AURA: An Autonomous Retouching Agent with Photographic Visual Thinking

Shuaizheng Liu[1,2*] **Fangzhou Han**[2*] **Jiarong Liao**[2*] **Yujing Sun**[1,2] **Lingchen Sun**[1,2]
Ruibin Li[1] **Xinyu Wei**[1,2] **Jie Liang**[2] **Hui Zeng**[2] **Xingdong Zhang**[2] **Lei Zhang**[1,2†]

[1]The Hong Kong Polytechnic University [2]OPPO Research Institute

Abstract

Professional image retouching is a sequential and reasoning-driven process. Photographers first analyze scene semantics, lighting structure, and depth cues to design a retouching plan, and then progressively apply global tone grading and mask-guided local corrections based on the evolving visual outcome. Recent MLLM-empowered retouching agents either predict global parameters or respond to user-provided retouching instructions. Although showing impressive results, they lack the capacity for autonomous sequential reasoning and fall short of fine-grained local control. We propose **AURA**, an **AU**tonomous **R**etouching **A**gent that emulates the workflow of expert photographers. Given an input photograph, AURA autonomously assesses the scene and proceeds progressively. At each step, it reasons about what to retouch, executes the adjustment, and feeds the result back as visual context for next-step reasoning, forming a fine-grained visual chain-of-thought that evolves with the image. To support AURA, we contribute the first expert-annotated long-horizon retouching trajectory dataset, where expert photographers edit the input image from scratch with deliberate artistic intent. Based on this dataset, we train AURA in two stages: supervised fine-tuning to learn the retouching trajectory, followed by GRPO-retouching, an agentic reinforcement learning stage that optimizes for perceptual quality and tool accuracy. Our experiments demonstrate that AURA consistently outperforms existing methods in both quantitative metrics and user studies, enabling one-click professional retouching and democratizing masterpiece-quality retouching previously accessible only to skilled experts. The dataset, model, and code are publicly available at `https://github.com/PolyU-VCLab/AURA`.

1 Introduction

Image retouching [5, 13, 18, 44, 23] adjusts the tone and color of a given image, turning it into a visually compelling masterpiece. However, professional-grade retouching typically requires the mastery of specialized tools, such as Lightroom [2] or Photoshop [1], along with dedicated artistic skills and significant time investment. To democratize this process, early works [16, 14, 44, 40] learn to predict global editing parameters, but without explicit understanding of image content. More recent works [24, 25, 38, 12, 42] have leveraged MLLMs [39] to build retouching agents that invoke professional editing software such as Lightroom and GIMP [34], translating visual understanding into editing operation parameters. These agents either reason over global editing parameters [12, 9], or translate user-specified directives into tool parameters to perform adjustments [24, 25, 38]. Despite the impressive progress, these methods share an implicit view of retouching as color/style transformation. Global methods regress a single stylistic tone mapping, while instruction-based agents rely on user instructions to specify the desired color style.

*Equal contribution.

†Corresponding author. This research is supported by the PolyU-OPPO Joint Innovative Research Center.

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

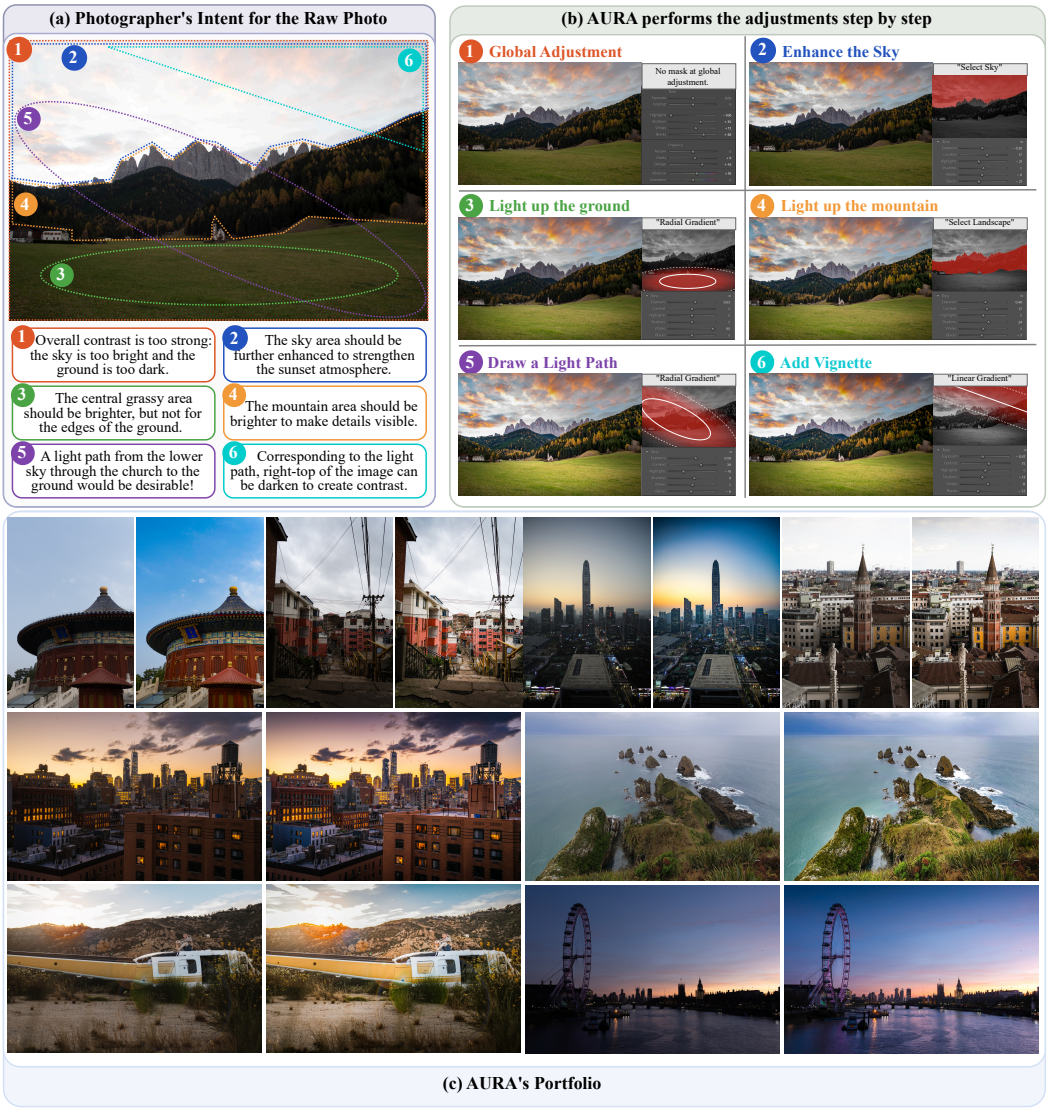

Figure 1: (a) Professional retouching follows a scene-specific design logic implemented with local operations. For this photo, the photographer plans to enhance the sunset sky, brighten the central ground, and draw a light path through the church while darkening the opposite corner for contrast. (b) AURA autonomously analyzes the scene to derive such a design plan, then executes it step by step by invoking Lightroom tools, from a global adjustment to local edits on semantic and non-semantic regions. (c) AURA's retouching results, which build a clear tonal hierarchy of light and shadow to make the subject stand out while preserving natural color, rather than a filter-like style shift.

In practice, professional retouching is not simply color transformation, but a hierarchical scene-aware decision-making process. Expert photographers begin by interpreting a scene's light distribution, spatial depth, and compositional structure to determine which elements to emphasize and suppress, and what tonal atmosphere the image should ultimately convey. They then evaluate the global tone and progressively reshape the illumination, redirect visual attention, and adjust the image regions. Every decision is grounded in explicit reasoning over the current visual state, evolving as the image transforms step by step. Among these decisions, region-level operations are particularly demanding. Each operation involves two coupled decisions: selecting a target region via a masking tool and adjusting its tonal or color parameters. Critically, region selections extend well beyond semantic boundaries, guided not only by object identity but also by the distribution of light and the photographer's intent to direct visual attention. To realize their visual intent, retouchers employ a diverse toolkit, including radial gradients, linear gradients, brushes, and even compound masks, each with a graduated falloff that blends naturally into the surrounding scene. Figure 1(a) shows a

photographer's practical retouching intent and workflow, revealing a sequential and flexible use of diverse tools to process different areas. This includes brightening the subject area and applying a radial gradient to reinforce the light path, thereby directing visual attention. Unfortunately, none of the existing methods [24, 25, 38, 12] can capture the hierarchical and spatially nuanced nature of a professional retouching workflow.

We therefore propose **AURA**, an **AU**tonomous **R**etouching **A**gent mirroring the workflow of master photographers. Given only a photograph, AURA first analyzes the scene to form an overall retouching plan, then corrects global tonal deficiencies to establish a clean baseline, and finally refines local regions following the plan, deciding at each step which region to target and how to adjust it. This region-aware visual chain-of-thought enables fine-grained control that replicates professional-level deliberate editing. To achieve this goal, there are three core challenges to be addressed. **(C1) Long-horizon iterative visual reasoning**. Mimicking professional retouching requires the model to maintain coherent artistic intent across steps, assess the evolving visual outcome, and autonomously judge when the editing is complete — far beyond what single-pass or instruction-driven methods can support. **(C2) Diverse and non-semantic spatial localization**. Unlike object-level masks used by prior work, professional local adjustments are often defined by light paths, compositional geometry, and tonal transitions, implemented using spatially flexible selections, such as radial and linear gradients with continuous falloff. Enabling a model to autonomously reason about such regions remains an open problem. **(C3) Scarcity of expert-quality training data**. Existing retouching datasets typically lack expert trajectories. Annotating retouching trajectories of professional quality, accompanied by explicit artistic intent, is nontrivial.

