RewardFlow: Generate Images by Optimizing What You Reward

Supplementary Material

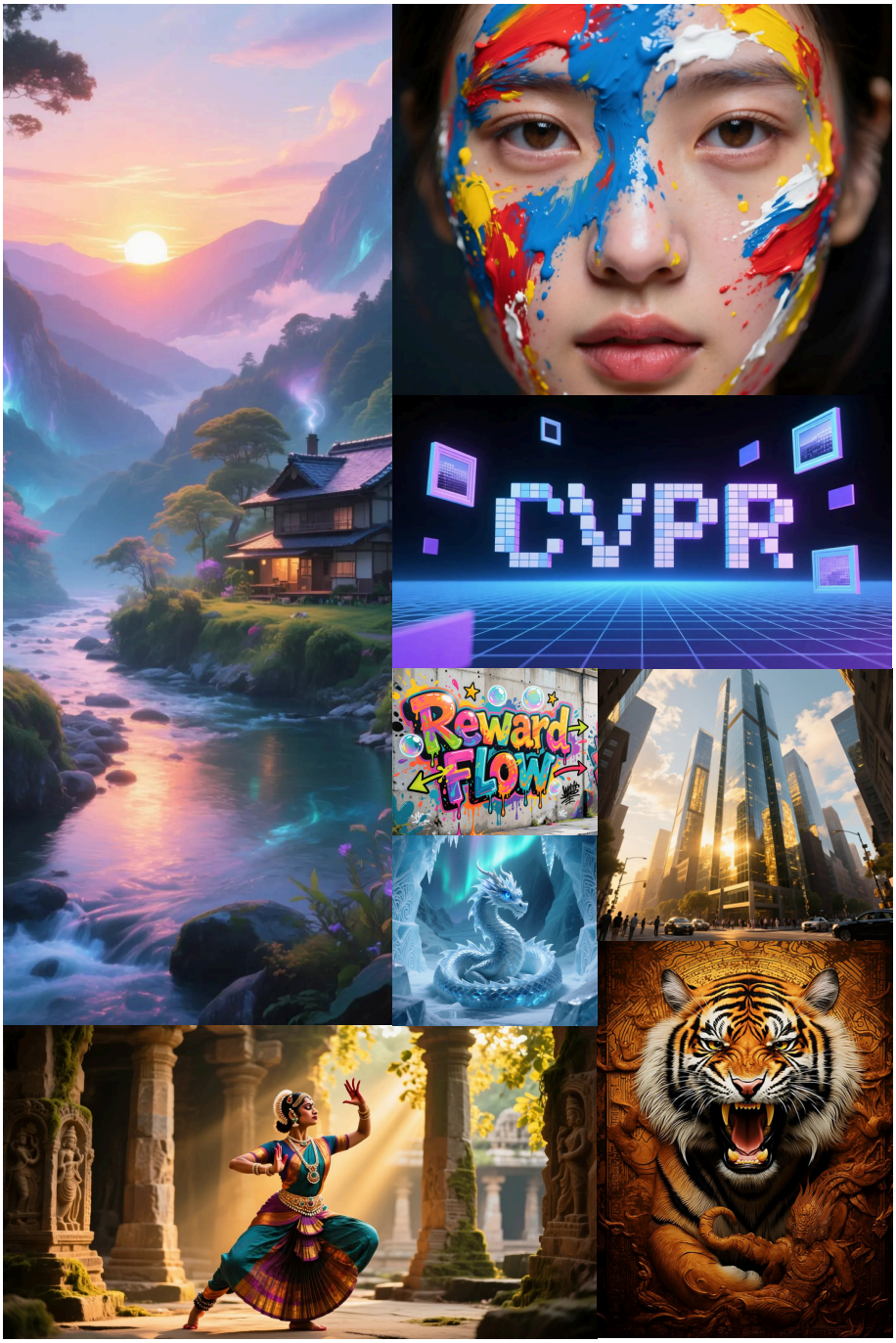

Figure 8. **High-resolution images generated by RewardFlow.**

7. SDE Formulation

In this section, we detail the stochastic differential equation (SDE) that grounds the Langevin-style reverse update in Eq. (2), specify the diffusion-strength schedule γ_k, and provide a derivation showing how Eq. (2) emerges from sampling a prompt-tilted latent density.

Prompt-tilted target density. Let $q_t(\boldsymbol{z} \mid \boldsymbol{p})$ denote the unconditional latent distribution at time t for prompt $\boldsymbol{p}$. Given total reward $R_{\text{tot}}(\boldsymbol{z}, t, \boldsymbol{p})$ obtained by combining the differentiable rewards and KL potential

$$K(\boldsymbol{z}, t; \boldsymbol{z}_0) := \tfrac{1}{2} \left\| \tilde{\boldsymbol{z}}(\boldsymbol{z}, t, \boldsymbol{p}) - \boldsymbol{z}_0 \right\|_2^2, \tag{6}$$

we define the *prompt-tilted* target density

$$\rho_t(\boldsymbol{z} \mid \boldsymbol{p}, x) \propto q_t(\boldsymbol{z} \mid \boldsymbol{p}) \exp\Big(\lambda_R R_{\text{tot}}(\boldsymbol{z}, t, \boldsymbol{p}) - \lambda_{\text{KL}} K(\boldsymbol{z}, t; \boldsymbol{z}_0)\Big). \tag{7}$$

where $\boldsymbol{z}_0 = \text{Enc}(\boldsymbol{x})$ denotes the clean latent of the (optional) source image $\boldsymbol{x}$. Taking the gradient of the log-density in Eq. (7) yields

$$\begin{aligned} \nabla_{\boldsymbol{z}} \log \rho_t(\boldsymbol{z} \mid p, x) = \nabla_{\boldsymbol{z}} \log q_t(\boldsymbol{z} \mid p) \\ + \lambda_R \nabla_{\boldsymbol{z}} R_{\text{tot}}(\boldsymbol{z}, t, \boldsymbol{p}) \\ - \lambda_{\text{KL}} \nabla_{\boldsymbol{z}} K(\boldsymbol{z}, t; \boldsymbol{z}_0). \end{aligned}$$

Each reward $R_i(I, p)$ is defined in image space. For step k, let $g_{\boldsymbol{I},i}^{(k)} := \nabla_{\boldsymbol{I}} R_i(I^{(k)}, p)$ denote the image-space gradient. Using the decoder and denoiser Jacobians, the reward drift in Eq. (1) can be written as

$$g_{R_i,k} = \lambda_R \, J_{\text{Den}}(\boldsymbol{z}^{(k)}, t_k, \boldsymbol{p})^\top J_{\text{Dec}}(\tilde{\boldsymbol{z}}^{(k)})^\top \, g_{\boldsymbol{I},i}^{(k)}. \tag{8}$$

Summing over rewards then yields the fused reward drift $g_{R\text{tot},k} = \lambda_R \nabla_{\boldsymbol{z}^{(k)}} R_{\text{tot}}(\boldsymbol{z}^{(k)}, t_k, p)$. In addition, differentiating $K(\boldsymbol{z}^{(k)}, t_k; \boldsymbol{z}_0)$ with respect to $\boldsymbol{z}^{(k)}$ yields Eq. (5), so $g_{\text{KL},k} = -\lambda_{\text{KL}} \nabla_{\boldsymbol{z}^{(k)}} K(\boldsymbol{z}^{(k)}, t_k; \boldsymbol{z}_0)$.

