise C_m a simplified environment E_m^{simple} where $\Sigma_{\tau,m}$ is reduced, the paradigm runs as:

$$\text{Stage 1 (Decouple):} \quad \pi_m^\star \leftarrow \mathrm{RL}\big(\pi_0;\, R_m,\, E_m^{\text{simple}}\big),\ \text{for each high-noise } C_m, \tag{2}$$

$$\text{Stage 2 (Fuse):} \quad \pi_{\text{fused}} \leftarrow \mathrm{SFT}\Big(\pi_0;\, \bigcup\nolimits_m \mathrm{Filter}(\mathrm{Traj}(\pi_m^\star))\Big), \tag{3}$$

$$\text{Stage 3 (Co-Optimize):} \quad \pi^\star \leftarrow \mathrm{RL}\big(\pi_{\text{fused}};\, \{R_1, \ldots, R_M\},\, E\big). \tag{4}$$

Decouple brings the signal-to-noise ratio (SNR) back for each high-noise capability by reducing $\Sigma_{\tau,m}$ through a simpler environment. *Fuse* transfers the learned capabilities via supervised learning, which avoids high-variance RL gradients. *Co-Optimize* then only fine-tunes from a balanced prior rather than learning from scratch, and needs far fewer exposures to high-variance regimes. A step-by-step pseudocode is deferred to Algorithm 1 (Appendix F); applicability beyond visual generation (e.g., reasoning + tool use, dialogue + retrieval) is discussed in Appendix I.

5.2 RPCO: Decouple-then-Fuse for Reflection-Plan Co-Optimization

We instantiate the Decouple-then-Fuse paradigm for $M = 2$ capabilities in visual generation: reflection (high-noise, $\Sigma_\tau \gg \Sigma_a$) and planning (deterministic, $\Sigma_\tau \approx 0$). The simplified environment E_m^{simple} for reflection is the single-image task (one generation step $\Rightarrow$ low diffusion noise). Concretely, we use base model π_0 (Qwen3VL32B), single-image queries Q_s, multi-image queries Q_m, a planning expert E_{plan} (Gemini2.5Pro), a VLM judge J, and an LLM evaluator f_{eval}, with active reward sets $W_1 = \{\text{reflect, format, tool, result}\}$ and $W_3 = W_1 \cup \{\text{plan}\}$ (Sec. 3.2), where $\mathrm{Traj}(\cdot,\cdot)$ rolls out UTPCR trajectories and $\mathrm{Filter}(\cdot)$ keeps only trajectories that satisfy $R_{\text{plan}}{=}R_{\text{format}}{=}R_{\text{tool}}{=}1$.

As shown in Fig. 2(b), each generic stage (Eqs. 2–4) maps directly to a concrete RPCO stage, detailed below. A step-by-step pseudocode is provided in Algorithm 2 (Appendix G).

Stage 1 – Reflection on Single-Image Tasks. This stage runs the *Decouple* step of Eq. 2 by picking single-image queries Q_s as the simplified environment E_m^{simple}, where one generation step yields low diffusion noise:

$$\pi_{\text{SR}} \leftarrow \mathrm{GRPO}\Big(\mathrm{SFT}\big(\pi_0;\, \mathrm{Traj}(E_{\text{plan}}, Q_s)\big);\, Q_s,\, W_1\Big). \tag{5}$$

Concretely, we first bootstrap π_0 via $\mathrm{SFT}(\pi_0;\, \mathrm{Traj}(E_{\text{plan}}, Q_s))$ with UTPCR trajectories rolled out from Gemini2.5Pro (E_{plan}), which sets up basic reflection behaviors. We then apply GRPO on the SFT-initialized model with the active reward set W_1 (planning omitted since it is trivial under single-image tasks), so that the visual reflection reward R_{reflect} dominates and the rule-based terms regularize structure. The resulting π_{SR} is the *Strong-Reflection* model (Sec. 4.1), which shows reflection quality above Gemini2.5Pro and supplies high-quality reflection trajectories for Stage 2.

Stage 2 – Advantage-Complementary SFT. This stage runs the *Fuse* step of Eq. 3 by transferring heterogeneous capabilities via supervised learning, and thus avoids the high-variance reflection RL gradients identified in Proposition 4.2:

$$\pi_{\text{RP}} \leftarrow \mathrm{SFT}\big(\pi_{\text{SR}};\, \mathrm{Filter}(\mathrm{Traj}(\pi_{\text{SR}}, Q_s)) \cup \mathrm{Filter}(\mathrm{Traj}(E_{\text{plan}}, Q_m))\big). \tag{6}$$

The SFT corpus is advantage-complementary: (i) *Reflection-strong single-image trajectories* from π_{SR} on Q_s give high-quality reflection patterns with clear error diagnosis and correction; (ii) *Planning-strong multi-image trajectories* from E_{plan} on Q_m show coherent global plans and high plan scores. The $\mathrm{Filter}(\cdot)$ operator keeps only trajectories with $R_{\text{plan}}{=}R_{\text{format}}{=}R_{\text{tool}}{=}1$, which keeps the supervision clean. The resulting π_{RP} is the *Reflection-Plan SFT* model, which, as shown in Tab. 6 (row (e)), largely improves multi-image planning (plan score 0.945) while bringing balanced reflection behavior back.

Stage 3 – Multi-task RL & Co-Optimization. This stage runs the *Co-Optimize* step of Eq. 4 by fine-tuning π_{RP} with GRPO jointly on single-image and multi-image queries, using the full active reward set W_3:

$$\pi^\star \leftarrow \text{GRPO}\big(\pi_{\text{RP}};\, Q_s \cup Q_m,\, W_3\big). \tag{7}$$

Starting from the balanced prior built in Stage 2 is key: we only fine-tune rather than learn from scratch, so exposure to the high-variance reflection regime stays low, while R_{plan} and R_{reflect} jointly drive co-optimization and the rule-based terms act as structural regularizers. Once training converges, we obtain the final policy $\pi^\star$, namely **VisionCreator-R1**.

6 VCR Dataset

We construct VCR-SFT and VCR-RL datasets to support the RPCO methodology, and further propose VCR-Bench for standardized assessment. Fig. 4 shows the task distribution of the VCR dataset, which includes the tasks of Single-Img, Multi-Img, and Img2Img.

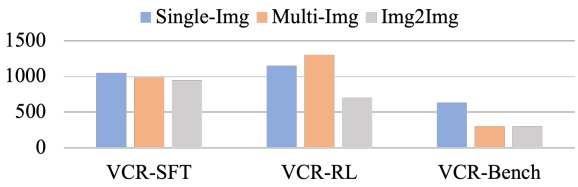


Figure 4: Task distribution of the VCR dataset.

6.1 VCR-Data: SFT and RL Resources

VCR-SFT. VCR-SFT is designed for supervised fine-tuning, and gives explicit UTPCR-format trajectories with explicit tags for understanding, thinking, planning, creation, and reflection. VCR-SFT dataset distribution is shown in Fig. 5. To keep SFT data quality high, we strictly filter the trajectories by keeping only those that get perfect scores with $\{R_{\text{plan}} = 1, R_{\text{format}} = 1, R_{\text{tool}} = 1\}$. More importantly, VCR-SFT gives effective supervision for improving both reflection capability and planning capability. Specifically, VCR-SFT has two complementary subsets: (i) *Single-image reflection-strong trajectories*, collected from our Strong-Reflection model. They give high-quality supervision for grounded visual reflection behaviors. (ii) *Multi-image planning-strong trajectories*, collected from Gemini2.5Pro on multi-image tasks. These trajectories emphasize strong global planning.