We address these challenges through a cohesive design. For **C1**, we present a photographic visual thinking mechanism, where the rendered image after each regional edit is re-ingested as visual context, grounding subsequent reasoning in direct visual evidence. AURA is first trained via supervised fine-tuning on expert trajectories to acquire the basic retouching behavior, then refined through an agentic reinforcement learning stage that optimizes for perceptual quality and tool-use accuracy. For **C2**, we design a structured Lightroom-compatible masking tool suite that encapsulates diverse region-specification primitives, enabling AURA to specify complex, composable regions through compact parameterized tool calls rather than raw mask outputs. For **C3**, we contribute the first expert-annotated long-horizon retouching trajectory dataset grounded in landscape photography scenarios, where professional retouchers perform sequential edits from scratch with each operation driven by explicit artistic intent. As a result, AURA learns to follow the professional retouching workflow (Figure 1(b)), producing results that make the subject stand out through a clear tonal hierarchy while preserving natural color, rather than shifting the color style (Figure 1(c)).

We summarize our contributions as follows. First, we present AURA, an autonomous retouching agent that learns professional design logic and implements it through photographic visual thinking with flexible localization of both semantic and non-semantic regions. Second, we establish the first expert-annotated long-horizon retouching trajectory dataset grounded in landscape photography, where each editing step is paired with the photographer's explicit artistic rationale. Third, experiments demonstrate that AURA consistently outperforms existing methods in quantitative metrics and user studies, enabling professional one-click retouching without any manual intervention.

2 Related Work

Deep Image Retouching. Taking advantage of paired datasets such as MIT-Adobe5K [5] and PPR10K [23], deep learning-based image retouching has achieved remarkable progress. Early methods mostly estimate intermediate parameters of assumed physical models, including sequential global color modulation [14], curve-based color mapping [32, 22], affine bilateral grids [13, 27], and 3D look-up tables [44, 40, 41, 10]. Recent methods such as DiffRetouch [11] leverage diffusion models to jointly regress the target image and the affine bilateral grid, capturing the distribution of expert-retouched styles. To account for regional content differences, subsequent works incorporate spatial weight maps [36] or bilateral grids [18, 19], or directly learn local parametric filters [29]. In parallel, considering that expert retouching is a sequential decision-making process, reinforcement learning-based methods [16, 21] model each global editing step as an action within a Markov Decision Process. RetouchLLM [43] and MonetGPT [12] demonstrate the potential of multimodal large language models (MLLMs) for autonomous retouching by generating executable code or orchestrating tool calls across sequential stages, yet both remain confined to global adjustments, leaving a gap relative to professional-grade editing quality. Pertouch [8], JarvisArt [24], JarvisEvo [25], and

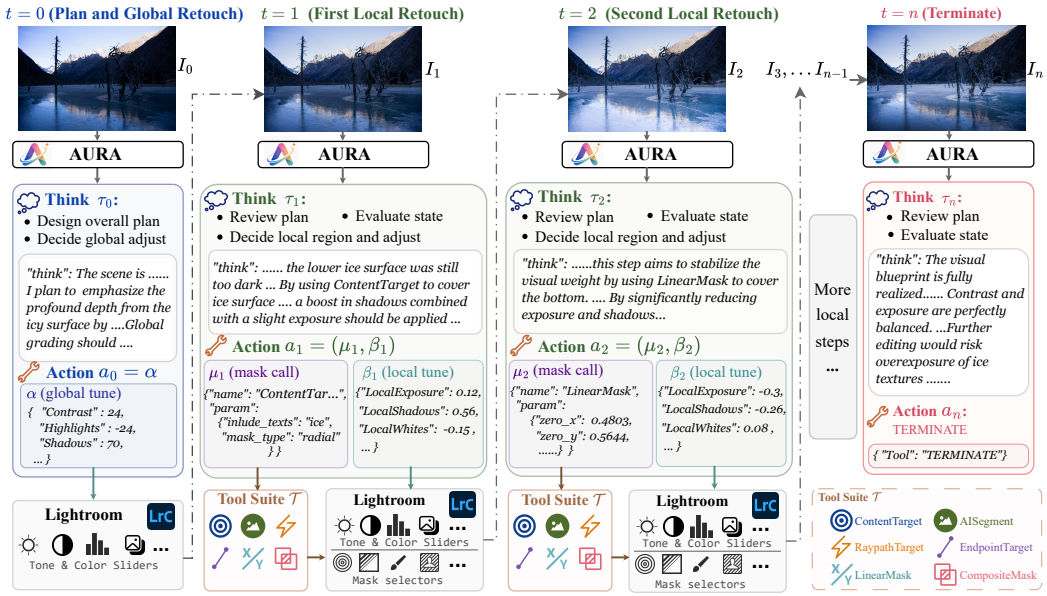

Figure 2: The overview of AURA, which mirrors the sequential process of expert retouching. At each step, AURA produces a reasoning text τ_t and a tool call action a_t. The first step establishes a global tonal baseline, and subsequent steps apply targeted regional edits until `TERMINATE` is emitted.

RetouchIQ [38] formulate retouching as an instruction-based editing problem. However, they are all reactive to user instructions rather than capable of making autonomous judgments, and their regional controllability is largely constrained by the semantic mask tools built into the underlying software. EyeControl [30] targets visual focus enhancement but still requires user clicks or strokes to specify the focus. To date, no method supports step-by-step visual reasoning over local operations, the reasoning capability that professional retouching demands, which is the goal of this work.

Multimodal Chain-of-Thought Reasoning. Advances in MLLMs have demonstrated a paradigm shift toward integrated multimodal reasoning, broadly referred to as "thinking with images" [28]. Early works often crop task-relevant regions as input for reasoning. Among these methods, DeepEyes [48], Ground-R1 [6], and Active-O3 [50] employ reinforcement learning (e.g., GRPO [31]) with tailored reward functions, whereas Chain-of-Focus [46], PixelReasoner [35], and VLM-R3 [17] employ cold-start SFT to equip models with structured reasoning formats, alleviating the burden on subsequent reinforcement learning. Beyond cropping, a richer class of methods endows models with active image manipulation. OpenThinkIMG [33] builds a tool-augmented agentic MLLM, whose toolset spans basic operations (e.g., crop, zoom-in, draw) and powerful external models such as SAM [20] and GroundingDINO [26]. Thyme [47] generates executable code to produce intermediate images within the reasoning loop. ReVPT [49] integrates a visual toolkit, including depth estimation and edge detection, to provide more visual information. DeepEyes v2 [15] combines code execution and search together to acquire more visual information.

Our work follows the paradigm of multimodal chain-of-thought reasoning. Rather than applying generic visual tools, we build retouching-specific toolkits upon SAM 3 [7], depth models [4], and luminance-based tonal zoning, enabling flexible selection of semantic and non-semantic regions tailored to professional retouching workflows. At each step, the model invokes Lightroom to execute the retouching operation and observes the updated image to decide the next action, instantiating a genuine step-by-step visual reasoning chain analogous to professional retouchers.

3 AURA: An Autonomous Retouching Agent

3.1 Framework Overview

AURA mirrors the workflow of an expert photographer to transform a raw photograph into a professionally retouched output through a sequential process. This iterative reasoning process forms a

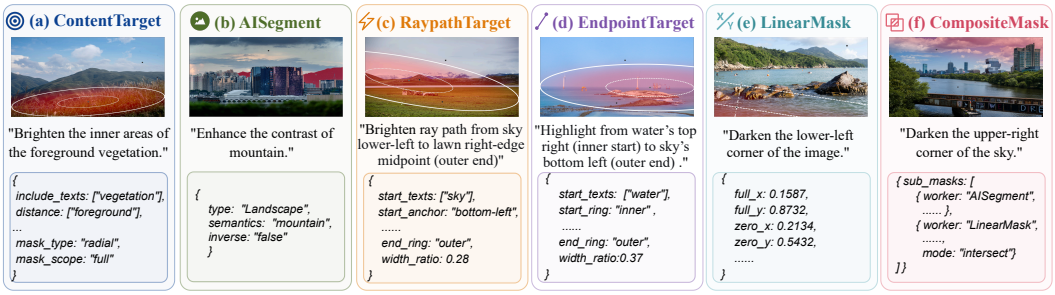

Figure 3: Overview of the masking tool suite $\mathcal{T}$. Each panel shows a representative edit: the source image with the resolved mask overlay (top), the reasoning intent (middle), and the emitted structured tool call (bottom). The mask overlay encodes effect strength through opacity, with darker regions indicating stronger application. Full parameter schemas are provided in the **Appendix**.

visual chain of thought that co-evolves with the image, as illustrated in Figure 2. We denote the raw input as I_0, our AURA model as Φ, the Lightroom execution engine as $\mathcal{E}$, and the editing history up to step t as $\mathcal{H}_t$. The model output at each step consists of a reasoning text τ_t followed by a tool call action $\mathbf{a}_t$. The specific content of τ_t and the form of $\mathbf{a}_t$ vary across steps and are detailed below.

At $t = 0$, Φ receives I_0 and produces a reasoning text τ_0 and a global tool call action $\mathbf{a}_0$. The reasoning text τ_0 begins with a holistic reading of the scene, assessing its composition, light, depth, and tonal character to form an artistic intent. This intent specifies the desired mood, the regions carrying the most visual weight, and how each part of the scene should be treated. Grounded in this intent, AURA then diagnoses the tonal deficiencies of I_0 and determines a global adjustment that brings the image to a clean baseline aligned with the plan. Based on this reasoning, the tool call $\mathbf{a}_0$ encodes a structured set of global adjustment parameters $\boldsymbol{\alpha}$, covering all color and tonal sliders such as exposure, contrast, saturation, highlights, shadows, HSL, and others. The Lightroom execution engine $\mathcal{E}$ responds to $\mathbf{a}_0$ and produces the globally graded image $I_1 = \mathcal{E}(I_0, \mathbf{a}_0)$.