7.1. Langevin SDE and Discrete Update

We introduce an *algorithmic time* variable $s \in [0, S]$ and a monotone schedule $t(s)$ from algorithmic time to diffusion time, with $t(0) = \bar{t}$ and $t(S) = 0$. We consider the overdamped Langevin SDE whose stationary distribution at each t is the prompt-tilted density ρ_t:

$$\mathrm{d}\boldsymbol{z}_s \;=\; \nabla_{\boldsymbol{z}} \log \rho_{t(s)}\big(\boldsymbol{z}_s \mid p, x\big)\, \mathrm{d}s \;+\; \sqrt{2\,\gamma(s)}\, \mathrm{d}W_s, \tag{8}$$

where W_s is standard Brownian motion and $\gamma(s) > 0$ controls the diffusion strength.

Let $s_0 < s_1 < \cdots < s_K$ be a discretization of $[0, S]$ with step sizes $\eta_k = s_{k+1} - s_k$, and write $t_k = t(s_k)$, $\gamma_k = \gamma(s_k)$, and $\boldsymbol{z}^{(k)} \approx \boldsymbol{z}_{s_k}$. Applying Euler–Maruyama to Eq. (8) yields

$$\begin{aligned} \boldsymbol{z}^{(k+1)} = \boldsymbol{z}^{(k)} + \eta_k \nabla_{\boldsymbol{z}} \log \rho_{t_k}\big(\boldsymbol{z}^{(k)} \mid p, x\big) \\ + \sqrt{2\,\gamma_k \eta_k}\, \xi_k, \qquad \xi_k \sim \mathcal{N}(0, \boldsymbol{I}). \end{aligned} \tag{9}$$

Substituting Eq. (7) and using flow-matching score approximation $v_\theta(\boldsymbol{z}, t, \boldsymbol{p}) \approx \nabla_{\boldsymbol{z}} \log q_t(\boldsymbol{z} \mid p)$ we obtain

$$\begin{aligned} \nabla_{\boldsymbol{z}} \log \rho_t(\boldsymbol{z} \mid p, x) \approx v_\theta(\boldsymbol{z}, t, \boldsymbol{p}) + g_{R_{\text{tot}}}(\boldsymbol{z}, t, \boldsymbol{p}) \\ + g_{\text{KL}}(\boldsymbol{z}, t; \boldsymbol{z}_0). \end{aligned} \tag{10}$$

Evaluating this at $(\boldsymbol{z}^{(k)}, t_k)$ in Eq. (9) yields

$$\begin{aligned} \boldsymbol{z}^{(k+1)} = \boldsymbol{z}^{(k)} + \eta_k \Big(f_k + g_{R\text{tot},k} + g_{\text{KL},k} \Big) + \xi_k, \\ \xi_k \sim \mathcal{N}\big(0, 2\gamma_k \eta_k \boldsymbol{I}\big), \end{aligned} \tag{11}$$

where $f_k := v_\theta(\boldsymbol{z}^{(k)}, t_k, \boldsymbol{p})$ is the backbone drift. This is exactly the stochastic update stated in Eq. (2) of the main paper, now seen as an Euler–Maruyama discretization of the Langevin SDE in Eq. (9) targeting the prompt-tilted density in Eq. (7), with the reward terms defining the controllability potential and the KL tether stabilizing identity and layout.

7.2. Noise variance schedule γ_k

The Gaussian perturbation in Eq. (2) is parameterized by the time-dependent variance γ_k. We instantiate γ_k with a monotonically decreasing schedule

$$\begin{aligned} \gamma_k = \gamma_{\min} + \big(\gamma_{\max} - \gamma_{\min}\big) \left(\frac{t_k}{\bar{t}} \right)^\rho, \\ \text{where } \gamma_{\min}, \gamma_{\max} > 0,\ \rho > 0, \end{aligned} \tag{12}$$

so that early steps at high noise levels ($t_k \approx \bar{t}$) use larger diffusion (exploration), while late steps near $t_k \approx 0$ use smaller diffusion, focusing on refinement. In the special case $\gamma_{\min} = \gamma_{\max}$, Eq. (12) reduces to a constant-noise Langevin sampler.

8. Datasets and Evaluation

To ensure a fair comparison, we adopt the same evaluation protocols and metrics as defined in the original papers of each respective dataset.

T2I-CompBench. T2I-CompBench is a large-scale benchmark designed to evaluate compositional text-to-image generation in open-world settings, and consists of approximately 6,000 prompts categorized into three key tasks: *attribute binding*, *object relationships*, and *complex compositions*. Each prompt describes scenes with multiple objects and attributes, requiring precise alignment between textual semantics and visual structure. For evaluation, the benchmark employs a set of compositional metrics. Attribute binding is assessed using BLIP-based VQA that queries each object's attribute independently (*e.g.*, "What color is the bench?"). Spatial relations are evaluated using UniDet, a pre-trained object detector, to check the relative positioning of objects via bounding box analysis. For complex scenes, a compositional consistency score is computed by aggregating CLIPScore, BLIP-VQA accuracy, and UniDet spatial relation correctness. This framework enables a detailed understanding of how well models handle fine-grained compositional constraints beyond conventional image-text similarity.

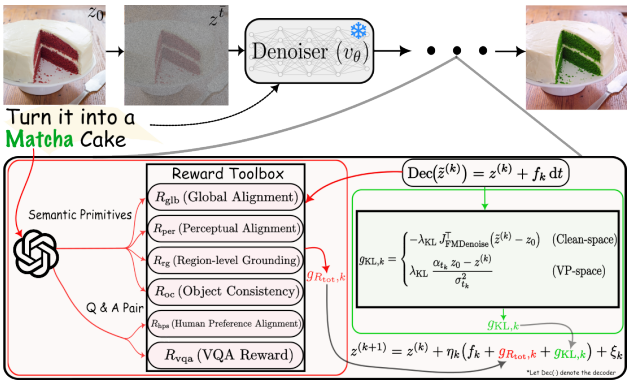

Figure 9. **Overview of the RewardFlow framework.**

GENEVAL. GENEVAL is a structured evaluation suite targeting fine-grained text-to-image alignment at the object level. It introduces prompts designed to probe a model's ability to generate images with correct object *presence*, *co-occurrence*, *counting*, *spatial arrangement*, and *color attribution*. Each generated image is evaluated using automated pipelines based on pre-trained vision models. Object detectors verify the existence and number of instances for specified entities, while spatial metrics assess whether objects appear in the correct geometric configuration (*e.g.*, left/right or above/below). Color attributes are checked by segmenting object regions and comparing predicted colors with prompt specifications. Each task yields binary correctness judgments, and the results are reported as per-category accuracies along with an overall compositional accuracy score. GenEval has been shown to correlate strongly with human judgments and helps isolate specific failure modes such as incorrect object counts or attribute swaps.