VCR-RL. VCR-RL is designed for RL. Each VCR-RL sample has a user query and a list of reflection evaluation checkpoints that specify desired properties of the *final visual outputs*. During RL training, the *reflection reward* is computed by running a VLM-based judge on the post-reflection images against the query-specific reflection checkpoints, i.e., each query's checkpoints are tailored to its specific requirements. This diversity trains the agent to adapt its reflection and correction strategies to varied task demands, which helps robustness and generalization.

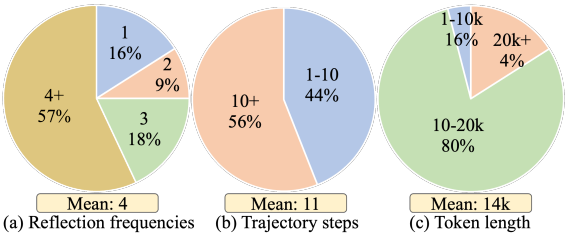


Figure 5: VCR-SFT dataset distribution.

6.2 VCR-Bench: Evaluation Suite

VCR-Bench reports final task performance, and also supports trajectory-level analysis by computing reflection and plan statistics from inference traces. Each benchmark query has a user instruction and a set of evaluation checkpoints, and we use *Gemini2.5Pro* as a VLM-based judge for automatic evaluation. To check robustness against judge-specific bias, we also re-evaluate all methods with *Qwen3VL235B* as a second independent judge in Sec. 7.1.

7 Experiment

Baselines and Implementations. The used image generation and editing tool Qwen-Image-Fast is Qwen-Image-Edit-2509-Lightning-4steps-V1.0 [24]. All compared agent models (i.e., Qwen3VL32B, Gemini2.5Pro) use the same UTPCR-format system prompt. Our VisionCreator-R1 adopts Qwen3VL32B as the base model, and sets $N = 6$ for plan reward.

Table 3: Performance on VCR-Bench under two independent VLM judges.

	Gemini2.5Pro Judge			Qwen3VL235B Judge			Average Result		
Model	Single	Multi	Img2Img	Single	Multi	Img2Img	Single	Multi	Img2Img
Qwen-Image-Fast	0.505	0.480	0.672	0.795	0.561	0.632	0.650	0.521	0.652
Qwen3VL32B	0.500	0.601	0.806	0.835	0.717	0.740	0.668	0.659	0.773
Gemini2.5Pro	0.515	0.649	0.816	0.844	0.783	0.737	0.680	0.716	0.777
VisionCreator-R1	**0.532**	**0.700**	**0.836**	**0.858**	**0.832**	**0.750**	**0.695**	**0.766**	**0.793**
Pairwise human evaluation (VC-R1 vs. Gemini2.5Pro): Single +14.8%, Multi +9.3%, Img2Img +5.8%									

7.1 Results on VCR-Bench

Tab. 3 summarizes the performance comparison on **VCR-Bench** across three representative task categories: single-image generation, multi-image generation, and image-to-image editing. To reduce the circularity concern of using Gemini2.5Pro as both judge and a baseline, we also evaluate with a second independent VLM judge (Qwen3VL235B) and report the averaged score.

Automatic evaluation by VLM judges. Under both judges and their average, *VisionCreator-R1* gets the best performance across all three task categories. In particular, the gain on *multi-image tasks* is the most pronounced, where VisionCreator-R1 beats Gemini2.5Pro itself by a large margin (0.700 vs. 0.649 under Gemini judge; 0.832 vs. 0.783 under Qwen3VL235B judge). This shows that explicit reflection and Reflection-Plan co-optimization enable more reliable long-horizon reasoning and error correction than purely plan-driven generation.

Human evaluation. To check that automatic scores align with human preference, we run a pairwise human evaluation between *VisionCreator-R1* and *Gemini2.5Pro*. Human annotators are asked to judge which output better satisfies the user intent across all evaluation checkpoints. As shown in Tab. 3, VisionCreator-R1 is preferred over Gemini2.5Pro in **14.8%** of single-image task, **9.3%** of multi-image task, and **5.8%** of image-to-image task. These results confirm that the improvements measured by VLM-based evaluation match real quality gains perceived by human users. The consistency between automatic and human evaluations supports the reliability of VCR-Bench as a testbed for evaluating visual generation.

Table 4: Comparison of semantic consistency (G_SC), perceptual quality (G_PQ), overall score (G_O) on GEdit-Bench.

Model	G_SC	G_PQ	G_O
Gemini 2.0 [25]	6.73	6.61	6.32
BAGEL [26]	7.36	6.83	6.52
FLUX.1 Kontext Pro [27]	7.02	**7.60**	6.56
Step1X-Edit [13]	**7.66**	7.35	6.97
Qwen-Image-Fast	7.26	7.34	7.03
VisionCreator-R1	7.60	7.33	**7.23**

7.2 Results on GEdit-Bench

Tab. 4 reports the results on the **GEdit-Bench** [13], which evaluates image editing models on user instructions across 11 diverse categories. The results show that our agent, VisionCreator-R1, gets a competitive result. In particular, VisionCreator-R1 largely improves the semantic consistency score over the base tool (7.60 vs. 7.26), which shows that our agent's reflection and planning abilities enable more accurate and intent-aligned editing.

7.3 Results on MultiBanana Benchmark

We further evaluate on **MultiBanana** [14], a recent public benchmark for multi-reference image generation. As shown in Tab. 5, VisionCreator-R1 beats IPR-Agent [14] across all six categories, and

Table 5: Results on MultiBanana (10-point scale).

Model	Single Obj	Two Obj	X Obj	X-1+Local	X-1+Global	X-1+BG
Nano Banana [25]	7.817	4.891	4.453	4.118	4.698	3.575
GPT-Image-1 [28]	7.804	6.585	5.086	5.147	5.757	5.019
Qwen-Image-Fast	7.276	3.489	2.067	2.001	2.260	2.011
+ IPR-Agent-32B [14]	7.644	4.494	3.640	3.514	3.658	3.300
+ VisionCreator-R1	**7.891**	**4.834**	**4.133**	**3.914**	**4.062**	**3.603**

Table 6: Systematic ablation isolating each RPCO stage. (a) skips Stage 1&2, applying RL directly from base. (b) skips Stage 2 (= Reflection-Plan Conflict). (c)/(d) ablate Stage 2 SFT data composition. (e) skips Stage 3 (= Reflection-Plan SFT). (f) ablates the plan reward in Stage 3. (g) is the full RPCO. All reflection (U/G/O) and Plan statistics are computed on Multi-Img traces.

#	Configuration	Stage 1	Stage 2 Data	Stage 3	Multi-Img	Ref. (U/G/O)	Plan
(a)	Base → RL	×	–	w/ plan	0.576	74.2/15.8/8.0	0.819
(b)	Strong-Refl. → RL	✓	–	w/ plan	0.617	73.9/16.5/9.7	0.843
(c)	Refl.-Plan SFT (Refl only) → RL	✓	Refl traj only	w/ plan	0.640	66.2/27.5/6.3	0.860
(d)	Refl.-Plan SFT (Plan only) → RL	✓	Plan traj only	w/ plan	0.668	67.1/25.8/7.1	0.952
(e)	Refl.-Plan SFT (no Stage 3)	✓	Mixed	–	0.647	64.0/**32.4**/3.6	0.945
(f)	VisionCreator-R1 (w/o plan reward)	✓	Mixed	w/o plan	0.672	64.8/30.1/5.1	0.954
(g)	**VisionCreator-R1 (full RPCO)**	✓	**Mixed**	**w/ plan**	**0.700**	63.5/31.0/5.5	**0.975**

also beats Nano Banana [25] on Single-Object and X-1+Background, which confirms generalization to an independent benchmark.