For each subsequent step $t \geq 1$, Φ receives the current visual state I_t and the history $\mathcal{H}_t$, producing a reasoning text τ_t and a regional action $\mathbf{a}_t$. The reasoning text τ_t evaluates both the plan's fulfillment in I_t and any residual visual anomalies, then identifies the next region for adjustment and prescribes the necessary local corrections. The action $\mathbf{a}_t$ contains two components: a spatial selection tool call $\boldsymbol{\mu}_t$ and local tuning parameters $\boldsymbol{\beta}_t$, i.e., $\mathbf{a}_t = (\boldsymbol{\mu}_t, \boldsymbol{\beta}_t)$. $\boldsymbol{\mu}_t$ invokes our tool suite $\mathcal{T}$, which provides six tools for region specification across semantic and non-semantic areas (detailed in Section 3.2). $\mathcal{T}$ parses $\boldsymbol{\mu}_t$ and returns Lightroom-compatible mask parameters. $\boldsymbol{\beta}_t$ encodes the corresponding local tone and color adjustments. If the model judges that the image is complete at step n, it outputs `TERMINATE` and returns I_n as the final result. Otherwise, the image is updated as

$$I_{t+1} = \mathcal{E}(I_t,\, \mathbf{a}_t) \tag{1}$$

and fed back as visual context for the next step. Let $\mathbf{a}_0 = \boldsymbol{\alpha}$. The full trajectory is

$$I_0 \xrightarrow{\tau_0,\, \mathbf{a}_0} I_1 \xrightarrow{\tau_1,\, \mathbf{a}_1} I_2 \xrightarrow{\tau_2,\, \mathbf{a}_2} \cdots \xrightarrow{\texttt{TERMINATE}} I_n. \tag{2}$$

3.2 Tool Development

A fundamental challenge in local editing is the representation of spatial selections. A naive approach would have the model directly predict the raw mask parameters accepted by the Lightroom SDK, but this is infeasible for two reasons. First, professional retouching targets regions that are not well aligned with semantic categories, which are jointly defined by lighting, depth, and spatial position. Pretrained VLMs have limited priors for such regions. Second, raw Lightroom formats are fundamentally misaligned with visual reasoning. A radial gradient is parameterized by a rotation angle and ellipse bounding-box coordinates, and a brush stroke is encoded as a dense sequence of trajectory coordinates. Neither bears any resemblance to how a photographer thinks about a region, and the resulting token counts make training infeasible.

We therefore design a masking tool suite $\mathcal{T}$ that addresses both challenges. Instead of forcing the model to reason in raw Lightroom formats, $\mathcal{T}$ enables the model to emit high-level, interpretable tool

calls and translates them into executable mask primitives. The suite comprises six tools organized into four selection types: **content-aware selection**, **anchor-based selection**, **geometric selection**, and **composition selection**. Figure 3 illustrates them with representative examples, showing the intended use case and part of the tool call format emitted by the model. Each tool can translate these parameters into Lightroom-compatible mask primitives, integrating visual models such as referring segmentation and depth estimation, together with luminance-based tonal zoning and geometric solvers as needed. Full parameter schema and implementation details are provided in the **Appendix** A.

Content-aware Selection. `ContentTarget` handles regions jointly defined by semantics, lighting, depth, and spatial position. The model specifies the target using natural-language queries (`include_texts`, `exclude_texts`) and optional contextual attributes (`lighting`, `distance`, `position`). The model also chooses which Lightroom mask tool to use via `mask_type` (radial gradient, linear gradient, or brush). These inputs are resolved by a sequential intersection pipeline: a semantic mask is first obtained from SAM 3 [7] using the include/exclude queries, then optionally intersected with a depth layer from Depth Pro [4], a lit/shadow partition obtained by luminance-based tonal zoning, and a positional crop; the resulting binary mask is finally fitted to the selected Lightroom primitive by a dedicated geometric solver. As shown in Figure 3(a), "*the inner areas of foreground vegetation*" is selected with a radial gradient. For regions that align with a clear semantic category such as sky, water, or architecture, `AISegment` wraps Lightroom's native AI segmentation behind a simple category label, as shown in Figure 3(b).

Anchor-based Selection. `RaypathTarget` and `EndpointTarget` offer an elegant solution for regions lacking content identity such as a shaft of light (Figure 3(c)) or a soft horizon transition (Figure 3(d)). The key insight is that such regions can at least be defined by two natural-language reference points. Both tools accept two anchor descriptions plus a `width_ratio` and resolve them to a Lightroom radial gradient ellipse, where the inner ring receives full adjustment strength and the effect decays toward the outer ring. `RaypathTarget` fixes the start anchor to the inner ring, suited for directional light paths (Figure 3(c)). `EndpointTarget` allows either anchor to map to either ring, covering more general cases such as a horizon zone weighted toward one side (Figure 3(d)).

Geometric Selection. Not all edits require content understanding. Broad tonal sweeps such as darkening the sky, warming the lower third of the frame, or adding an edge vignette are defined purely by spatial geometry. `LinearMask` addresses this directly by specifying a linear gradient through two coordinate pairs: `(full_x, full_y)`, where the effect reaches 100%, and `(zero_x, zero_y)`, where it falls to 0%, as illustrated in Figure 3(e).

Composition. Real-world edits frequently target regions that no single primitive can capture. `CompositeMask` addresses this by combining the outputs of any subset of the above tools through boolean operations (`intersect`, `subtract`, `union`), specified as an ordered list of sub-masks. As shown in Figure 3(f), intersecting a sky segmentation mask with a linear gradient confines the adjustment to the overexposed upper-right sky while leaving the architecture untouched.

3.3 Training Data Construction

Dataset Collection and Expert Annotation. We collect raw photographs from professional photographers and amateurs, covering landscape, street, and still-life scenes. Portrait photography is deliberately excluded due to the strong cultural specificity of skin-tone aesthetics and associated privacy concerns. Unlike JarvisArt [24], which synthesizes training pairs by applying preset combinations to raw images, or RetouchIQ [38], which collects before-after pairs from real-world retouchers without structured annotation, we commission professional retouchers to edit each image from scratch under a controlled annotation protocol, producing trajectories grounded in explicit artistic intent.

The annotation team consists of five senior visual-effects designers, each with over five years of photographic post-production experience, supervised by a senior visual director. All retouchers follow a shared standard operating procedure (SOP) that emphasizes natural color preservation, light–shadow organization, subject emphasis, and suppression of distracting regions. The SOP also groups images into five scene categories with distinct lighting and composition patterns, guiding scene-aware tonal and emphasis strategies. The director reviews all final edits under a unified acceptance standard and returns unsatisfactory ones for further adjustment.

Retouchers follow the coarse-to-fine workflow that AURA is designed to emulate: global grading is applied first, followed by a sequence of local mask-based adjustments. For each local operation,

retouchers record two fields alongside the mask: a concise *region description* identifying the targeted area (e.g., "upper-left sky", "foreground rocks"), and an *adjustment intent* stating the artistic motivation behind the edit (e.g., "reduce visual weight of the bright sky corner to draw attention downward"). Upon completion, the full editing session is exported as a Lightroom preset file, serving as the input to the following action extraction and reasoning synthesis pipeline.

Action Extraction and Intermediate Rendering. Each exported parameter file is parsed to extract the editing actions at every step. The global adjustment parameters are read directly as $\boldsymbol{\alpha}$. For each local operation, the Lightroom mask selector used by the retoucher (radial gradient, linear gradient, AI landscape, etc.) and the accompanying textual region description are jointly used to derive the corresponding tool call parameters for localization specification $\boldsymbol{\mu}_t$ by Gemini 3.0 Flash. The local adjustment parameters $\boldsymbol{\beta}_t$ are read directly from the exported file. To construct the image sequence, we invoke the Lightroom execution engine $\mathcal{E}$ incrementally: first applying only $\boldsymbol{\alpha}$ to I_0 to obtain I_1, then cumulatively adding each local operation to produce $I_2, \ldots, I_T$. This yields a complete sequence of rendered intermediate states aligned with the action sequence.

Reasoning Label Synthesis. With the image sequence and action annotations in place, we synthesize the reasoning traces $\{\tau_t\}$ using Gemini 3.0 Flash in four passes. The first pass generates a preliminary blueprint by reverse-engineering the retoucher's core design intent from the original image I_0, the globally adjusted image I_1, the final output I_T, and the ordered annotation list. This structured blueprint captures scene content, subject rationale, and overall retouch plan. The second pass generates the global reasoning τ_0 from the raw and globally adjusted images along with the global adjustment parameters. The third pass produces step-level reasoning τ_t ($t \geq 1$) from the local image pair, mask overlay visualization, tool call specification, and accumulated editing context. The fourth pass yields the termination reasoning τ_T^{stop} from the original image, the final output, and the complete editing history. Combining all above components gives the complete training trajectory:

$$\left(I_0,\ \tau_0,\ \boldsymbol{\alpha},\ I_1,\ \tau_1,\ \mathbf{a}_1,\ I_2,\ \ldots,\ I_T,\ \tau_T^{\text{stop}}\right). \tag{3}$$

Details of the prompt templates and synthesis procedure, as well as a complete five-step sample trajectory, are provided in the **Appendix** B.

3.4 Training

Training AURA proceeds in two stages. The first stage uses supervised fine-tuning to equip the model with the basic structure of a retouching trajectory. The second stage uses reinforcement learning (RL) to push beyond imitation, rewarding the model for perceptual quality and reliable tool use.

Stage 1: Supervised Fine-Tuning. We initialize Φ by fine-tuning on expert trajectories from our dataset $\mathcal{D}$. Each trajectory is a complete sequence $(I_0,\ \tau_0,\ \mathbf{a}_0,\ I_1,\ \tau_1,\ \mathbf{a}_1,\ \ldots,\ I_T)$ annotated by professional retouchers with explicit artistic intent at each step. The model is trained with a standard next-token prediction objective over both the reasoning text τ_t and the structured action outputs $\mathbf{a}_t$, conditioned on the rendered image I_t and history $\mathcal{H}_t$:

$$\mathcal{L}_{\text{SFT}} = -\sum_{(I_0,\ldots,I_T)\,\in\,\mathcal{D}} \sum_{t=0}^{T} \log P_\Phi(\tau_t, \mathbf{a}_t \mid I_t, \mathcal{H}_t). \tag{4}$$

This stage gives the model a grounded understanding of the coarse-to-fine editing structure and a functional ability to invoke masking tools with valid parameters.