PIE-BENCH. PIE-BENCH is a comprehensive benchmark for evaluating text-guided image editing systems. It comprises 700 real-world and artistic images, each paired with a *source prompt*, a *target prompt*, a natural language *editing instruction*, and a binary *editing mask*. The edits are drawn from ten categories, including object addition, removal, replacement, attribute changes (*e.g.*, color, pose), material substitution, background edits, and global style transformations. The benchmark evaluates two core criteria: (1) **Edit Fidelity**, which measures how well the edited image aligns with the target prompt, typically using CLIPScore or similar semantic similarity metrics; and (2) **Content Preservation**, which assesses how much of the non-edited image content remains unchanged, computed via PSNR or SSIM on unmasked (non-edit) regions. PIE-Bench allows for quantitative and targeted assessment of how effectively models perform localized or global edits while preserving image realism and structure.

9. Implementation Details

In this section, we provide additional implementation details for RewardFlow. An overview of the method is illustrated in Figure 9. Unless otherwise stated, we use the same hyperparameters across all backbones, datasets, and tasks. All experiments are run on a single node with 2× NVIDIA A100 GPUs (80 GB each). We implement RewardFlow in PyTorch with automatic mixed precision (AMP) for all backbones and reward networks, which reduces memory footprint and latency without affecting visual quality. Unless otherwise noted, we use a batch size of 1 per GPU for editing experiments and 2 for text-to-image generation.

Backbones and Resolution. We instantiate RewardFlow on three pretrained flow-matching / diffusion backbones: PixArt-α, Flux, and a Qwen-based latent diffusion model. All images are generated and edited at 1024×1024 resolution. We use the official checkpoints and sampling schedules for each backbone and do not fine-tune any model weights; RewardFlow operates purely at inference time. For image editing, given a prompt $\boldsymbol{p}$ and source image $\boldsymbol{x}$, we encode the image into a clean latent $\boldsymbol{z}_0 = \mathrm{Enc}(\boldsymbol{x})$, initialize a noisy latent $\boldsymbol{z}^{(0)}$ at a fixed noise level $\bar{t}$ as in the backbone, and run $K = 35$ reverse steps following the update in Eq. (2). For unconditional text-to-image generation, $\boldsymbol{z}_0$ is sampled from the backbone's prior and the KL tether is disabled ($\lambda_{\mathrm{KL}} = 0$).

Prompt Parsing and Semantic Primitives. Before sampling, we parse each prompt $\boldsymbol{p}$ once using GPT-5 to extract:

- A set of Semantic Primitives $\mathrm{SP}(\boldsymbol{p}) = \{p_m\}_{m=1}^{M}$, where each p_m is a short, atomic instruction (*e.g.*, "remove cap", "add sunglasses", *etc.*).
- A small set of VQA pairs $\{(q_j, a_j^\star)\}_{j=1}^{J_{\text{vqa}}}$ that probe fine-grained aspects of the intended edit (*e.g.*, "What is on the person's head?" → "Nothing").

As shown in Figure 10 prompt template, we prompt the model to ensure that each SP is self-contained and that the VQA questions are answerable from the final image without ambiguity. This one-time parsing step is performed offline and cached for all subsequent sampling runs with the same prompt. For multi-instruction prompts, SPs prevent interference between unrelated objectives and enable per-primitive reward computation.

Rewards and Feature Extractors. At every denoising step, each reward is evaluated on $\boldsymbol{I}^{(k)}$ and the corresponding SPs, producing both a scalar score and an image-space gradient. We briefly summarize implementation choices for each.

Global and perceptual rewards (R_{glb}, R_{per}). For the global semantic reward R_{glb} we use a SigLIP-style vision–language model $\phi_{\text{img}}^{\text{sig}}, \phi_{\text{text}}^{\text{sig}}$ and compute cosine similarity between the image and each SP:

$$R_{\text{glb}}(\boldsymbol{I}^{(k)}, \boldsymbol{p}) = \cos\big(\phi_{\text{img}}^{\text{sig}}(\boldsymbol{I}^{(k)}), \phi_{\text{text}}^{\text{sig}}(\boldsymbol{p})\big).$$

For the perceptual reward R_{per} we employ a Perception Encoder $\phi_{\text{img}}^{\text{per}}, \phi_{\text{text}}^{\text{per}}$ and cosine similarity. Prompt-level scores $R_{\text{glb}}(\boldsymbol{I}^{(k)}, \boldsymbol{p})$ and $R_{\text{per}}(\boldsymbol{I}^{(k)}, \boldsymbol{p})$ are obtained by aggregating over SPs (uniform averaging modulated by the policy).

Region grounding reward (R_{rg}). Region-level grounding

Vision-language Editing Assistant.

You are a vision-language assistant. You receive an image and a short edit instruction.

1) Extract short edit prompts: output a compact list of 5–12 atomic, actionable tags/phrases that guide the image edit. Include:

- Visible subject descriptors (pose, angle, clothing items) actually present.
- The edit action(s) and key visual attributes (style, color, size, placement).
- Constraints to preserve identity, lighting, composition, realism, and continuity.
- Any practical rendering notes (alignment, shadows, reflections, edges).

2) Create exactly one Q&A pair focused on the final edited image's appearance.

- Ask **one** question that would most affect the final look (*e.g.*, style, colorway, size/scale, placement, material/finish, mood/lighting continuity).
- Give **one** concise answer based on the image/instruction; if not determinable, answer "Unspecified from image."

Rules

- **Output JSON only** in the exact schema below—no extra text.
- Keep each short prompt <= 6 words; imperative, neutral wording.
- Do not invent details not visible or implied by the instruction.
- Avoid sensitive inferences (*e.g.*, ethnicity, health, etc.).
- American English.

Input

EDIT_INSTRUCTION: `{edit_instruction}`

Output schema (JSON only)

```
{
 "short_prompts":  ["<tag1>", "<tag2>", "..."],
 "qna":  {
    "question":  "<visual-outcome question>",
    "answer":  "<concise answer or 'Unspecified from image'>"
    }
}
```

Figure 10. **Prompt template** used for semantic primitives and $\mathcal{R}_{\text{vqa}}$.

uses RegionCLIP image-region $\psi^{\text{reg}}_{\text{img}}(\boldsymbol{I}, r_m)$ and text embeddings $\psi^{\text{reg}}_{\text{text}}(\boldsymbol{p})$. Given region proposals $\{r_m\}$ we compute $s_m(\boldsymbol{p})$ and soft attention weights $\alpha_m(\boldsymbol{p})$, and define

$$R_{\text{rg}}(\boldsymbol{I}^{(k)}, \boldsymbol{p}) = \sum_m \alpha_m(\boldsymbol{p})\, s_m(\boldsymbol{p}).$$

This reward encourages gradients to concentrate on spatial regions that are both semantically and visually aligned with each SP, matching the behavior illustrated in Figure 9.