7.4 Systematic Ablation on VCR-Bench

We report a systematic stage-wise ablation of RPCO in Tab. 6, which covers all seven configurations obtained by enabling/disabling each stage or by varying Stage-2 SFT data composition.

Stage-wise contribution. (i) *Stage 1 matters* ((a) vs. (b)): RL-trained reflection improves Multi-Img 0.576→0.617. (ii) *Advantage-complementary SFT is key* ((c) vs. (d) vs. (e)): mixed data gets the best balance of Good-Reflection (32.4%) and Plan score (0.945). (iii) *Stage 3 RL is essential* ((e) vs. (g)): +0.053 Multi-Img. (iv) *Plan reward in Stage 3 contributes* ((f) vs. (g)): +0.028 Multi-Img.

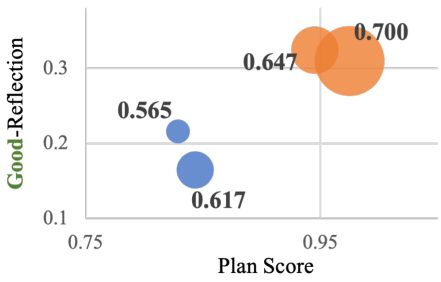


Figure 6: Plan-Reflection with Performance.

Discussion. From *Strong-Reflection* (b) to full *VisionCreator-R1* (g), we see that when planning capability is weak, directly optimizing reflection via RL leads to reflection degradation. According to Proposition 4.2, weak planning produces more invalid intermediate images, which triggers frequent reflection and extra tool invocations. This lengthens trajectories and amplifies diffusion-induced stochasticity, which causes the trajectory variance Σ_τ to dominate the gradient estimator. Under this high-variance regime, GRPO cannot reliably attribute rewards to reflection actions, which results in unstable optimization. This explains why strong planning before reflection-optimized RL is necessary. In line with this analysis, initializing from *Reflection-Plan SFT* largely stabilizes training, and leads to the strongest performance in *VisionCreator-R1*. Fig. 6 shows that improving reflection and planning capabilities brings clear performance gains on multi-image tasks.



Figure 7: Visualization comparisons.

8 Conclusion

We study reflection as a trainable capability in native visual generation agents, and show that naïvely optimizing reflection via RL is fundamentally limited by a structural variance asymmetry. By isolating reflection in low-noise single-image settings, we show that reflection is learnable and largely improves visual quality. We further show that in multi-image tasks, planning and reflection have asymmetric optimization regimes, which motivates a decoupled-then-fused strategy. Guided by this idea, we propose RPCO to train VisionCreator-R1, which beats Gemini2.5Pro across single-image, multi-image, and image-to-image benchmarks. Our results set up principled guidelines for training reflection-aware visual agents under long-horizon stochastic environments. We further release VCR-SFT, VCR-RL, and VCR-Bench to support future research on reflection-aware visual generation.

Acknowledgments and Disclosure of Funding

This research was supported by fundings from the Hong Kong RGC General Research Fund (152228/23E, 162161/24E, 162116/25E, 162180/25E), National Natural Science Foundation of China (NSFC) Key Program (No.62532005), Collaborative Research Fund (No. C1042-23GF, No. 5097-25G), NSFC/RGC Collaborative Research Scheme (Grant No. 62461160332 & CRS_HKUST602/24), Research Impact Fund (No. R5011-23), and Areas of Excellence Scheme (AoE/E-601/22-R).

References

[1] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[2] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," *arXiv preprint arXiv:2204.06125*, vol. 1, no. 2, p. 3, 2022.

[3] B. F. Labs, "Flux," https://github.com/black-forest-labs/flux, 2024.

[4] B. Lin, Y. Ge, X. Cheng, Z. Li, B. Zhu, S. Wang, X. He, Y. Ye, S. Yuan, L. Chen *et al.*, "Open-sora plan: Open-source large video generation model," *arXiv preprint arXiv:2412.00131*, 2024.

[5] W. Wu, Z. Zhu, and M. Z. Shou, "Automated movie generation via multi-agent cot planning," *arXiv preprint arXiv:2503.07314*, 2025.

[6] J. Xiao, C. Yang, L. Zhang, S. Cai, Y. Zhao, Y. Guo, G. Wetzstein, M. Agrawala, A. Yuille, and L. Jiang, "Captain cinema: Towards short movie generation," *arXiv preprint arXiv:2507.18634*, 2025.

[7] X. Xu, J. Mei, C. Li, Y. Wu, M. Yan, S. Lai, J. Zhang, and M. Wu, "Mm-storyagent: Immersive narrated storybook video generation with a multi-agent paradigm across text, image and audio," *arXiv preprint arXiv:2503.05242*, 2025.

[8] X. Xue, Z. Lu, D. Huang, Z. Wang, W. Ouyang, and L. Bai, "Comfybench: Benchmarking llm-based agents in comfyui for autonomously designing collaborative ai systems," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 24 614–24 624.

[9] L. Guo, X. Xu, L. Wang, J. Lin, J. Zhou, Z. Zhang, B. Su, and Y.-C. Chen, "Comfymind: Toward general-purpose generation via tree-based planning and reactive feedback," *arXiv preprint arXiv:2505.17908*, 2025.

[10] J. Lai, Z. Lu, J. He, R. Quan, W. Zhao, Q. Yang, Q. Chen, Q. Lin, C. Li, T. Gao *et al.*, "Visioncreator: A native visual-generation agentic model with understanding, thinking, planning and creation," *CVPR Findings*, 2026.

[11] Y. Lin, L. Wang, K. Lin, Z. Lin, K. Gong, W. Li, B. Lin, Z. Li, S. Zhang, Y. Peng *et al.*, "Jarvisevo: Towards a self-evolving photo editing agent with synergistic editor-evaluator optimization," *arXiv preprint arXiv:2511.23002*, 2025.

[12] F. Yin, S. Liu, Y. Han, Z. Wang, P. Xing, R. Wang, W. Cheng, Y. Wang, A. Li, Z. Yin *et al.*, "Reasonedit: Towards reasoning-enhanced image editing models," *arXiv preprint arXiv:2511.22625*, 2025.

[13] S. Liu, Y. Han, P. Xing, F. Yin, R. Wang, W. Cheng, J. Liao, Y. Wang, H. Fu, C. Han *et al.*, "Step1x-edit: A practical framework for general image editing," *arXiv preprint arXiv:2504.17761*, 2025.

[14] Y. Oshima, D. Miyake, K. Matsutani, Y. Iwasawa, M. Suzuki, Y. Matsuo, and H. Furuta, "Multibanana: A challenging benchmark for multi-reference text-to-image generation," in *CVPR*, 2026.

[15] M. Chen, A. Radford, R. Child, J. Wu, H. Jun, D. Luan, and I. Sutskever, "Generative pretraining from pixels," in *International conference on machine learning*. PMLR, 2020, pp. 1691–1703.

[16] L. Fan, T. Li, S. Qin, Y. Li, C. Sun, M. Rubinstein, D. Sun, K. He, and Y. Tian, "Fluid: Scaling autoregressive text-to-image generative models with continuous tokens," *arXiv preprint arXiv:2410.13863*, 2024.

[17] J. Han, J. Liu, Y. Jiang, B. Yan, Y. Zhang, Z. Yuan, B. Peng, and X. Liu, "Infinity: Scaling bitwise autoregressive modeling for high-resolution image synthesis," *CVPR*, 2025.