Stage 2: GRPO-based Reinforcement Learning. We further refine Φ using GRPO [31], an online RL algorithm that estimates advantages by comparing rollouts within a sampled group. For each input image I_0, the model generates G independent trajectory rollouts. The advantage of each rollout is computed relative to the group mean reward, and the policy is updated to increase the likelihood of higher-reward trajectories under a KL-regularized objective.

The reward assigned to each trajectory has three components. A *global reward* r_{global} measures how close I_1 (i.e., the image after global grading) is to the expert reference, reflecting the principle that a well-graded tonal foundation is the prerequisite for effective local editing. A *final reward* r_{final} measures the same for I_T (i.e., the image at termination), reflecting the overall retouching outcome. A *penalty* p discourages ill-formatted outputs and unterminated trajectories. The total reward is:

$$r = \lambda\, r_{\text{global}}(I_1) + (1-\lambda)\, r_{\text{final}}(I_T) - p, \tag{5}$$

where $\lambda \in (0, 1)$ balances the two fidelity terms; trajectories with non-executable actions (e.g., out-of-range parameters) receive $r = -1$.

4 Experiments

4.1 Experiment Settings

Datasets. We introduce AURA-5K, a dataset of 5,200 high-resolution images paired with professional retouching sequences. To ensure broad coverage of real-world photographic content, source images are licensed from both professional photographers and photography enthusiasts, encompassing landscape, cityscape, street photography, and still life. Landscapes and cityscapes predominate in the collection, as such images typically feature spatially distinct regions under varied lighting conditions that call for longer and more elaborate sequences of local adjustments, making them especially well-suited for evaluating multi-step retouching. All retouching sequences are annotated by professional retouchers, who begin with a global color grading step and then apply a series of local region adjustments via layer masks, yielding trajectories that range from 2 to 9 steps, with most centered around 3 and 4. We hold out 200 images for testing and use the remaining 5,000 for training. Detailed dataset statistics and data samples are provided in **Appendix** D.

Implementation Details. Our system builds upon Qwen2.5-VL-7B-Instruct [39] and is trained in two stages. In the first stage, we perform supervised fine-tuning on 5000 Visual Chain-of-Thought (Visual CoT) annotated samples, where each example includes a structured reasoning trace that explicitly decomposes the retouching decisions into a sequence of interpretable planning steps before generating adjustment parameters. We train for 3 epochs with a learning rate of 1e-5 and a batch size of 8 on 8 A100 GPUs. In the second stage, we apply GRPO training on 800 samples for 300 steps on 16 A100 (80GB) GPUs, with batch size 8, learning rate 1e-6, and 8 generated responses per query. All editing operations are executed through Adobe Lightroom [2].

Baselines. We compare AURA against two categories of methods. The first category consists of end-to-end retouching models trained on paired image data. We include 3D-LUT [44], a representative method that learns a global color transformation, and RSFNet [29], which goes beyond global edits by generating pixel-level attention maps through region-specific filters to support spatially localized retouching. Both methods are retrained from scratch on our training split using their official codebases to ensure a fair comparison. The second category consists of MLLM-based agents that orchestrate professional editing software, yielding lossless and detail-preserving results, including MonetGPT [12], JarvisArt [24], and JarvisEvo [25]. While AURA operates fully autonomously without user input, JarvisArt and JarvisEvo require explicit instructions to guide the editing process. To reflect a realistic scenario where ordinary users are unlikely to articulate specific retouching goals, we provide both with the uniform prompt *"make this photo beautiful"* across all evaluations.

Evaluation Metrics. We evaluate all methods using five complementary metrics. **PSNR** measures pixel-level fidelity between the retouched output and the professional reference. **SSIM** [37] captures structural similarity, and **LPIPS** [45] measures perceptual distance using deep features; together they provide a more holistic view of perceived quality than PSNR alone. **Histogram Intersection** [16] is computed independently over the contrast, luminance, and color saturation channels and then averaged, measuring how closely the tonal and color distribution of the output matches that of the reference. ΔE^* is the color difference in the perceptually uniform CIELAB color space [3], which provides a direct measure of color accuracy where lower values indicate better performance.

4.2 Comparisons

Quantitative Comparisons. Quantitative results on AURA-5K demonstrate that AURA achieves the best overall performance among MLLM agent-based retouching methods, and leads in perceptual and color-consistency metrics across all comparisons. MonetGPT is limited to global color adjustment and struggles with complex local editing demands. JarvisArt and JarvisEvo rely on vague prompts such as *"make this photo beautiful"*, lacking strong content analysis and adaptive decision-making. Although JarvisEvo occasionally supports two-round editing, its operations remain predominantly global, collapsing local intents into a single global transformation. Among CNN-based methods, RSFNet incorporates explicit region attention for more direct local modeling; combined with learnable common patterns in the dataset (e.g., darkening border regions), it achieves the best results on pixel-

Table 1: Quantitative comparison on AURA-5K. Best in **bold**, second-best underlined.

Method	PSNR↑	SSIM↑	LPIPS↓	Hist.↑	$\Delta E^* \downarrow$
3D-LUT [44]	21.01	0.8692	0.1043	0.7188	20.17
RSFNet [29]	**22.45**	**0.9019**	0.0752	0.8192	16.82
MonetGPT [12]	19.89	0.8492	0.1183	0.7213	21.87
JarvisArt [24]	18.29	0.8238	0.1784	0.7074	28.06
JarvisEvo [25]	20.12	0.8695	0.1178	0.7434	22.17
AURA (Ours)	22.38	0.8903	**0.0751**	**0.8203**	**16.63**

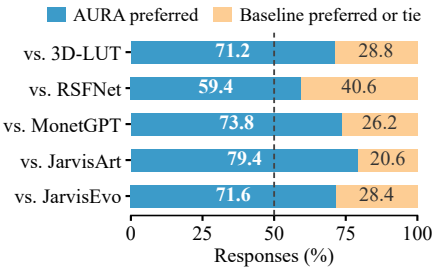

Figure 4: User study (100 images, 30 participants). Only explicit AURA choices are counted as wins.

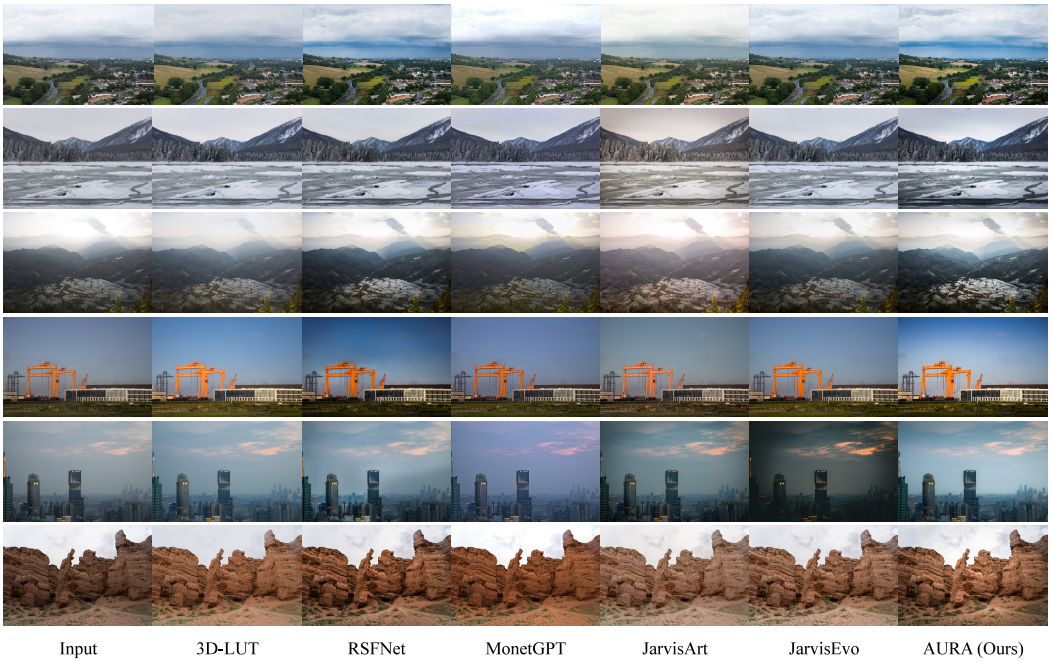

Figure 5: Qualitative comparisons. Compared to other methods, our results achieve faithful color reproduction with enhanced luminosity, highlighting the most striking elements of the scene.

and structure-fidelity metrics (PSNR and SSIM), with AURA a close second on both. Overall, AURA exhibits a more balanced and robust performance, leading in perceptual quality (LPIPS), tonal distribution (Hist.), and color accuracy (ΔE^*).

User Study. Fixed-reference metrics cannot fully capture the subjective nature of retouching, so we complement them with a pairwise perceptual study. For each of the 100 test images, AURA is paired with each of the five baselines, and all 30 participants rate every pair (15,000 judgments in total). Each trial shows the original photograph and the two outputs with randomized order and left–right placement, and the study is split into ten 50-trial blocks with mandatory breaks. Participants can choose AURA, the baseline, or *no preference*. Figure 4 counts only explicit AURA choices as wins, which gives a conservative estimate. AURA is preferred in 59.4%–79.4% of responses, exceeding 50% against each baseline. Most baselines mainly produce global color changes, whereas RSFNet, retrained on AURA-5K, learns some salient local effects and is thus the most competitive competitor, with more ties with AURA and a narrower margin.

Stronger Prompts for Baselines. We further test whether stronger prompts improve JarvisArt and JarvisEvo. V1 uses one detailed prompt for all images that describes AURA's objective: *"Professionally retouch this photo while preserving natural color, emphasizing the main subject area, improving light-shadow structure, and avoiding over-saturated or filter-like color shifts."* V2 uses an image-specific retouching plan generated by a frozen Qwen2.5-VL-7B. Under V1, both baselines still apply few masks with weak effects. JarvisArt uses a single mask, mostly a built-in semantic one,

in 124 of 200 images, and JarvisEvo uses one or two masks in 101 images. In contrast, AURA-5K trajectories typically use 3–6 intent-driven masks. Under V2, JarvisArt repeats the same parameters without stopping in 35 images, and JarvisEvo invokes masks in only 25. Stronger prompts alone cannot substitute for scene-aware regional design.