Object consistency reward (R_{oc}). For object-level localization, we use text-guided SAM2 [37] (Florence-SAM2[1]) to obtain soft masks $\{\boldsymbol{M}_j\}$ and confidences $\{a_j\}$ for each semantic primitive. For each SP p, we query SAM2 with the text description and optional point prompts derived from its coarse localization (*e.g.*, from the region-level gradients), yielding soft foreground masks M_j and their confidences a_j. Mixture weights ω_j are formed via a softmax over a_j. We compute an object alignment score $F_{\text{obj}}(M_j, g_{\text{SP}}(\boldsymbol{p}))$ that rewards correct semantics in the mask and penalizes leakage in the background. The object reward for SP p is

$$R_{\text{oc}}(\boldsymbol{I}^{(k)}, \boldsymbol{p}) = \sum_j \omega_j\, F_{\text{obj}}(M_j, g_{\text{SP}}(\boldsymbol{p})),$$

and is further modulated by the add/remove intent scalar $s_{\text{obj}} \in [-1, 1]$ predicted by the adaptive policy.

Human Preference Reward (R_{hps}). For R_{hps}, we use HPSv2, a pretrained human preference scorer that takes $(\boldsymbol{I}^{(k)}, \boldsymbol{p})$ as input and outputs a scalar score. We normalize this score with a fixed running mean and variance so that it is numerically comparable to the other rewards and can be combined without further scaling:

$$R_{\text{hps}}(\boldsymbol{I}^{(k)}, \boldsymbol{p}) = \text{norm}\big(H_{\text{HPS}}(\boldsymbol{I}^{(k)}, \boldsymbol{p})\big).$$

In practice, R_{hps} is evaluated on the full prompt and primarily stabilizes overall aesthetic quality and prompt adherence.

VQA reward (R_{vqa}). For R_{vqa} we use the Qwen-2.5-VL 3B model, accessed via the HuggingFace Transformers interface. For each Q&A pair $(q, a^\star)$ produced by ChatGPT, we feed $(\boldsymbol{I}^{(k)}, \boldsymbol{p})$ into Qwen-2.5-VL and obtain the token-level logits $\{\ell_t\}_{t=1}^{T^\star}$ for the answer sequence $a^\star - (a_t^\star)_{t=1}^{T^\star}$. We then form the VQA reward from these logits. In practice, we cap $T^\star$ to a reasonable answer length (*e.g.*, $T^\star \leq 70$ tokens).

[1] https://huggingface.co/spaces/SkalskiP/florence-sam/blob/main/checkpoints/sam2_hiera_large.pt

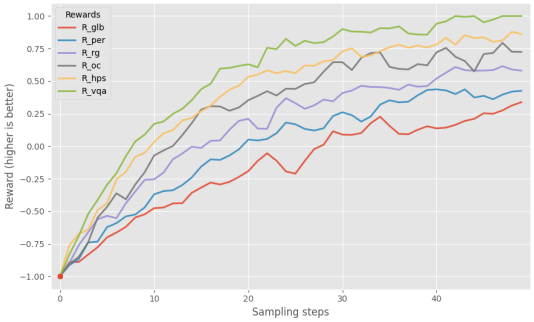

Figure 11. **Reward progression over time.**

Cosine similarity backbone. All rewards except R_{hps} and R_{vqa} are implemented as cosine similarities between the embeddings of semantic primitives and the current image at step k. Gradients are obtained via automatic differentiation through the corresponding vision–language encoders.

KL Tether for Image Editing. For all image editing experiments, we enable the clean-latent KL tether $g_{\text{KL},k}$ from Eq. (5). The tether is computed in the clean latent space $\tilde{z}^{(k)}$ and back-propagated through Den_θ using its Jacobian J_{Den}. We keep the KL strength $\lambda_{\text{KL}} = 1.5$ fixed across steps and applied only when a source image is provided. For pure text-to-image generation, we set $\lambda_{\text{KL}} = 0$ so that the sampler targets the prompt-tilted distribution without anchoring to a particular source latent.

Editing vs. Generation Configurations. For text-to-image experiments, we use all rewards except the object-consistency reward R_{oc}, which is less relevant in the absence of a reference layout. For image editing, we enable the full set of rewards $\{R_{\text{glb}}, R_{\text{per}}, R_{\text{oc}}, R_{\text{rg}}, R_{\text{hps}}, R_{\text{vqa}}\}$, dynamic reward weighting, reward-aware step sizes, and the KL tether. As shown in Figure 11, all reward components in RewardFlow exhibit consistent and stable improvement over the course of sampling. Starting from an initial value of -1, the global semantics reward R_{glb}, perceptual reward R_{per}, region grounding reward R_{rg}, object consistency reward R_{oc}, human-preference reward R_{hps}, and VQA reward R_{vqa} all trend upward with natural fluctuations, eventually converging to high positive values. The smooth yet spiky trajectories indicate that the system is actively exploring while steadily refining the sample quality under each objective, rather than overfitting to any single reward. Taken together, these qualitative dynamics demonstrate that RewardFlow effectively coordinates and optimizes all reward signals, confirming that the full reward pipeline operates as intended.

10. Additional Results

Text-to-Image Generation. We perform additional text-to-image generation evaluation on GENEVAL. As shown in Table 5, RewardFlow consistently improves compositional faithfulness over both backbone models and the ReNO baseline. Starting from weaker backbones such as PixArt-α DMD and FLUX-schnell, RewardFlow lifts the mean score from 0.45→0.65 and 0.64→0.81, respectively, and further improves over ReNO by +0.06 and +0.09 in overall performance. The gains are largest on the most compositional sub-tasks: for PixArt-α, Two objects and Counting increase from 0.38/0.46 to 0.77/0.65, and for Flux from 0.80/0.64 to 0.97/0.90. Even on the strong Qwen backbone, RewardFlow improves the overall performance from 0.83 to 0.91 and surpasses ReNO on all metrics, notably boosting Position from 0.27→0.47 and Color Attribution from 0.71→0.84. As a result, Qwen + RewardFlow achieves the best overall GENEVAL performance, outperforming powerful off-the-shelf models such as SDXL, DALL-E 3, and SD3 (8B), whose mean scores remain in the 0.55–0.68 range.