[18] K. Tian, Y. Jiang, Z. Yuan, B. Peng, and L. Wang, "Visual autoregressive modeling: Scalable image generation via next-scale prediction," *Advances in neural information processing systems*, 2024.

[19] P. Sun, Y. Jiang, S. Chen, S. Zhang, B. Peng, P. Luo, and Z. Yuan, "Autoregressive model beats diffusion: Llama for scalable image generation," *arXiv preprint arXiv:2406.06525*, 2024.

[20] Y. Pang, P. Jin, S. Yang, B. Lin, B. Zhu, Z. Tang, L. Chen, F. E. Tay, S.-N. Lim, H. Yang *et al.*, "Next patch prediction for autoregressive visual generation," *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026.

[21] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. Gontijo Lopes, B. Karagol Ayan, T. Salimans *et al.*, "Photorealistic text-to-image diffusion models with deep language understanding," *Advances in neural information processing systems*, vol. 35, pp. 36 479–36 494, 2022.

[22] Z. Xu, Y. Wang, X. Yang, L. Wang, W. Luo, K. Zhang, B. Hu, and M. Zhang, "Comfyui-r1: Exploring reasoning models for workflow generation," *arXiv preprint arXiv:2506.09790*, 2025.

[23] O. Huang, Y. Ma, Z. Zhao, M. Wu, J. Ji, R. Zhang, Z. Hu, X. Sun, and R. Ji, "Comfygpt: A self-optimizing multi-agent system for comprehensive comfyui workflow generation," *arXiv preprint arXiv:2503.17671*, 2025.

[24] Qwen, "Qwen-image-lightning," https://huggingface.co/lightx2v/Qwen-Image-Lightning, 2025.

[25] G. DeepMind, "Gemini 2.0," https://gemini.google.com/, 2025.

[26] C. Deng, D. Zhu, K. Li, C. Gou, F. Li, Z. Wang, S. Zhong, W. Yu, X. Nie, Z. Song *et al.*, "Emerging properties in unified multimodal pretraining," *arXiv preprint arXiv:2505.14683*, 2025.

[27] B. F. Labs, S. Batifol, A. Blattmann, F. Boesel, S. Consul, C. Diagne, T. Dockhorn, J. English, Z. English, P. Esser *et al.*, "Flux. 1 kontext: Flow matching for in-context image generation and editing in latent space," *arXiv preprint arXiv:2506.15742*, 2025.

[28] OpenAI, "Gpt-image-1," https://openai.com/index/introducing-4o-image-generation/, 2025.

A Extended Related Works

A.1 Image Generation

Modern image generation models, such as autoregressive models [15–20] and diffusion models [1–3, 21, 4], offer single-step text-to-image and image-to-image capabilities. However, they do not themselves provide mechanisms for task decomposition, tool orchestration, or structured reflection over intermediate results. VisionCreator-R1 treats these generators as callable tools, focusing on how to *plan, reflect, and correct* across long visual trajectories.

A.2 Visual Generation Agents

Workflow-Specific Agents. [5–7] design domain-specific pipelines for tasks like movie-style storyboarding or episodic content creation. They struggle when task structure or tools deviate from the hand-crafted templates.

Workflow-Guided Agents. [8, 9, 22, 23] focus on generating ComfyUI-compatible node graphs. They advance automated workflow construction, but are tied to a specific execution backend, and typically realise control flow via static prompts and rules.

Native Visual Agents. Such as VisionCreator [10] unify UTPC in a trainable framework, outperforming "VLM+tools" baselines. However, existing UTPC agents are still *plan-driven*: training and evaluation primarily supervise whether plans are reasonable, and tools are called valid. They lack an explicit, structured *reflection* path to evaluate intermediate images and trigger self-corrections. Our VisionCreator-R1 builds on this line of work but introduces reflection and studies how to co-optimise reflection and planning.

B VisionCreator-R1 Reward Design

In this section, we describe the multi-dimensional reward system used to train VisionCreator-R1 with reinforcement learning. The reward mechanism decomposes the supervision signal into complementary components that jointly guide the agent through high-level planning, structural compliance, tool execution, result satisfaction, and visual reflection.

B.1 Overall Reward Function

We aggregate multiple sub-rewards via a weighted average to ensure the agent receives learning signals at all stages of the UTPCR pipeline. The total reward is calculated as

$$R_{\text{total}} = \frac{1}{|\mathcal{W}|} \sum_{i \in \mathcal{W}} w_i \cdot R_i, \tag{8}$$

where $\mathcal{W}$ represents the set of reward dimensions comprising plan, format, tool, result, and reflection components. The weight w_i for each dimension is set to 1 by default, while each sub-reward R_i is normalized to the range $[0, 1]$. Specifically, the Plan Reward R_{plan}, Format Reward R_{format}, and Reflection Reward R_{reflect} take values within $[0, 1]$. The Tool Call Reward R_{tool} assumes discrete values from the set $\{0, 0.1, 0.8, 1\}$, and the Result Reward R_{result} is a binary value in $\{0, 1\}$. This design balances the training process, preventing the agent from being driven solely by the final output quality.

B.2 Plan Reward

The Plan Reward evaluates the quality of the agent's task plan, ensuring it correctly understands user requirements and proposes a coherent, tool-aware execution strategy. Let the user requirement be Q and the agent's plan be P. We utilize an external LLM-based evaluator $f_{\text{eval}}(Q, P)$ that returns a discrete score in the range $\{1, 2, \ldots, N\}$ with $N = 6$. This score is normalized to obtain the final reward

$$R_{\text{plan}} = \frac{f_{\text{eval}}(Q, P)}{N}. \tag{9}$$

The evaluator f_{eval} jointly considers three critical aspects: requirement completeness, which checks if the plan covers the correct output types and counts; logical coherence, which assesses the reasonableness of sub-task decomposition and ordering; and tool matching, which verifies that the chosen tools are appropriate for the intended sub-goals.

B.3 Format Reward

The Format Reward assesses whether the agent's outputs strictly follow the predefined UTPCR tag schema, a crucial factor for stable training and reliable tool orchestration. We enforce three canonical patterns depending on the dialogue turn. The first turn must follow the sequence of thinking, planning, thinking, tool calling, and tool result processing. Middle turns consist of reflection, thinking, tool calling, and tool result processing. The final turn requires a sequence of reflection, thinking, and the final answer.

For a trajectory with T turns, let $s_t \in [0, 1]$ denote the format score of turn t. The trajectory-level reward is defined as the minimum score across all turns, $R_{\text{format}} = \min_t s_t$, ensuring that a single badly formatted turn penalizes the entire interaction. For each individual turn, the score is calculated as

$$s_t = \frac{N_{\text{valid}} + \mathbb{1}_{\text{order}}}{N_{\text{required}} + 1}, \tag{10}$$

where N_{valid} is the number of tags passing validation, N_{required} is the expected number of tags for that turn type, and $\mathbb{1}_{\text{order}}$ is an indicator function that equals 1 if the tag order is fully correct. A tag is considered valid if and only if it exists, appears exactly once, and contains non-empty content.

B.4 Tool Call Reward

The Tool Call Reward measures the correctness and robustness of tool usage based on execution success. It encourages consistent success while providing partial credit for self-correcting behavior. We define a success indicator $success(r)$ which equals 1 if the tool execution result r indicates success, and 0 otherwise. For a trajectory with n tool calls and results $(r_1, \ldots, r_n)$, we define three conditions: $\alpha = \prod_{i=1}^{n-1} success(r_i)$ represents success in all intermediate calls; $\beta = success(r_n)$ represents success in the final call; and $\gamma = \max_i success(r_i)$ indicates at least one successful call occurred.