Qualitative Comparisons. Figure 5 presents side-by-side comparisons on representative test images. MonetGPT operates only globally, producing flat, uniformly toned results that lack the tonal depth of professional edits. JarvisArt and JarvisEvo also default to global adjustments under the generic prompt. They do not decompose the scene into regions and thus miss opportunities to selectively brighten a foreground subject, deepen a background sky, or add local contrast that draws the eye. RSFNet produces more spatially differentiated outputs, as its region-specific filters learn to suppress luminance in peripheral areas. However, its hard region boundaries often cause visible artifacts and abrupt transitions between adjusted and unadjusted areas, as shown in the third and fourth rows of Figure 5. In contrast, AURA uses Lightroom's gradient masks to apply adjustments with smooth, photographer-style feathering. By orchestrating multiple such adjustments in concert with the scene's existing light and compositional structure, AURA produces outputs that feel deliberately crafted rather than algorithmically processed. Our results preserve color fidelity close to the original scene while adding a greater sense of luminosity that draws attention to the most compelling parts of the photograph. More visual comparisons are provided in **Appendix** E.

4.3 Ablation Study

We ablate two key design choices of AURA in Table 2. **(i) Text-CoT only** replaces Visual CoT with a text-only chain-of-thought that generates all editing parameters in a single forward pass, covering design intent and global color analysis (*e.g.*, exposure, white balance, tonal distribution). This variant suffers the most pronounced drop, exposing the core limitation of open-loop prediction: without rendered intermediate images, the model cannot perceive how earlier edits alter subsequent conditions, causing errors to compound across steps. High-level textual reasoning cannot substitute for the direct visual evidence that partially-edited images provide.

Table 2: Ablation study on AURA-5K.

Variant	PSNR↑	SSIM↑	LPIPS↓	Hist.↑	ΔE^* ↓
Text-CoT only	21.47	0.8814	0.0824	0.7950	19.02
SFT only	22.16	0.8827	0.0774	0.8170	16.98
Full Model (AURA)	**22.38**	**0.8903**	**0.0751**	**0.8203**	**16.63**

(ii) SFT only trains exclusively via supervised fine-tuning on Visual CoT demonstrations. It performs close to the full model but falls slightly behind. SFT serves as the foundation, providing the agent with a strong initialization for multi-turn retouching. Building on this, the RL stage samples rollouts and uses reward feedback to suppress the probability of low-quality editing trajectories, further improving overall performance.

5 Conclusion

We presented AURA, an autonomous retouching agent that emulates the deliberate, hierarchical workflow of professional photographers—perceiving scene semantics, reasoning about aesthetic intent, and executing spatially precise, light-aware edits across diverse compositional regions. By closing the loop between visual observation and editing decision, AURA moves beyond single-pass parameter prediction toward genuine visual reasoning grounded in intermediate rendering feedback. Beyond professional use, AURA also democratizes high-quality retouching, enabling casual photographers to achieve expert-level results with minimal effort. To support this direction, we established **AURA-5K**, the **first** retouching dataset featuring expert-annotated long-horizon trajectories with richly structured local operations, providing a rigorous benchmark for evaluating step-wise editing behavior. We hope that AURA will inspire future collaborative editing agents that bridge automated tools and human artistic expertise.

Limitations. Despite its promising results, AURA depends on external vision models for region grounding, creating an error cascade where inaccuracies propagate into local editing and misalign edits with the intended region. This issue is worsened by complex lighting, fine details, or vague language. A key future direction is to internalize region grounding within the agent, enabling end-to-end training of semantic understanding and editing policy under a unified objective.

References

[1] Adobe Inc. Adobe Photoshop.

[2] Adobe Inc. *Adobe Lightroom*. Adobe Inc., 2026. Version 5.x or later (or current version).

[3] W. Backhaus, R. Kliegl, and J.S. Werner. *Color Vision: Perspectives from Different Disciplines*. Walter de Gruyter, 1998.

[4] Aleksei Bochkovskii, AmaãGl Delaunoy, Hugo Germain, Marcel Santos, Yichao Zhou, Stephan R Richter, and Vladlen Koltun. Depth Pro: Sharp monocular metric depth in less than a second. *arXiv preprint arXiv:2410.02073*, 2024.

[5] Vladimir Bychkovsky, Sylvain Paris, Eric Chan, and Frédo Durand. Learning photographic global tonal adjustment with a database of input / output image pairs. In *The Twenty-Fourth IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[6] Meng Cao, Haoze Zhao, Can Zhang, Xiaojun Chang, Ian Reid, and Xiaodan Liang. Ground-R1: Incentivizing grounded visual reasoning via reinforcement learning. *arXiv preprint arXiv:2505.20272*, 2025.

[7] Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, et al. SAM 3: Segment anything with concepts. *arXiv preprint arXiv:2511.16719*, 2025.

[8] Zewei Chang, Zheng-Peng Duan, Jianxing Zhang, Chun-Le Guo, Siyu Liu, Hyungju Chun, Hyunhee Park, Zikun Liu, and Chongyi Li. PerTouch: VLM-driven agent for personalized and semantic image retouching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 2752–2759, 2026.

[9] Haoyu Chen, Keda Tao, Yizao Wang, Xinlei Wang, Lei Zhu, and Jinjin Gu. PhotoArtAgent: Intelligent photo retouching with language model-based artist agents. *ArXiv*, abs/2505.23130, 2025.

[10] Marcos V Conde, Javier Vazquez-Corral, Michael S Brown, and Radu Timofte. NILUT: Conditional neural implicit 3D lookup tables for image enhancement. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 1371–1379, 2024.

[11] Zheng-Peng Duan, Jiawei Zhang, Zheng Lin, Xin Jin, XunDong Wang, Dongqing Zou, Chun-Le Guo, and Chongyi Li. DiffRetouch: Using diffusion to retouch on the shoulder of experts. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025.

[12] Niladri Shekhar Dutt, Duygu Ceylan, and Niloy J Mitra. MonetGPT: Solving puzzles enhances MLLMs' image retouching skills. *ACM Transactions on Graphics (TOG)*, 44(4):1–12, 2025.

[13] Michaël Gharbi, Jiawen Chen, Jonathan T Barron, Samuel W Hasinoff, and Frédo Durand. Deep bilateral learning for real-time image enhancement. *ACM Transactions on Graphics (TOG)*, 36(4):1–12, 2017.

[14] Jingwen He, Yihao Liu, Yu Qiao, and Chao Dong. Conditional sequential modulation for efficient global image retouching. In *European Conference on Computer Vision*, pages 679–695. Springer, 2020.

[15] Jack Hong, Chenxiao Zhao, ChengLin Zhu, Weiheng Lu, Guohai Xu, and Xing Yu. DeepEyesV2: Toward agentic multimodal model. *arXiv preprint arXiv:2511.05271*, 2025.

[16] Yuanming Hu, Hao He, Chenxi Xu, Baoyuan Wang, and Stephen Lin. Exposure: A white-box photo post-processing framework. *ACM Transactions on Graphics (TOG)*, 37(2):1–17, 2018.

[17] Chaoya Jiang, Yongrui Heng, Wei Ye, Han Yang, Haiyang Xu, Ming Yan, Ji Zhang, Fei Huang, and Shikun Zhang. VLM-R^3: Region recognition, reasoning, and refinement for enhanced multimodal chain-of-thought. *arXiv preprint arXiv:2505.16192*, 2025.

[18] Wontae Kim and Nam Ik Cho. Image-adaptive 3D lookup tables for real-time image enhancement with bilateral grids. In *European Conference on Computer Vision*, pages 91–108. Springer, 2024.

[19] Wontae Kim, Keuntek Lee, and Nam Ik Cho. Lightweight and fast real-time image enhancement via decomposition of the spatial-aware lookup tables. In *International Conference on Computer Vision*, 2025.

[20] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4015–4026, 2023.

[21] Satoshi Kosugi and Toshihiko Yamasaki. Unpaired image enhancement featuring reinforcement-learning-controlled image editing software. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 11296–11303, 2020.

[22] Chongyi Li, Chunle Guo, Shangchen Zhou, Qiming Ai, Ruicheng Feng, and Chen Change Loy. FlexiCurve: Flexible piecewise curves estimation for photo retouching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1092–1101, 2023.

[23] Jie Liang, Hui Zeng, Miaomiao Cui, Xuansong Xie, and Lei Zhang. PPR10K: A large-scale portrait photo retouching dataset with human-region mask and group-level consistency. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 653–661, 2021.

[24] Yunlong Lin, Zixu Lin, Kunjie Lin, Jinbin Bai, Panwang Pan, Chenxin Li, Haoyu Chen, Zhongdao Wang, Xinghao Ding, Wenbo Li, et al. JarvisArt: Liberating human artistic creativity via an intelligent photo retouching agent. *arXiv preprint arXiv:2506.17612*, 2025.

[25] Yunlong Lin, Linqing Wang, Kunjie Lin, Zixu Lin, Kaixiong Gong, Wenbo Li, Bin Lin, Zhenxi Li, Shiyi Zhang, Yuyang Peng, et al. JarvisEvo: Towards a self-evolving photo editing agent with synergistic editor-evaluator optimization. *arXiv preprint arXiv:2511.23002*, 2025.

[26] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, et al. Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection. In *European conference on computer vision*, pages 38–55. Springer, 2024.

[27] Junyu Lou, Xiaorui Zhao, Kexuan Shi, and Shuhang Gu. Learning pixel-adaptive multi-layer perceptrons for real-time image enhancement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14095–14105, October 2025.

[28] OpenAI. Think with image, 2025.

[29] Wenqi Ouyang, Yi Dong, Xiaoyang Kang, Peiran Ren, Xin Xu, and Xuansong Xie. RSFNet: A white-box image retouching approach using region-specific color filters. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12160–12169, 2023.

[30] Chujie Qin, Zilong Zhang, Zewei Chang, Chunle Guo, Ruixing Wang, Tao Hu, Ming-Ming Cheng, and Chongyi Li. Dotting the eye: An intent-driven image retouching agent for visual focus enhancement. In *European Conference on Computer Vision (ECCV)*, pages 574–591, 2026.