Table 5. **T2I generation on GENEVAL.**

Model	Overall ↑	Single ↑	Two ↑	Counting ↑	Colors ↑	Position ↑	Color Attribution ↑
SD v2.1	0.50	0.98	0.51	0.44	0.85	0.07	0.17
SDXL	0.55	0.98	0.74	0.39	0.85	0.15	0.23
IF-XL	0.61	0.97	0.74	0.66	0.81	0.13	0.35
PixArt-α	0.48	0.98	0.50	0.44	0.80	0.08	0.07
DALL-E 2	0.52	0.94	0.66	0.49	0.77	0.10	0.19
DALL-E 3	0.67	0.96	0.87	0.47	0.83	0.43	0.45
SD3 (8B)	0.68	0.98	0.84	0.66	0.74	0.40	0.43
(1) PixArt-α DMD	0.45	0.95	0.38	0.46	0.76	0.05	0.09
(1) + ReNO	0.59	0.98	0.72	0.58	0.85	0.15	0.27
(1) + RewardFlow	0.65	0.99	0.77	0.65	0.89	0.21	0.33
(2) Flux	0.64	0.98	0.80	0.64	0.78	0.18	0.43
(2) + ReNO	0.72	0.99	0.90	0.79	0.87	0.21	0.56
(2) + RewardFlow	0.81	0.99	0.97	0.90	0.95	0.39	0.72
(5) Qwen	0.83	0.99	0.98	0.92	0.92	0.27	0.71
(5) + ReNO	0.85	0.99	0.98	0.94	0.95	0.35	0.75
(5) + RewardFlow	0.91	0.99	0.99	0.97	0.98	0.47	0.84

Table 6. **VLM Comparison on PIE-BENCH.**

VLMs	Distance ↓ ($\times 10^3$)	PSNR ↑	LPIPS ↓ ($\times 10^3$)	MSE ↓ ($\times 10^4$)	SSIM ↑ ($\times 10^2$)	Whole ↑	Edited ↑
Qwen 2.5VL 3B†	**7.64**	**32.09**	**38.47**	**33.57**	**90.21**	**29.78**	**27.57**
LLaMa-4-8B	6.57	33.43	37.19	31.31	91.33	30.44	28.82
Qwen 3 Next-34B	6.53	32.34	38.05	32.76	91.49	31.01	29.03

These quantitative gains stem from the way RewardFlow integrates diverse, task-aligned rewards into test-time optimization. Instead of relying primarily on a global alignment signal as in ReNO, RewardFlow evaluates a heterogeneous set of differentiable rewards covering semantic and perceptual alignment, regional and object-level consistency, and QA-style reasoning and fuses their gradients through a prompt-aware adaptive policy that adjusts reward weights and step sizes along the denoising trajectory. This richer, spatially and semantically grounded feedback allows the sampler to correct fine-grained failures such as incorrect counts, swapped colors, or mislocalized objects, while preserving the overall realism of the backbone generator. Consequently, RewardFlow is better able to satisfy complex multi-object, attribute, and localization constraints, which is reflected in its strong improvements on Two objects, Counting, Position, and Color Attribution compared to both unmodified backbones and prior reward-guided baselines.

Ablation on VLMs. We further conduct an ablation study by replacing the visual-language model (VLM) used for R_{vqa} with different architectures. As shown in Table 6, the overall

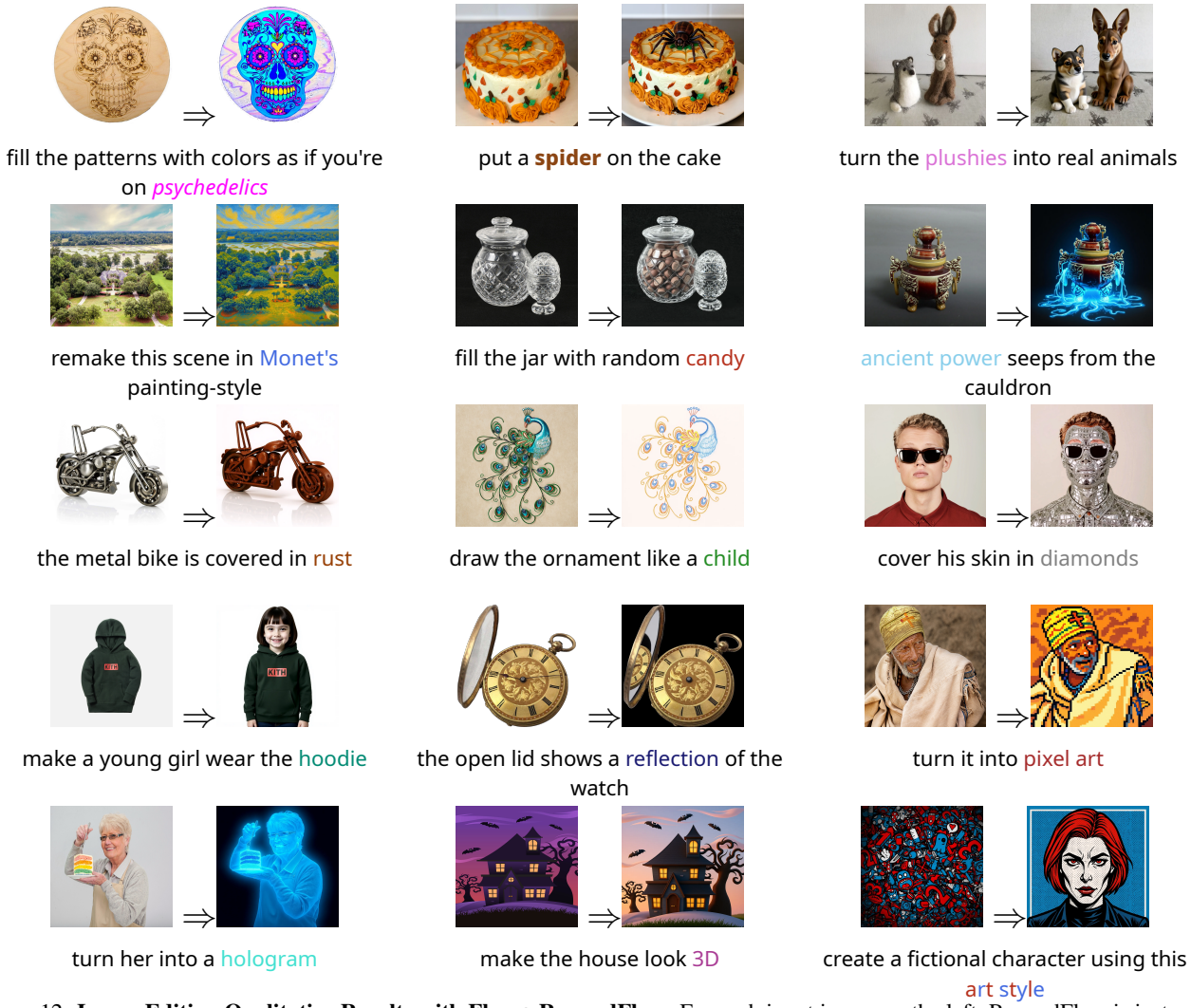

Figure 12. **Image Editing Qualitative Results with Flux + RewardFlow**. For each input image on the left, RewardFlow is instructed to apply a targeted edit (text below), and the right image shows the generated result. Tasks span from global scene modifications and object-level edits to very fine-grained, localized edits.

performance remains relatively stable when scaling from 3B to 8B parameters, indicating that moderate model scaling yields limited benefit for this task. However, substituting with the larger and more recent Qwen3-Next-34B model leads to a noticeable $\sim$7% improvement across most evaluation metrics, suggesting that more capable VLMs enhance semantic reasoning in the reward estimation process, however, at the expense of increased computational overhead.