The Tool Call Reward is assigned according to a piecewise schedule:

$$R_{\text{tool}} = \begin{cases} 1.0, & \text{if } \alpha = 1 \wedge \beta = 1, \\ 0.8, & \text{if } \alpha = 0 \wedge \beta = 1, \\ 0.1, & \text{if } \beta = 0 \wedge \gamma = 1, \\ 0.0, & \text{otherwise.} \end{cases} \tag{11}$$

This scheme assigns the maximum reward of 1.0 for consistently correct usage. It grants 0.8 for self-corrected paths where early failures are followed by a successful final call, 0.1 for partial progress where failures persist but some attempts succeeded, and 0.0 for complete failure.

B.5 Result Reward

The Result Reward validates whether the final task output satisfies basic quantitative constraints, specifically the requested numbers of images and videos. Let $N_{\text{img}}^{\text{exp}}$ and $N_{\text{img}}^{\text{act}}$ be the expected and actual image counts, and $N_{\text{vid}}^{\text{exp}}$ and $N_{\text{vid}}^{\text{act}}$ be the expected and actual video counts. The reward is defined as

$$R_{\text{result}} = \mathbb{1}_{(N_{\text{img}}^{\text{act}} = N_{\text{img}}^{\text{exp}})} \cdot \mathbb{1}_{(N_{\text{vid}}^{\text{act}} = N_{\text{vid}}^{\text{exp}})}. \tag{12}$$

This formula yields a value of 1 only if both image and video counts strictly match the requirements, and 0 otherwise. This binary signal enforces minimum task completion criteria independent of finer-grained visual quality.

B.6 Reflection Reward

The Reflection Reward leverages a VLM to evaluate whether the *post-reflection* images satisfy a set of pre-defined evaluation points derived from the user requirement. We decompose the user

requirement into K checkpoints $\{q_1, \ldots, q_K\}$. Given the final image set $\mathcal{I}$, a VLM-based judge $g(q_k, \mathcal{I})$ determines whether each checkpoint q_k is accepted or refused. Letting c_k denote the judgment result, the reward is calculated as the fraction of satisfied checkpoints:

$$R_{\text{reflect}} = \frac{1}{K} \sum_{k=1}^{K} \mathbb{1}_{(c_k=\text{accept})}. \tag{13}$$

The evaluation points typically cover subject correctness, style consistency, attribute accuracy, scene matching, action depiction, and text accuracy.

B.7 Reward Properties and Interactions

The proposed reward system mixes discrete and continuous signals to stabilize RL training while giving rich supervision over structure and content. R_{format} acts as a continuous signal, while R_{plan}, R_{tool}, R_{result}, and R_{reflect} give discrete feedback steps. There are built-in dependencies among these dimensions: high-quality plans raise the chance of meeting basic output constraints, and correct tool usage is a prerequisite for desired results. Also, good reflection combined with strong initial results leads to high post-reflection visual quality. Taken together, this multi-dimensional design covers the full lifecycle of a VisionCreator-R1 trajectory, and lets the Reflection-Plan Co-Optimization framework balance improvements in planning and reflection capabilities.

C Rigorous Derivation of Gradient Variance in GRPO

This appendix provides the full derivation of Theorem 4.1 (the gradient-variance decomposition) and justifies Proposition 4.2 (the planning-vs-reflection asymmetry).

C.1 The GRPO Gradient Estimator

The gradient of the GRPO objective is estimated by sampling a group of outputs $\{o_i\}_{i=1}^{G}$. To analyze the optimization difficulty, we focus on the stochastic gradient estimator $\hat{g}$ for a single token generation step t in a single trajectory i.

Let s be the context $(q, o_{i,<t})$ and a be the current token $o_{i,t}$. The estimator $\hat{g}$ is (assume $\pi_\theta = \pi_{old}$ for simplified analysis):

$$\hat{g} = \left[\hat{A}_{i,t} + \beta \left(\frac{\pi_{ref}(a|s)}{\pi_\theta(a|s)} - 1 \right) \right] \nabla_\theta \log \pi_\theta(a|s), \tag{14}$$

where $\hat{A}_{i,t}$ denotes the advantage derived from the group rewards ($\hat{A}_{i,t} = \frac{r_i - \mu_r}{\sigma_r}$), in which r_i crucially depends on the entire final trajectory τ. The term involving β represents the KL regularization; for a fixed policy state θ and reference policy π_{ref}, given action a, this term is deterministic. We denote this deterministic scalar part as $D(a, s) = \beta \left(\frac{\pi_{ref}}{\pi_\theta} - 1 \right)$. Consequently, the stochasticity inherent in $\hat{g}$ stems from two sources: the policy-driven action sampling ($a \sim \pi_\theta$) for token selection, and the trajectory stochasticity ($\tau|a$). The latter encapsulates the variability of the image diffusion process, which governs the advantage estimate $\hat{A}_{i,t}$.

C.2 Variance Decomposition via Law of Total Variance

We decompose the total variance $\Sigma := \text{Var}(\hat{g})$ by recursively applying the Law of Total Variance.

First, we decompose the variance with respect to the state s:

$$\Sigma = \text{Var}_{s,a,\tau}(\hat{g}) = \mathbb{E}_s \left[\text{Var}_{a,\tau|s}(\hat{g}) \right] + \text{Var}_s \left(\mathbb{E}_{a,\tau|s}[\hat{g}] \right). \tag{15}$$

Next, we further decompose the inner term $\text{Var}_{a,\tau|s}(\hat{g})$ by conditioning on the action a, given the state s:

$$\begin{aligned}\Sigma &= \mathbb{E}_s\left[\mathbb{E}_{a|s}\left[\text{Var}_{\tau|s,a}(\hat{g})\right] + \text{Var}_{a|s}\left(\mathbb{E}_{\tau|s,a}[\hat{g}]\right)\right] + \underbrace{\text{Var}_s\left(\mathbb{E}_{a,\tau|s}[\hat{g}]\right)}_{\Sigma_s} \\ &= \underbrace{\mathbb{E}_{s,a}\left[\text{Var}_{\tau|s,a}(\hat{g})\right]}_{\Sigma_\tau} + \underbrace{\mathbb{E}_s\left[\text{Var}_{a|s}\left(\mathbb{E}_{\tau|s,a}[\hat{g}]\right)\right]}_{\Sigma_a} + \Sigma_s,\end{aligned} \tag{16}$$

where Σ_τ represents the variance attributable to environmental dynamics and reward stochasticity, conditioned on a specific state and action. Σ_a denotes the variance resulting from the policy's stochastic action sampling given a state; notably, this is the specific component minimized by the baseline in GRPO. Finally, Σ_s captures the variability stemming from the underlying distribution of states (i.e., input prompts). Since Σ_s measures the variance of the true gradients across different prompts and is independent of the algorithm design, we omit it from our consideration. Thus, $\Sigma = \Sigma_\tau + \Sigma_a$.

C.3 Step-by-Step Derivation of Terms

First, we analyze the variance of the gradient estimator $\hat{g}$ conditioned on a specific action a, denoted as Σ_τ. Given a fixed action a, both the function $\mathbf{v} = \nabla_\theta \log \pi_\theta(a|s)$ and the KL-divergence term $D(a, s)$ are treated as constants. Consequently, the conditional variance arises solely from the Advantage term $\hat{A}_{i,t}$, which fluctuates due to the stochastic nature of the trajectory rewards r_i (e.g., inherent noise in the diffusion process).