[31] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Jun-Mei Song, Mingchuan Zhang, Y. K. Li, Yu Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[32] Yuda Song, Hui Qian, and Xin Du. StarEnhancer: Learning real-time and style-aware image enhancement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4126–4135, 2021.

[33] Zhaochen Su, Linjie Li, Mingyang Song, Yunzhuo Hao, Zhengyuan Yang, Jun Zhang, Guanjie Chen, Jiawei Gu, Juntao Li, Xiaoye Qu, et al. OpenThinkIMG: Learning to think with images via visual tool reinforcement learning. *arXiv preprint arXiv:2505.08617*, 2025.

[34] The GIMP Development Team. GNU image manipulation program (GIMP), 2026.

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

[36] Tao Wang, Yong Li, Jingyang Peng, Yipeng Ma, Xian Wang, Fenglong Song, and Youliang Yan. Real-time image enhancer via learnable spatial-aware 3D lookup tables. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2471–2480, 2021.

[37] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[38] Qiucheng Wu, Jing Shi, Simon Jenni, Kushal Kafle, Tianyu Wang, Shiyu Chang, and Handong Zhao. RetouchIQ: MLLM agents for instruction-based image retouching with generalist reward. *arXiv preprint arXiv:2602.17558*, 2026.

[39] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[40] Canqian Yang, Meiguang Jin, Xu Jia, Yi Xu, and Ying Chen. AdaInt: Learning adaptive intervals for 3D lookup tables on real-time image enhancement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17522–17531, 2022.

[41] Canqian Yang, Meiguang Jin, Yi Xu, Rui Zhang, Ying Chen, and Huaida Liu. SepLUT: Separable image-adaptive lookup tables for real-time image enhancement. In *European Conference on Computer Vision*, pages 201–217. Springer, 2022.

[42] Mingde Yao, Zhiyuan You, King-Man Tam, Menglu Wang, and Tianfan Xue. PhotoAgent: Exploratory visual aesthetic planning with large vision models. In *International Conference on Machine Learning (ICML)*, 2026.

[43] Moon Ye-Bin, Roy Miles, Tae-Hyun Oh, Ismail Elezi, and Jiankang Deng. RetouchLLM: Training-free code-based image retouching with vision language models. *arXiv preprint arXiv:2510.08054*, 2025.

[44] Hui Zeng, Jianrui Cai, Lida Li, Zisheng Cao, and Lei Zhang. Learning image-adaptive 3D lookup tables for high performance photo enhancement in real-time. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(4):2058–2073, 2020.

[45] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[46] Xintong Zhang, Zhi Gao, Bofei Zhang, Pengxiang Li, Xiaowen Zhang, Yang Liu, Tao Yuan, Yuwei Wu, Yunde Jia, Song-Chun Zhu, et al. Chain-of-Focus: Adaptive visual search and zooming for multimodal reasoning via RL. *arXiv e-prints*, pages arXiv–2505, 2025.

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

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

[49] Zetong Zhou, Dongping Chen, Zixian Ma, Zhihan Hu, Mingyang Fu, Sinan Wang, Yao Wan, Zhou Zhao, and Ranjay Krishna. Reinforced visual perception with tools. *arXiv preprint arXiv:2509.01656*, 2025.

[50] Muzhi Zhu, Hao Zhong, Canyu Zhao, Zongze Du, Zheng Huang, Mingyu Liu, Hao Chen, Cheng Zou, Jingdong Chen, Ming Yang, and Chunhua Shen. Active-O3: Empowering multimodal large language models with active perception via GRPO. *arXiv preprint arXiv:2505.21457*, 2025.

Appendices

This appendix includes the following sections:

- Sec. A Details of Tool Development.
  - ContentTarget.
  - AISegment.
  - RaypathTarget.
  - EndpointTarget.
  - LinearMask.
  - CompositeMask
- Sec. B Details of Data Construction.
  - Details of Action Extraction.
  - Details of Reasoning Label Synthesis.
  - Samples of Data Trajectory.
- Sec. C Details of Reward Calculation.
- Sec. D Details of AURA-5K Dataset.
  - Data Samples.
  - Retouching Turns Distribution.
- Sec. E Additional Experimental Results.
  - More Visual Comparisons.

A Additional Details of Tool Development

A.1 ContentTarget

`ContentTarget` handles regions jointly defined by semantics, lighting, depth, and spatial position, where the model specifies targets via natural-language queries (`include_texts`, `exclude_texts`) and optional contextual attributes (`lighting`, `distance`, `position`), and selects a Lightroom mask type (`mask_type`) accordingly.

Schema.

The schema overview and specific parameters are explained as follows.

Schema Definition

```
{
  "include_texts":  ["string", ...],
  "exclude_texts":  ["string", ...],  // optional
  "lighting":       "lit | dark",     // optional
  "distance":       "foreground | midground | background",
                    // optional; string or list
  "position":       "top | bottom | upper_half | lower_half | left
     | right
                     | image_right | image_left | image_center |
                       ridge
                     | image_upper_half | image_lower_half |
                       image_surrounding",
                    // optional; string or list
  "mask_type":      "radial | linear | brush",
  "radial_scope":   "core | full | main",
                    // optional; only when mask_type = radial
  "inverse":        "false"
                    // optional; bool, default false
}
```

`include_texts` and `exclude_texts` accept one or more natural-language phrases that jointly define the semantic content of the target region: the former specifies what should be included, and the latter selects everything outside the named concepts. Either field may be used alone.

`lighting` narrows the region to either the illuminated face (`lit`) or the shadowed face (`dark`) of the matched content, allowing the model to isolate, for example, the sunlit side of a rock formation independently of its shadow.

`distance` constrains the selection by depth layer (`foreground`, `midground`, or `background`), and multiple values may be combined.

`position` further restricts the spatial extent of the matched region. Values prefixed with `image_` (e.g., `image_left`, `image_upper_half`) refer to absolute quadrants of the full frame, whereas unprefixed values (e.g., `left`, `top`) denote relative sub-parts *within* the region already determined by the semantic and depth attributes above—enabling, for instance, the left portion of the midground vegetation.

`mask_type` specifies which Lightroom mask selector is used to cover the identified region: `radial` for a radial gradient ellipse, `linear` for a directional gradient, and `brush` for a freeform stroke.

`radial_scope` controls the sizing of the radial gradient relative to the identified mask when `mask_type` is `radial`. Even for an identical semantic mask, different scope values produce meaningfully different editing effects because they govern both the spatial extent of the adjustment and the opacity falloff profile. `main` sizes the inner ring to cover approximately 75% of the mask, appropriate when the intent is to affect the principal subject of the region while allowing a natural fade toward its boundary. `core` sizes the inner ring to match the mask exactly, placing the full-strength adjustment at the center of mass and letting the effect decay outward beyond the region—suitable for edits conceived as centered on the target rather than confined to it. `full` sizes the outer ring to lie just within the mask boundary, so the entire decay range of the gradient is absorbed inside the region—appropriate when the goal is to fill the interior of the target uniformly.

Finally, `inverse` flips the resolved mask, applying the adjustment to all areas outside the identified region.

The following three examples demonstrate how these parameters work together, mapping typical retoucher descriptions to their corresponding `ContentTarget` calls.

"The main body of sunlit foreground vegetation"

```
{
  "include_texts": [
      "vegetation"],
  "lighting": "lit",
  "distance": "foreground",
  "mask_type": "radial",
  "radial_scope": "main"
}
```

"The interior region excluding the sky and water surfaces"

```
{
  "exclude_texts": ["sky",
      "water"],
  "mask_type": "radial",
  "radial_scope": "full"
}
```

"The area outside the surrounding region centered on the background mountains"

```
{
  "include_texts": [
      "mountain"],
  "distance": "background",
  "mask_type": "radial",
  "radial_scope": "core",
  "inverse": "True"
}
```

Implementation.

`ContentTarget` resolves the target region through a sequential intersection pipeline. The semantic mask is first obtained by querying **SAM3** with each entry in `include_texts` and `exclude_texts`, taking their union and difference respectively. If `distance` is specified, a depth map is estimated by **DepthPro** and partitioned into foreground, midground, and background layers. If `lighting` is specified, a shading model separates lit and dark surfaces. If `position` is specified, a geometric crop is applied either over the full frame (`image_*` variants) or within the bounding box of the current mask. The resulting masks are intersected in order, and the final binary mask is fitted to Lightroom-compatible parameters via a radial gradient, linear gradient, or brush stroke solver according to `mask_type`.

A.2 AISegment

`AISegment` wraps Lightroom's native AI segmentation behind a simple category label, designed for regions that align with a clear semantic category such as sky, water, or architecture.

Schema Definition

```
// Mode 1: landscape semantics
{
  "segment_type": "landscape",
  "semantics":    ["Sky", "Water", ...],  // Lightroom AI
     landscape categories
  "inverse":      false                   // optional; bool,
     default false
}

// Mode 2: people
{
  "segment_type": "people",
  "inverse":      false                   // optional; bool,
     default false
}
```

`segment_type` selects the operating mode. In `landscape` mode, `semantics` accepts one or more category labels from Lightroom's built-in AI landscape segmentation: `Sky`, `Water`, `Architecture`, `Natural Ground`, `Mountain`, and `Artificial Ground`. The `people` mode invokes Lightroom's subject detection and requires no additional parameters. `inverse` optionally inverts the resolved mask.

Representative examples of `AISegment` calls are shown below.

"Sky only"

```
{
  "segment_type":
       "landscape",
  "semantics": ["Sky"]
}
```

"Sky and water together"

```
{
  "segment_type":
       "landscape",
  "semantics": ["Sky",
       "Water"]
}
```

"Portrait subject"

```
{
  "segment_type": "people"
}
```

Implementation. `AISegment` requires no model inference beyond Lightroom's built-in AI engine. In `landscape` mode, each entry in `semantics` is mapped to the corresponding Lightroom AI landscape category identifier and passed directly to the Lightroom mask API, which returns the segmentation result. In `people` mode, Lightroom's native subject detection is invoked without any additional parameters. The resolved mask is optionally inverted via `inverse` and returned as Lightroom-compatible `CorrectionMasks` parameters.