11. Additional Qualitative Results

Image Editing Qualitative Results. Using Flux as the base model, as shown in Figure 12, RewardFlow follows a wide variety of fine-grained instructions while preserving background layout and image identity. RewardFlow can perform strong stylistic changes, such as recoloring the carved wooden ornament "as if on psychedelics," translating a natural landscape into Monet's painting style, and turning a portrait into pixel art, all while keeping shapes and composition intact. Our proposed method also accurately handles object insertion and modification: a spider is added on top of the cake, plush toys are turned into realistic animals, a jar is filled with random candy, and "ancient power" is made to seep from the cauldron with coherent lighting. Local attribute edits are also precisely localized, *e.g.*, metal parts of the bike are rusted without any corruptions, the ornament is redrawn in a child-like manner, the subject's skin is covered with diamonds, and the pocket watch lid reflects the watch face without hallucinating unrelated content. Finally, RewardFlow successfully performs more abstract edits such as making a young girl wear the same hoodie, turning the woman into a hologram, making the cartoon house appear 3D, and synthesizing a new fictional character inspired

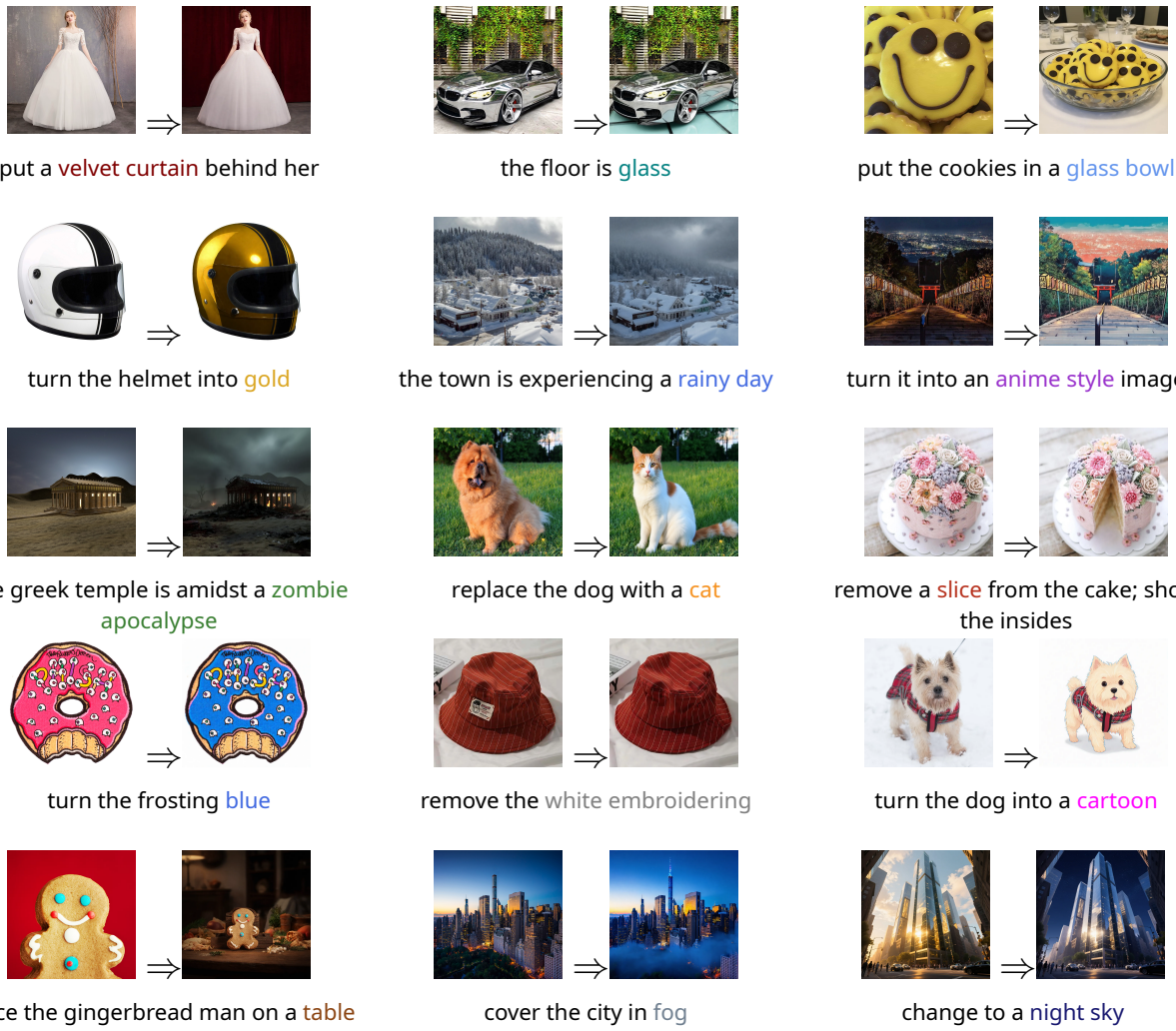

Figure 13. **Image Editing Qualitative Results with Qwen Image + RewardFlow**. For each input image on the left, RewardFlow is instructed to apply a targeted edit (text below), and the right image shows the generated result. Tasks span from global scene modifications and object-level edits to very fine-grained, localized edits.

by a textured input image. Across all examples, edits are restricted to instruction-relevant regions and avoid semantic leakage into the rest of the scene.

With Qwen Image as the backbone, shown in Figure 13, RewardFlow exhibits similarly precise and diverse editing capabilities. Global scene edits include converting a sunny town into a rainy day, covering a city with fog, and changing a bright skyline to a night sky, while preserving camera pose and urban geometry. Attribute and material changes are handled cleanly, *e.g.*, a velvet curtain is placed behind the bride, the showroom floor becomes glass, helmet material is changed to gold, and cake frosting is recolored blue without affecting decorations. RewardFlow also supports challenging object-level manipulations, such as putting cookies into a glass bowl, turning a dog into a cartoon, and placing the gingerbread man onto a table with consistent perspective. Fine, localized modifications, such as removing the embroidered text on the hat, removing a slice from the cake and revealing the inside, and staging a "zombie apocalypse" around a Greek temple, are executed while maintaining sharp structure and coherent lighting. Results demonstrate RewardFlow generalizes across backbones and instruction types, delivering semantically faithful, spatially localized edits from global scene transforms down to pixel-level adjustments.