Substituting the expression for $\hat{g}$ into the conditional variance formulation yields:

$$\begin{aligned}\text{Var}_{\tau|a}(\hat{g}) &= \text{Var}_{\tau|a}\left((\hat{A}_{i,t} + D(a,s)) \cdot \mathbf{v}\right) \\ &= \mathbf{v}\mathbf{v}^\top \cdot \text{Var}_{\tau|a}(\hat{A}_{i,t}).\end{aligned} \tag{17}$$

The total trajectory variance Σ_τ is obtained by taking the expectation over the action space:

$$\Sigma_\tau = \mathbb{E}_a\left[\|\nabla_\theta \log \pi_\theta(a|s)\|^2 \cdot \text{Var}_{\tau|a}(\hat{A}_{i,t})\right]. \tag{18}$$

This derivation reveals a critical challenge in image-generating tasks, particularly when employing reflection mechanism. Since the advantage $\hat{A}_{i,t}$ depends on the stochastic outcomes of the diffusion model, the term $\text{Var}_{\tau|a}(\hat{A}_{i,t})$ can become exceedingly large. Even when the agent selects a consistent reflection token a, the varying quality of the generated images introduces significant noise into the reward signal, thereby amplifying the overall variance of the estimator. Moreover, the synthesis of multiple images introduces significantly higher environmental dynamics and compounded stochasticity. This amplified variance renders the optimization landscape far more volatile, making the training process significantly more difficult than in standard single-image tasks.

Next, we analyze the expectation term $\mathbb{E}_{\tau|a}[\hat{g}]$.

$$\begin{aligned}\mathbb{E}_{\tau|a}[\hat{g}] &= \mathbb{E}_{\tau|a}\left[(\hat{A}_{i,t} + D(a,s)) \cdot \nabla_\theta \log \pi_\theta(a|s)\right] \\ &= \left(\mathbb{E}_{\tau|a}[\hat{A}_{i,t}] + D(a,s)\right) \cdot \nabla_\theta \log \pi_\theta(a|s).\end{aligned} \tag{19}$$

Let $\mathcal{A}(s, a) = \mathbb{E}_{\tau|a}[\hat{A}_{i,t}]$ be the expected advantage of taking action a. This gives the quality of the action, averaged over environmental noise.

Thus, Σ_a is the variance of this expected gradient across different actions:

$$\Sigma_a = \text{Var}_a\left(\left[\mathcal{A}(s,a) + \beta\left(\frac{\pi_{ref}(a|s)}{\pi_\theta(a|s)} - 1\right)\right] \cdot \nabla_\theta \log \pi_\theta(a|s)\right). \tag{20}$$

This term captures the informative learning signal, which quantifies the gradient variability induced by the agent's exploration across the action space a.

Finally, combining the terms:

$$\text{Var}(\hat{g}) = \underbrace{\mathbb{E}_a[\|\nabla \log \pi\|^2 \cdot \text{Var}_{\tau|a}(\hat{A}_{i,t})]}_{\Sigma_\tau} + \underbrace{\text{Var}_a\left(\left[\mathcal{A}(s,a) + \beta\left(\frac{\pi_{ref}(a|s)}{\pi_\theta(a|s)} - 1\right)\right] \cdot \nabla_\theta \log \pi_\theta(a|s)\right)}_{\Sigma_a} \tag{21}$$

We point to a critical gap in the variance composition between the planning and reflection objectives, which we call *structural asymmetry*.

For the planning mechanism (R_{plan}), the reward evaluation is typically deterministic. Given a fixed action a, the reward signal stays invariant, which makes the conditional variance negligible:

$$\text{Var}_{\tau|a}(\hat{A}_{i,t}) \approx 0 \quad \Longrightarrow \quad \Sigma_\tau \approx 0. \tag{22}$$

Under these conditions, the total variance is dominated by the variance term Σ_a. So the optimization process stays stable, as the gradient mostly reflects the useful learning signal from action exploration.

In contrast, the reflection mechanism ($R_{reflect}$) directly depends on the stochastic outcomes of the diffusion model. This compounded stochasticity makes the conditional variance become excessively large:

$$\text{Var}_{\tau|a}(\hat{A}_{i,t}) \gg 0 \quad \Longrightarrow \quad \Sigma_\tau \gg \Sigma_a. \tag{23}$$

So the gradient estimator is dominated by the trajectory noise Σ_τ rather than the informative signal Σ_a. This imbalance causes a collapse in the Signal-to-Noise Ratio, and makes the optimization landscape highly volatile. Under such conditions, the stable acquisition of gradients is blocked, and the reflection capabilities become hard to optimize.

D Empirical Variance Measurement: Full Protocol

This appendix provides the complete protocol and per-step table for the empirical variance measurement summarized in Sec. 4.3.

Protocol. We use the Reflection-Plan Conflict model's GRPO training rollouts ($G = 4$ trajectories per prompt) on multi-image tasks. At each selected training step, we sample prompts from the multi-image subset of VCR-RL, collect G rollouts per prompt, and compute the variance of R_{plan} and R_{reflect} across the G trajectories within each prompt, then average over prompts. Since R_{plan} is evaluated by a deterministic LLM judge (no diffusion involved), its variance approximates Σ_a only. R_{reflect} depends on stochastic diffusion outcomes, so its excess variance reflects Σ_τ.

Table 7: Empirical variance of plan vs. reflection rewards across GRPO training steps on multi-image tasks. The ratio Var(R_{reflect})/Var(R_{plan}) grows from 4.8× (early) to 15.5× (later), confirming the structural variance gap predicted by Proposition 4.2.

Training Step	Var(R_{plan})	Var(R_{reflect})	Ratio
Step 10	0.035	0.168	4.8×
Step 50	0.008	0.124	15.5×

Interpretation. As the policy converges (lower entropy), Var(R_{plan}) drops while Var(R_{reflect}) keeps a high floor due to irreducible diffusion stochasticity. The ratio therefore *worsens* as training deepens, which confirms that the asymmetry is a structural property rather than an artifact of early-stage initialization.

E Intuitive Explanation of the Variance Asymmetry

This appendix provides the detailed intuitive reasoning that supports Proposition 4.2. The main text (Sec. 4.3) only retains the condensed conclusion used by RPCO.

Planning: Deterministic Reward Yields Stable Gradients. The reward signal R_{plan} scores the action a_{plan} directly via a deterministic LLM-based evaluator (which assesses logical coherence and tool matching). Since R_{plan} excludes the stochastic image generation process, environmental noise is

negligible ($\Sigma_\tau \approx 0$). The gradient variance is thus dominated by Σ_a, which gives a stable exploration signal for the optimizer.

Reflection: Stochastic Visual Outcomes Collapse the SNR. The optimization of reflection faces severe challenges due to the specific nature of multi-image tasks. First, unlike single-image generation, multi-image tasks ask for cross-image consistency and narrative structure, which leads to an exponential blow-up of the decision space. Second, the reflection reward R_{reflect} relies on the *post-reflection visual outcome*, which means the conditional trajectory $\tau|a_{\text{reflect}}$ covers the entire generation chain. This ties the reflection action to the high stochasticity of image generation and tool execution, and lands us in a regime where trajectory variance dominates ($\Sigma_\tau \gg \Sigma_a$). The useful signal is drowned out by generation noise, which makes it hard for the agent to tell apart erroneous reflection from the built-in noise of image generation.