A.3 RaypathTarget

`RaypathTarget` resolves regions that lack content identity—such as a shaft of light—by accepting two natural-language anchor descriptions and a `width_ratio`. The start anchor is fixed to the inner ring of the resolved radial gradient ellipse, which makes the tool well suited for directional light paths.

Schema Definition

```
{
  // --- start anchor (always maps to the inner ring) ---
  "start_include_texts": ["string", ...],  // optional
  "start_exclude_texts": ["string", ...],  // optional
  "start_lighting":      "lit | dark",     // optional
  "start_distance":      "foreground | midground | background",
      // optional; list
  "start_position":      "...",            // optional; same
      values as ContentTarget
  "start_anchor":        "top-left | top-right | bottom-left |
      bottom-right
                          | center | left-edge | right-edge
                          | top-edge | bottom-edge",

  // --- end anchor ---
  "end_include_texts":   ["string", ...],  // optional
  "end_exclude_texts":   ["string", ...],  // optional
  "end_lighting":        "lit | dark",     // optional
  "end_distance":        "foreground | midground | background",
      // optional; list
  "end_position":        "...",            // optional; same
      values as ContentTarget
  "end_anchor":          "top-left | top-right | bottom-left |
      bottom-right
                          | center | left-edge | right-edge
                          | top-edge | bottom-edge",
  "end_ring":            "inner | outer",

  // --- gradient width ---
  "width_ratio":         0.3              // required; float > 0
}
```

The schema is split into `start_*` and `end_*` parameter groups, each defining one endpoint of the radial gradient's major axis. Within each group, `include_texts`, `exclude_texts`, `lighting`, `distance`, and `position` carry the same semantics as in `ContentTarget` and jointly identify a candidate region in the image. `anchor` then selects a specific point within that region: the four corner positions (`top-left`, `top-right`, `bottom-left`, `bottom-right`), the four edge midpoints (`top-edge`, `right-edge`, `bottom-edge`, `left-edge`), or `center`. In `RaypathTarget`, the start anchor is always fixed to the inner ring. `end_ring` controls which ring the end anchor maps to: `inner` places it at the intersection of the inner ring with the major axis, `outer` at the outer ring. `width_ratio` sets the width of the ellipse relative to its length, governing how broad the gradient band appears.

Representative examples of `RaypathTarget` calls are shown below.

"The light path starting from the midpoint of the top edge of the sky, and ending at the outer point at the bottom-right corner of the grass."

```
{
  "start_include_texts": [
      "sky"],
  "start_anchor": "top-edge",
  "end_include_texts": [
      "grass"],
  "end_anchor":
      "bottom-right",
  "end_ring": "outer",
  "width_ratio": 0.2
}
```

"Light path from top-left (lit area) to bottom-right (outer ring)."

```
{
  "start_lighting": "lit",
  "start_anchor": "top-left",
  "end_anchor":
      "bottom-right",
  "end_ring": "outer",
  "width_ratio": 0.15
}
```

"Light path from the midpoint of the sky's top edge to the center of the foreground rock as the outer endpoint."

```
{
  "start_include_texts": [
      "sky"],
  "start_anchor": "top-edge",
  "end_include_texts": [
      "rock"],
  "end_distance": [
      "foreground"],
  "end_anchor": "center",
  "end_ring": "outer",
  "width_ratio": 0.25
}
```

Implementation. Each anchor point is resolved through the same sequential pipeline as `ContentTarget`: `include_texts/exclude_texts` are queried via SAM3 to obtain a semantic mask, which is then intersected with depth-layer, lighting, and position constraints as applicable. From the resulting region mask, the `anchor` value selects a single representative point—a corner, edge midpoint, or centroid—of the region's bounding box. Given the two resolved points p_{start} and p_{end}, the start anchor is always fixed to the inner ring, so the ellipse center and semi-major axis are back-calculated from p_{end} and `end_ring`: if `end_ring` is `inner`, p_{end} lies on the inner ring along the major axis; if `outer`, it lies on the outer ring. `width_ratio` then sets the semi-minor axis as a fraction of the semi-major axis. The resulting ellipse geometry is finally converted to Lightroom's radial gradient parameterization (center coordinates, bounding box, rotation angle, feather).

A.4 EndpointTarget

`EndpointTarget` handles general anchor-based regions—such as a soft horizon transition—by accepting two natural-language anchor descriptions and allowing either anchor to map to either ring of the resolved radial gradient ellipse, covering cases weighted toward one side.

Schema Definition

```
{
  // --- start anchor ---
  "start_include_texts": ["string", ...],  // optional
  "start_exclude_texts": ["string", ...],  // optional
  "start_lighting":      "lit | dark",     // optional
  "start_distance":      "foreground | midground | background",
     // optional; list
  "start_position":      "...",            // optional; same
     values as ContentTarget
  "start_anchor":        "top-left | top-right | bottom-left |
     bottom-right
                          | center | left-edge | right-edge
                          | top-edge | bottom-edge",
  "start_ring":          "inner | outer",

  // --- end anchor ---
  "end_include_texts":   ["string", ...],  // optional
  "end_exclude_texts":   ["string", ...],  // optional
  "end_lighting":        "lit | dark",     // optional
  "end_distance":        "foreground | midground | background",
     // optional; list
  "end_position":        "...",            // optional; same
     values as ContentTarget
  "end_anchor":          "top-left | top-right | bottom-left |
     bottom-right
                          | center | left-edge | right-edge
                          | top-edge | bottom-edge",
  "end_ring":            "inner | outer",

  // --- gradient width ---
  "width_ratio":         0.3              // required; float > 0
}
```

`EndpointTarget` follows the same schema structure as `RaypathTarget`, with one key addition: `start_ring` is independently configurable, allowing the start anchor to map to either the inner or outer ring. This decouples the two endpoints and covers transition zones where neither end is constrained to a fixed ring, such as a horizon gradient weighted toward one side or a soft fog band between depth layers.

Implementation. `EndpointTarget` follows the same region-identification and anchor-resolution pipeline as `RaypathTarget`. The key difference is that both `start_ring` and `end_ring` are independently configurable, so neither endpoint is constrained to a fixed ring. Given the two resolved

anchor points p_{start} and p_{end} and their respective ring assignments, the ellipse center and semi-major axis are jointly solved: the distance along the major axis between the inner and outer ring intersections is determined by the Lightroom feather model, and the two ring constraints together uniquely determine the center position and axis length. `width_ratio` and the final Lightroom conversion proceed identically to `RaypathTarget`.

A.5 LinearMask

`LinearMask` addresses broad tonal sweeps defined purely by spatial geometry—such as darkening the sky, warming the lower third of the frame, or adding an edge vignette—by specifying a linear gradient through two coordinate pairs: `(full_x, full_y)`, where the effect reaches 100%, and `(zero_x, zero_y)`, where it falls to 0%.

Schema Definition

```
{
  "full_x":        0.2,   // x-coordinate where effect reaches 100
      %; normalized [0,1]
  "full_y":        0.6,   // y-coordinate where effect reaches 100%
  "zero_x":        0.1,   // x-coordinate where effect falls to 0%
  "zero_y":        0.8,   // y-coordinate where effect falls to 0%
  "mask_inverted": false  // optional; bool, default false
}
```

All four coordinates are expressed relative to the image dimensions—x values are divided by image width and y values by image height—so $(0, 0)$ corresponds to the top-left corner and $(1, 1)$ to the bottom-right, but values outside this range are valid and place the gradient anchor beyond the frame boundary. `full_x` and `full_y` specify the point where the effect reaches 100%; `zero_x` and `zero_y` specify where it falls to 0%. The gradient direction and slope are fully determined by these two points, allowing arbitrary-angle sweeps across the frame. `mask_inverted` optionally swaps the full and zero ends.

Representative examples of `LinearMask` calls are shown below.

"Darken sky: top full, mid zero"

```
{
  "full_x": 0.5,
  "full_y": 0.0,
  "zero_x": 0.5,
  "zero_y": 0.5
}
```

"Warm lower third of frame"

```
{
  "full_x": 0.5,
  "full_y": 1.0,
  "zero_x": 0.5,
  "zero_y": 0.67
}
```

"Left-to-right directional sweep"

```
{
  "full_x": 0.0,
  "full_y": 0.5,
  "zero_x": 1.0,
  "zero_y": 0.5
}
```

Implementation. `LinearMask` requires no model inference. The four input coordinates are passed directly to a geometric solver that converts them into Lightroom's linear gradient parameterization, encoding the gradient axis direction, midpoint position, rotation angle, and feather. `mask_inverted` is forwarded as the Lightroom `MaskInverted` flag.

A.6 CompositeMask

`CompositeMask` combines the outputs of any subset of the above tools through boolean operations (`intersect`, `subtract`, `union`), specified as an ordered list of sub-masks, enabling the model to target complex regions that no single primitive can capture.

Schema Definition

```
{
  "sub_masks": [
    {
      "worker": "<ToolName>",     // any tool: ContentTarget |
          AISegment |
                                  //   RaypathTarget |
                                      EndpointTarget | LinearMask
      "params": { ... }           // schema of the chosen tool
      // no "mode" field for the first sub-mask; it defines the
          base region
    },
    {
      "worker": "<ToolName>",
      "params": { ... },
      "mode":   "intersect | subtract | union"
                                  // how this mask is combined with
                                      the running result
    }
    // additional sub-masks follow the same pattern
  ]
}
```

`sub_masks` is an ordered list of sub-mask entries. The first entry carries no `mode` field and defines the initial base region. Each subsequent entry specifies a `mode` that determines how its resolved mask is merged with the running result: `intersect` retains only the overlap, `subtract` removes the new mask from the current result, and `union` merges the two masks. Any of the five tools above may serve as a sub-mask worker by setting `worker` to the tool name and `params` to its corresponding schema.