Figure 14 presents a qualitative comparison between RewardFlow and recent image editing methods, including InfEdit, FlowEdit, FlowChef, InstantEdit, and KV-Edit, under the same input image and text instruction. The figure covers a range of challenging edit types, including material transformation, object-level semantic replacement, and color editing. In the first row, the instruction asks to make the frame of the bike rusty. Baseline methods exhibit different failure modes, *e.g.*, some methods under-edit the image and leave large parts of the bicycle frame nearly unchanged (such as In-

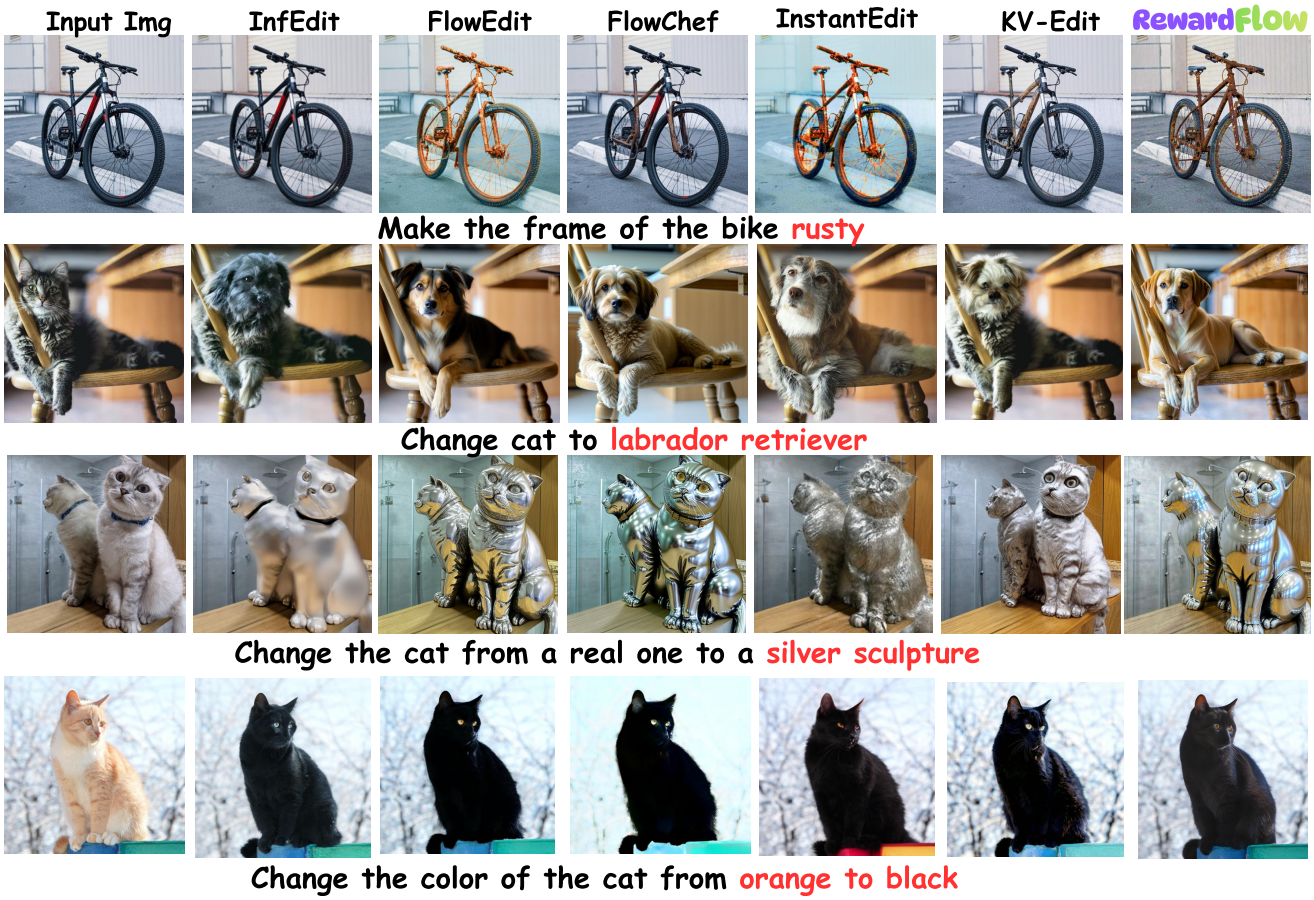

Figure 14. **Image Editing Qualitative Comparisons.** Comparison across a range of challenging edits, such as diverse attribute-, style-, and object-level transformations. Each row shows the source image followed by results from strong baselines and RewardFlow.

fEdit, FlowChef, and KV-Edit), while others apply the rusty texture too aggressively or inconsistently, affecting broader regions and introducing unnatural appearance changes (such as FlowEdit and InstantEdit). In contrast, RewardFlow successfully transfers the rusty material appearance onto the bicycle frame while preserving the overall structure, viewpoint, wheel geometry, and background scene, resulting in a more coherent and realistic edit. In the cat-to-labrador transformation (second row), several baselines either fail to fully realize the target dog breed or generate inconsistent appearances, whereas RewardFlow produces a more convincing labrador retriever that remains in the same position on the chair, while keeping the surrounding environment intact. For the real-cat to silver-sculpture edit, baselines either fail to fully impose the metallic sculptural material or introduce artifacts in shape and surface reflectance, whereas RewardFlow renders metallic texture and reflective highlights, while preserving the original pose, object boundaries, and scene composition. Finally, in the fourth row, baselines sometimes over-darken the image, alter contrast unnaturally, or fail to perform a clean color transition, whereas RewardFlow produces a cleaner black cat while maintaining the cat's silhouette, eye color, and overall scene context.

Figure 15. **RewardFlow counting failure case.**

Failure Modes. While robust, RewardFlow is bounded by its components. A primary failure mode arises from VQA limitations in fine-grained reasoning like counting. As shown in Figure. 15, if VQA model fails to accurately count small objects, reward signal becomes uninformative.