Consequently, reflection capabilities become hard to optimize in multi-image tasks, which makes direct transfer from single-image models via RL infeasible. We empirically verify this asymmetry in Sec. 4.3 (with the detailed measurement protocol and per-step table in Appendix D) and use it to motivate the Decouple-then-Fuse paradigm (Sec. 5.1).

F Decouple-then-Fuse Paradigm: Pseudocode

This appendix provides the step-by-step pseudocode of the generic Decouple-then-Fuse paradigm. It is equivalent to the compact formulation (Eqs. 2–4) in Sec. 5.1 and is included for completeness.

Algorithm 1 Decouple-then-Fuse (General Paradigm)

Require: base model π_0; capabilities $\{C_1, \ldots, C_M\}$ with rewards $\{R_1, \ldots, R_M\}$; full environment E

Ensure: co-optimized policy $\pi^\star$

// ==== Stage 1 (Decouple) – Eq. 2 ====

// For each high-noise capability, reduce $\Sigma_{\tau,m}$ via a simpler env to restore SNR.

$\mathcal{S} \leftarrow \emptyset$ ▷ set of specialist policies to be fused

for each capability C_m **do**

  if $\Sigma_{\tau,m} \gg \Sigma_{a,m}$ **then** ▷ high-noise case; needs decoupling

    Pick a simplified environment E_m^{simple} where $\Sigma_{\tau,m}$ is reduced

    $\pi_m^\star \leftarrow \mathrm{RL}\big(\pi_0;\, R_m,\, E_m^{\text{simple}}\big)$ ▷ Eq. 2

    $\mathcal{S} \leftarrow \mathcal{S} \cup \{\pi_m^\star\}$

  end if

end for

// ==== Stage 2 (Fuse) – Eq. 3 ====

// Transfer specialists' strengths via SFT; bypass high-variance RL gradients.

$D \leftarrow \emptyset$ ▷ fused SFT corpus

for each specialist $\pi_m^\star \in \mathcal{S}$ **do**

  Roll out specialist trajectories $T_m \leftarrow \mathrm{Traj}(\pi_m^\star)$

  $D \leftarrow D \cup \mathrm{Filter}(T_m)$ ▷ keep only high-quality trajectories

end for

$\pi_{\text{fused}} \leftarrow \mathrm{SFT}\big(\pi_0;\, D\big)$ ▷ Eq. 3, SFT from the base model π_0

// ==== Stage 3 (Co-Optimize) – Eq. 4 ====

// Joint RL from a balanced prior; only fine-tuning, fewer high-variance steps needed.

$\pi^\star \leftarrow \mathrm{RL}\big(\pi_{\text{fused}};\, \{R_1, \ldots, R_M\},\, E\big)$ ▷ Eq. 4

return $\pi^\star$

Why each stage works (intuitively). *Decouple*: $\Sigma_{\tau,m}$ depends on environment complexity (stochastic steps, horizon length). A simpler E_m^{simple} reduces $\Sigma_{\tau,m}$, which brings the SNR back for C_m. *Fuse*: SFT skips the high-variance RL optimization entirely – the learned capability is transferred via supervised learning, not via noisy gradients. *Co-Optimize*: With strong priors from Stage 2, the model only fine-tunes (not learns from scratch), so it needs fewer gradient steps and less exposure to high-variance regimes.

G RPCO Algorithm: Pseudocode

This appendix provides the step-by-step pseudocode of the RPCO training procedure. It is equivalent to the compact formulation (Eqs. 5–7) in Sec. 5.2 and to the schematic in Fig. 2(b), and is included for reproducibility.

Algorithm 2 Reflection-Plan Co-Optimization (RPCO): Decouple-then-Fuse for $M = 2$ visual capabilities

Require: base model π_0 (Qwen3VL32B); single-image queries Q_s, multi-image queries Q_m; planning expert E_{plan} (Gemini2.5Pro); VLM judge J (Qwen3VL32B); LLM evaluator f_{eval}; active reward sets $W_1 = \{\text{reflect, format, tool, result}\}$, $W_3 = W_1 \cup \{\text{plan}\}$ (Sec. 3.2); group size G

Ensure: VisionCreator-R1 model $\pi^\star$

// ==== Stage 1 (Decouple) – Eq. 5 ====

// Reflection isolation on single-image tasks (E_m^{simple}; low diffusion noise).

SFT sub-stage:

  Roll out $D_{\text{sft}} \leftarrow \text{Traj}(E_{\text{plan}}, Q_s)$ ▷ UTPCR trajectories from Gemini2.5Pro on Q_s

  $\pi_{\text{sft}} \leftarrow \text{SFT}\big(\pi_0;\ D_{\text{sft}}\big)$

GRPO sub-stage:

  $\pi \leftarrow \pi_{\text{sft}}$

for GRPO step $t = 1, \ldots, T_1$ **do**

    Sample a batch of prompts $\mathcal{B}_t \subseteq Q_s$

    for each $q \in \mathcal{B}_t$ **do**

      Roll out G trajectories $\{\tau_1, \ldots, \tau_G\} \sim \pi(\cdot \mid q)$

      For each τ_i: compute R_{reflect} via judge J; compute rule-based $R_{\text{format}}, R_{\text{tool}}, R_{\text{result}}$

      Compute R_{total} with active set W_1 (Sec. 3.2)

    end for

    Update π by one GRPO step with the collected group rewards

end for

$\pi_{\text{SR}} \leftarrow \pi$ ▷ Strong-Reflection model; combined SFT→GRPO matches Eq. 5

// ==== Stage 2 (Fuse) – Eq. 6 ====

// Advantage-complementary SFT; avoids high-variance reflection RL gradients.

Roll out single-img trajectories $T_s \leftarrow \text{Traj}(\pi_{\text{SR}}, Q_s)$

Roll out multi-img trajectories $T_m \leftarrow \text{Traj}(E_{\text{plan}}, Q_m)$

$D_{\text{refl}} \leftarrow \text{Filter}(T_s)$ ▷ keep $R_{\text{plan}}{=}R_{\text{format}}{=}R_{\text{tool}}{=}1$

$D_{\text{plan}} \leftarrow \text{Filter}(T_m)$ ▷ keep $R_{\text{plan}}{=}R_{\text{format}}{=}R_{\text{tool}}{=}1$

$\pi_{\text{RP}} \leftarrow \text{SFT}\big(\pi_{\text{SR}};\ D_{\text{refl}} \cup D_{\text{plan}}\big)$ ▷ Reflection-Plan SFT model; Eq. 6

// ==== Stage 3 (Co-Optimize) – Eq. 7 ====

// Multi-task GRPO on $Q_s \cup Q_m$ from the balanced prior π_{RP}.

$\pi \leftarrow \pi_{\text{RP}}$

for GRPO step $t = 1, \ldots, T_3$ **do**

    Sample a batch of prompts $\mathcal{B}_t \subseteq Q_s \cup Q_m$

    for each $q \in \mathcal{B}_t$ **do**

      Roll out G trajectories $\{\tau_1, \ldots, \tau_G\} \sim \pi(\cdot \mid q)$

      For each τ_i: compute R_{plan} via f_{eval}, R_{reflect} via J, and rule-based $R_{\text{format}}, R_{\text{tool}}, R_{\text{result}}$

      Compute R_{total} with active set W_3 (Sec. 3.2)

    end for

    Update π by one GRPO step with the collected group rewards

end for

$\pi^\star \leftarrow \pi$ ▷ **VisionCreator-R1**; Eq. 7

return $\pi^\star$

H Computational Cost

We train on 24× NVIDIA H20 GPUs (96GB each). Each SFT stage takes ∼8 hours; each RL stage takes ∼3–5 days, where image-generation rollouts dominate cost. Inference cost matches the underlying Qwen3VL32B backbone plus the diffusion tool calls per trajectory.

I Limitations and Broader Impact

Limitations. We discuss three limitations of this work. (i) *Single-image specialist tradeoff*: VisionCreator-R1 (0.532) slightly underperforms the Strong-Reflection specialist (0.543) on single-image tasks (Sec. 7.4); for single-image-only deployment, the Stage-1 specialist is still a stronger choice. (ii) *Bootstrapping dependency*: our Stage-2 planning trajectories are distilled from Gemini2.5Pro; while open-source planning experts can in principle replace it, we have not confirmed this substitution. (iii) *Cross-domain validation pending*: while we propose Decouple-then-Fuse as a general paradigm (Sec. 5.1), this paper validates it only on visual generation. Extending the empirical study to reasoning+tool-use and dialogue+retrieval is left to future work.

Broader Impact: Applicability of Decouple-then-Fuse beyond Visual Generation. The Decouple-then-Fuse paradigm (Sec. 5.1) applies whenever the asymmetry condition holds, i.e., some capability has $\Sigma_{\tau,m} \gg \Sigma_{a,m}$ while a co-trained capability has $\Sigma_{\tau,m'} \approx 0$. Two candidate domains we hypothesize:

(a) Reasoning + Tool Use. Reasoning rewards are deterministic ($\Sigma_\tau \approx 0$), but tool-use rewards depend on stochastic API responses, network state, and external service output ($\Sigma_\tau \gg \Sigma_a$). A single-tool simplified environment can act as E_m^{simple} to decouple tool-use RL from multi-tool noise.

(b) Dialogue + Retrieval. Dialogue quality can be judged from text alone ($\Sigma_\tau \approx 0$), while retrieval rewards depend on stochastic corpus/search results ($\Sigma_\tau \gg \Sigma_a$). A fixed or small-scale corpus can act as E_m^{simple} to decouple retrieval.

Negative impact. The agent could be misused to create misleading images. It calls already-released tools and keeps their original safety constraints.

NeurIPS Paper Checklist

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 Sec. 1 precisely state our three contributions: (i) identifying the reflection-plan conflict via empirical measurement and the structural variance-asymmetry analysis (Sec. 4, 4.3); (ii) the Decouple-then-Fuse paradigm and its RPCO instantiation (Sec. 5.1, 5.2); and (iii) the VCR-SFT/VCR-RL/VCR-Bench resources (Sec. 6). All claims are backed by the theoretical result in Theorem 4.1 and the empirical results in Sec. 7.

   Guidelines:

   - The answer NA 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 NA 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: We dedicate Appendix I to limitations, discussing (i) the single-image specialist tradeoff, (ii) the bootstrapping dependency on Gemini2.5Pro for Stage-2 planning trajectories, and (iii) that cross-domain validation of Decouple-then-Fuse (reasoning+tool-use, dialogue+retrieval) is left as future work.

   Guidelines:

   - The answer NA 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: [Yes]

   Justification: Theorem 4.1 (gradient-variance decomposition) and Proposition 4.2 (structural asymmetry) are formally stated in Sec. 4.3, with all assumptions made explicit. The full rigorous derivation is given in Appendix C, and the empirical verification protocol with the per-step table appears in Appendix D.

   Guidelines:

   - The answer NA 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 fully describe the backbone (Qwen3VL32B), the image-tool (Qwen-Image-Edit-2509-Lightning-4steps-V1.0), the UTPCR tag schema, the three-stage RPCO training pipeline with all reward definitions (Sec. 3.2, 5.2), the two dataset curation procedures (Sec. 6.1), the evaluation judges (Gemini2.5Pro and Qwen3VL235B) and benchmark construction (Sec. 6.2), and the compute setup (Appendix H). Pseudocode for the two algorithms is provided in Appendices F and G.

   Guidelines:

   - The answer NA 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: [No]

   Justification: We will release the training code, VCR-SFT, VCR-RL, VCR-Bench, and model checkpoints with the camera-ready version. The paper provides full architectural, algorithmic, and data-construction details (Sec. 5.2, 6, Appendices F, G) so that the results can be reproduced from the description alone.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.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://nips.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, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We specify the base model (Qwen3VL32B), the image tool, a 6-point plan reward (N=6), the SFT/GRPO rollout setup (G=4 per prompt), the three training stages (Strong-Reflection SFT, Reflection-Plan Co-optimization SFT, RPCO RL), the baselines, and the two independent VLM judges used for evaluation (Sec. 7). Reward definitions and their weights are given in Sec. 3.2 and Appendix B.

   Guidelines:

   - The answer NA 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: Each RL stage takes 3–5 days on 24×H20 GPUs (Appendix H), which makes multi-seed training prohibitively expensive. The main benchmark is evaluated under *two independent VLM judges* (Gemini2.5Pro and Qwen3VL235B) in Tab. 3, and the gains are further corroborated by an independent pairwise human evaluation; both consistently favor our method.

   Guidelines:

   - The answer NA 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: Appendix H reports the compute: 24×NVIDIA H20 GPUs (96 GB each), with ∼8 hours per SFT stage and ∼3–5 days per RL stage (image-generation rollouts dominate cost). Inference cost matches the Qwen3VL32B backbone plus the per-trajectory diffusion tool calls.

   Guidelines:

   - The answer NA 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: We have reviewed the NeurIPS Code of Ethics and confirm full compliance. The work uses publicly available models and image corpora, does not involve human subjects beyond anonymous pairwise preference annotation on synthetic images, and reports societal impact in Appendix I.

Guidelines:

- The answer NA 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: Appendix I notes limitations, a possible transfer to other agent tasks, and the misuse risk of misleading images, which is bounded by the upstream tools' existing safeguards.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA 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., pretrained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: Our agent composes an already-released image-generation tool (Qwen-Image-Edit-2509-Lightning) with an already-released VLM backbone (Qwen3VL32B), both of which carry their own upstream safety measures. VCR-SFT/VCR-RL/VCR-Bench contain task instructions and reflection checkpoints without scraped unsafe imagery, so no additional safeguards are required beyond those of the upstream assets.

Guidelines:

- The answer NA 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: We properly cite all existing assets used: the base model Qwen3VL32B, the image tool Qwen-Image-Edit-2509-Lightning [24], the judge Gemini2.5Pro [25], the GEdit-Bench [13], and the MultiBanana benchmark [14]. All assets are used in accordance with their original licenses for non-commercial research.

    Guidelines:

    - The answer NA 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: [Yes]

    Justification: We introduce three new assets: VCR-SFT (UTPCR-format supervised trajectories), VCR-RL (query plus reflection-checkpoint training data), and VCR-Bench (standardized agentic visual-generation benchmark). Their construction, composition, filtering criteria, task distribution, and evaluation protocol are documented in Sec. 6.1–6.2 and Fig. 5. They will be released with the camera-ready version under a permissive research license.

    Guidelines:

    - The answer NA 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: [No]

Justification: Tab. 3 uses a paid crowdsourced pairwise evaluation on synthetic images. The paper summarizes the guideline and pay, but does not include the full manual or screenshots.

Guidelines:

- The answer NA 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: The crowdsourced pairwise preference annotation collects no personal or sensitive information from participants, presents only synthetic images with no known risk to annotators, and follows the crowdsourcing platform's standard terms of service and minimum-wage policy. Under our institution's guidelines this type of quality-assessment annotation does not require IRB review.

Guidelines:

- The answer NA 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.