Text-to-Image Generation Qualitative Results. Figure 16 presents qualitative comparisons for text-to-image generation with the Flux backbone under three inference settings:

vanilla Flux, Flux guided by a global matching reward (Flux + GlobalReward), and the full reward-augmented model, RewardFlow (Flux + RewardFlow). Across a diverse set of prompts, including a chef portrait in a restaurant kitchen, a street-fashion scene in nighttime Tokyo, a multi-person family cooking scene, and a culturally specific festival portrait, the vanilla backbone generally captures the coarse scene semantics but frequently under-specifies fine-grained attributes, weakens environmental grounding, and exhibits limited compositional precision. Incorporating only the global reward improves overall prompt alignment and image aesthetics, yet the generations still miss localized details and precise relational cues, particularly in clothing structure, scene context, object placement, and human interaction. In contrast, RewardFlow consistently produces samples with stronger semantic fidelity, improved spatial and contextual grounding, and higher perceptual coherence. In the chef example, RewardFlow better realizes the warm kitchen environment, apron texture, flour details, and realistic skin appearance. In the Tokyo street scene, RewardFlow more faithfully captures the wet-pavement reflections, while in the family cooking example, RewardFlow yields more natural multi-person interaction, and better localized food and countertop details. In the festival portrait, RewardFlow more accurately renders traditional Indian attire through richer embroidery, more convincing jewelry, and a stronger festive lighting atmosphere.

Algorithm 1 RewardFlow: Prompt-aware multi-reward Langevin editing

Input: image $\boldsymbol{x}$, prompt $\boldsymbol{p}$, steps K
$\boldsymbol{z}_0 \leftarrow \mathrm{Enc}(\boldsymbol{x})$
$\{\boldsymbol{p}_m\}_{m=1}^{M} \leftarrow \textsc{ExtractSPs}(\boldsymbol{p})$ {semantic primitives}
$(q, a^\star) \leftarrow \textsc{MakeQA}(\boldsymbol{x}, \boldsymbol{p})$ {fixed once}
Initialize running stats $\{\mu_i, \sigma_i\}_i$ for all heads
Sample $\varepsilon \sim \mathcal{N}(0, \boldsymbol{I})$, set $\boldsymbol{z}^{(0)} \leftarrow \alpha_{\bar{t}}\boldsymbol{z}_0 + \sigma_{\bar{t}}\varepsilon$, $t_0 \leftarrow \bar{t}$
for $k = 0$ **to** $K-1$ **do**
  $\tilde{\boldsymbol{z}}^{(k)} \leftarrow \mathrm{Den}(\boldsymbol{z}^{(k)}, t_k, \boldsymbol{p})$
  $\boldsymbol{I}^{(k)} \leftarrow \mathrm{Dec}(\tilde{\boldsymbol{z}}^{(k)})$
  {SP-wise rewards: compute over $\boldsymbol{p} \in \mathrm{SP}(\boldsymbol{p})$}
  Initialize score vectors: $\mathbf{v}_{\text{glb}}, \mathbf{v}_{\text{per}}, \mathbf{v}_{\text{rg}}, \mathbf{v}_{\text{oc}} \leftarrow [\,]$
  for each $\boldsymbol{p} \in \mathrm{SP}(\boldsymbol{p})$ **do**
    $\mathbf{v}_{\text{glb}}$.append($\textsc{GlobReward}(\boldsymbol{I}^{(k)}, \boldsymbol{p})$)
    $\mathbf{v}_{\text{per}}$.append($\textsc{PercReward}(\boldsymbol{I}^{(k)}, \boldsymbol{p})$)
    $\mathbf{v}_{\text{rg}}$.append($\textsc{RegionReward}(\boldsymbol{I}^{(k)}, \boldsymbol{p})$)
    $\mathbf{v}_{\text{oc}}$.append($\textsc{ObjReward}(\boldsymbol{I}^{(k)}, \boldsymbol{p})$)
  end for
  {Adaptive fusion of SP scores per head}
  for $h \in$ {glb, per, rg, oc} **do**
    $\boldsymbol{\ell}_h \leftarrow \textsc{ComputeWeights}(\mathbf{v}_h, t_k, \mathrm{SP}(\boldsymbol{p}))$
    $\boldsymbol{\omega}_h \leftarrow \mathrm{softmax}(\boldsymbol{\ell}_h)$
    $R_h^{(k)} \leftarrow \sum_j \boldsymbol{\omega}_h[j] \cdot \mathbf{v}_h[j]$ {weighted fusion}
  end for
  {Prompt-wise rewards: computed once per step}
  $R_{\text{hps}}^{(k)} \leftarrow \textsc{HPSReward}(\boldsymbol{I}^{(k)}, \boldsymbol{p})$
  $R_{\text{vqa}}^{(k)} \leftarrow \textsc{VQAReward}(\boldsymbol{I}^{(k)}, q, a^\star)$
  {Normalize each head}
  for each head i **do**
    Update μ_i, σ_i; $\bar{R}_i^{(k)} \leftarrow (R_i^{(k)} - \mu_i)/(\sigma_i + \epsilon)$
  end for
  {Prompt-aware adaptive weights (Sec. 3.2)}
  $\{w_i^{(k)}\}_i \leftarrow \textsc{ComputeWeights}\big(\{\bar{R}_i^{(k)}\}, t_k, \boldsymbol{p}\big)$
  $R_{\text{tot}}^{(k)} \leftarrow \sum_i w_i^{(k)} \bar{R}_i^{(k)}$
  {Fused reward gradient in latent space (Sec. 3.1)}
  $g_{\boldsymbol{I}}^{(k)} \leftarrow \sum_i w_i^{(k)} \nabla_{\boldsymbol{I}^{(k)}} \bar{R}_i^{(k)}$
  $g_{R_{\text{tot}},k} \leftarrow \lambda_R J_{\mathrm{Den}}(\boldsymbol{z}^{(k)}, t_k, \boldsymbol{p})^\top J_{\mathrm{Dec}}(\tilde{\boldsymbol{z}}^{(k)})^\top g_{\boldsymbol{I}}^{(k)}$
  {Backbone drift and KL tether (Sec. 3.4)}
  $f_k \leftarrow v_\theta(\boldsymbol{z}^{(k)}, t_k, \boldsymbol{p})$
  $g_{\text{KL},k} \leftarrow -\lambda_{\text{KL}}\, J_{\mathrm{Den}}(\boldsymbol{z}^{(k)}, t_k, \boldsymbol{p})^\top (\tilde{\boldsymbol{z}}^{(k)} - \boldsymbol{z}_0)$
  {Reward-aware step size (Sec. 3.2)}
  $\eta_k \leftarrow \textsc{StepSize}(R_{\text{tot}}^{(k)})$; $\xi_k \sim \mathcal{N}(0, \boldsymbol{I})$
  {Langevin update}
  $\boldsymbol{z}^{(k+1)} \leftarrow \boldsymbol{z}^{(k)} + \eta_k(f_k + g_{R_{\text{tot}},k} + g_{\text{KL},k}) + \sqrt{2\gamma(t_k)\eta_k}\xi_k$
  $t_{k+1} \leftarrow t_k - \eta_k$
end for
return $\hat{\boldsymbol{I}} \leftarrow \mathrm{Dec}\left(\mathrm{Den}_\theta(\boldsymbol{z}^{(K)}, 0, \boldsymbol{p})\right)$

Figure 16. **Text-to-image qualitative results with Flux as backbone.** Qualitative comparison of Flux, Flux + Global Reward, and Flux + RewardFlow across diverse prompts.