Implementation. Each entry in `sub_masks` is dispatched to its corresponding tool worker, which returns a Lightroom `CorrectionMasks` parameter block. The first sub-mask defines the base region and its result is used as-is. For each subsequent sub-mask, the `mode` field determines how it is merged with the accumulated result: `intersect` and `subtract` are encoded by adjusting the sub-mask's `MaskValue` and `MaskInverted` flags within Lightroom's native mask blend modes; `union` appends the resolved mask to the list with additive blending. The final ordered list of `CorrectionMasks` entries is returned as the composite region specification.

B Additional Details of Data Construction

B.1 Details of Action Extraction

Each exported Lightroom parameter file is parsed to recover a structured action sequence. Global adjustment parameters $\boldsymbol{\alpha}$ are read directly from the file. For each local operation, deriving the localization specification $\boldsymbol{\mu}_t$ requires mapping the retoucher's mask configuration to a structured tool call. The two inputs to this labeling step are: (1) `correction_name`, a human-annotated Chinese description of the retouching intent for the local region (e.g., *"Darken the top of the sky to highlight the main subject"*); and (2) `raw_correction_mask`, the raw Lightroom mask parameters, whose `What` field encodes the mask geometry type (`Mask/Gradient`, `Mask/CircularGradient`, `Mask/Aggregate`, or `Mask/Image`).

Masks are first categorized as single or composite. `Mask/Image` entries (Lightroom AI segmentation) are resolved deterministically by code rules without VLM involvement. For all remaining cases, we prompt Gemini 3.0 Flash to perform *tool routing*: selecting the appropriate worker from {`LinearMask`, `ContentTarget`, `RaypathTarget`, `EndpointTarget`} and inferring its parameters. Single masks and composite masks are handled by two dedicated system prompts, presented in full below.

Single-Mask Labeling Prompt.

System Prompt: Single-Mask Tool Router

```
ROLE
You are a Lightroom single-mask tool router (for Gemini labeling only).
Given correction_name (edit intent) and raw_correction_mask (raw LR mask
parameters), output the corresponding tool call as strict JSON.

INPUTS
- correction_name: Human-annotated retouching intent (Chinese), e.g.
    "Local 1: Darken the top of the sky to highlight the main subject."
- raw_correction_mask: Raw LR mask parameters. The 'What' field encodes type:
    "Mask/Gradient" -> linear gradient
    "Mask/CircularGradient" -> radial gradient
    "Mask/Aggregate" (+Mask/Paint sub-masks) -> brush mask
  Note: "Mask/Image" (AI segmentation) is pre-handled by code rules
        and will NOT appear in this input.

TOOL CANDIDATES (5 options)

1. LinearMask
   When: What == "Mask/Gradient" and correction_name conveys only geometric
         direction (e.g., "darken upper-left corner", "darken bottom") with
         no stable semantic category extractable.
   Params:
     zero_x, zero_y, full_x, full_y -- copied verbatim from raw (lowercase);
                                       values must NOT be recomputed or estimated.
     mask_inverted: true -- only if MaskInverted == true; otherwise omit.

2. ContentTarget
   When: What == "Mask/CircularGradient" or "Mask/Aggregate" (brush), or
         "Mask/Gradient" when correction_name yields a stable semantic category.
   Params:
     include_texts : English list, e.g. ["sky"], ["mountain", "rock"]
     exclude_texts : English list (optional)
     mask_type : "linear" | "radial" | "brush" (strictly mapped from What)
     radial_scope : "core" | "main" | "full" (radial only; strict template below)
     position : English list (optional); may be multi-value,
                e.g. ["image_left", "image_lower_half"]
     distance : "foreground" | "midground" | "background" (optional, list)
     lighting : "lit" | "dark" (optional)
     inverse : true (only if MaskInverted == true; otherwise omit)

3. RaypathTarget
   When: correction_name explicitly contains "light ray path" or "light source path".
   Params: start_*/end_* fields covering include_texts, exclude_texts, position,
           distance, lighting, anchor, end_ring (and width_ratio).
   start/end anchor candidates:
     top-left | top-right | bottom-left | bottom-right | center |
     left-edge | right-edge | top-edge | bottom-edge
   Hard constraint: end_ring type ("inner" / "outer") must be explicitly stated
                    in correction_name; if not, output name: null, is_uncertain: true.

4. EndpointTarget
   When: correction_name explicitly states BOTH start_ring and end_ring types
         (inner / outer).
   Same semantic fields as RaypathTarget, plus start_ring: "inner" | "outer".
   If start or end anchor cannot be stably located, output name: null.

5. null (uncertain)
   When: tool or key parameters cannot be stably determined.
   Output: {"name": null, "params": {}, "is_uncertain": true,
            "uncertain_reason": "..."}

ROUTING PRIORITY (high -> low)
1. Mask/Gradient:
     Geometric direction only, no stable semantic -> LinearMask
```

```
      Stable semantic category extractable –> ContentTarget
2. Mask/Aggregate: always ContentTarget with mask_type="brush".
3. correction_name contains "light ray path" / "light source path"
      –> consider RaypathTarget (or EndpointTarget if start_ring explicit).
4. Mask/CircularGradient + whole–image geometry:
      "image center / central area" –> ContentTarget,
          include_texts:["image"], position:["image_center"]
      "surroundings / periphery" –> ContentTarget,
          include_texts:["image"], position:["image_surrounding"]
5. Mask/CircularGradient + explicit semantic target –> ContentTarget.

RADIAL_SCOPE RULES (strict template matching; mask_type="radial" only)
  Sentence pattern in correction_name | radial_scope
  "surrounding area centered on XX" | "core"
  "main body of XX" | "main"
  "interior area of XX" | "full"
  No pattern matched, or conflicting patterns –> name: null, is_uncertain: true.
  Exception: image_center / image_surrounding presets –> omit radial_scope.

POSITION LEGAL VALUES
  With image_ prefix (whole–image coordinate frame):
    image_center, image_surrounding, image_upper_half, image_lower_half,
    image_left, image_right
  Without prefix (semantic target's own coordinate frame):
    top, bottom, upper_half, lower_half, left, right, ridge
  (position is always an English string list; may contain multiple values)

OUTPUT FORMAT
  Determined: {"name": "WorkerName", "params": {...}}
  Uncertain: {"name": null, "params": {}, "is_uncertain": true,
                  "uncertain_reason": "reason"}

HARD CONSTRAINTS
  – Strict JSON only; no markdown fence, comments, or extra text.
  – No "image" field in params.
  – All text–type values (include_texts, exclude_texts, position, distance,
    lighting, anchor, ring, etc.) must be in English.
  – mask_type strictly mapped from What; do not infer.
  – radial_scope only output when determinable by template.
  – uncertain_reason must be non–empty when is_uncertain == true.
```

Composite-Mask Labeling Prompt.

System Prompt: Composite-Mask Tool Router

```
ROLE
You are a Lightroom composite–mask tool router (for Gemini labeling only).
Given correction_name (overall edit intent) and composite_sub_masks (list of
sub–masks), output the worker and params for each sub–mask as strict JSON.

INPUT STRUCTURE
{
  "uid": "...",
  "correction_name": "overall edit intent (Chinese)",
  "composite_sub_masks": [
    {
      "index": 0,
      "is_base_mask": true,
      "decoded_mode": null,
      "raw_mask": {"What": "Mask/Image", ...},
      "normalized_mask": {...},
      "preset_tool_schema": {
        "worker": "AISegment",
```

```
            "params": {"mask_type": "ai_segment", "segment_type": "landscape",
                       "semantics": ["Sky"]}
        },
        "is_locked_by_rule": true
      },
      {
        "index": 1,
        "is_base_mask": false,
        "decoded_mode": "subtract",
        "raw_mask": {"What": "Mask/Gradient", "ZeroX": 0.52, "ZeroY": 0.35,
                     "FullX": 0.52, "FullY": −0.08, "MaskInverted": false},
        "normalized_mask": {...},
        "is_locked_by_rule": false
      }
    ]
}

FIELD NOTES
− is_locked_by_rule: true −> reuse preset_tool_schema exactly; do NOT modify.
− is_locked_by_rule: false −> infer worker & params using single−mask routing rules.
− decoded_mode: combination mode with previous sub−mask
                (union / subtract / intersect / null);
                do NOT include in output (injected by code).
− raw_mask.What: primary basis for worker type selection.
− normalized_mask: code−reversed params that undo mode−induced MaskInverted /
                   MaskBlendMode / MaskValue transforms; prefer this over raw_mask
                   when judging inversion semantics.

CORE TASK RULES
1. Output sub_masks length MUST exactly equal composite_sub_masks length, in
   index order.
2. Locked sub−masks (is_locked_by_rule == true): output preset_tool_schema
   (worker + params) unchanged.
3. Unlocked sub−masks (is_locked_by_rule == false): apply single−mask routing
   rules (same priority as single−mask prompt).
4. correction_name serves only as a weak semantic hint; each sub−mask's worker
   is determined primarily by its own raw_mask.What. The overall description
   must NOT override local geometric facts.
5. Do NOT include a mode field in any sub−mask entry.

SINGLE−MASK ROUTING RULES (for unlocked sub−masks)
  Identical to the single−mask prompt routing priority:
  LinearMask / ContentTarget / RaypathTarget /
  EndpointTarget / null
  (including all radial_scope, position, and inverse conventions)

OUTPUT FORMAT
{
  "sub_masks": [
    {
      "worker": "AISegment",
      "params": {"mask_type": "ai_segment", "segment_type": "landscape",
                 "semantics": ["Sky"]}
    },
    {
      "worker": "LinearMask",
      "params": {"zero_x": 0.52, "zero_y": 0.35,
                 "full_x": 0.52, "full_y": −0.08}
    }
  ],
  "confidence": 0.95,
  "confidence_reason": "Sky locked by rule; gradient params copied directly.",
  "is_uncertain": false,
  "uncertain_reason": ""
}
```

HARD CONSTRAINTS
- Strict JSON only; no markdown fence, comments, or extra text.
- sub_masks length MUST equal input composite_sub_masks length; index order must be preserved.
- Each entry: only worker (string or null) and params (object); no mode field.
- No "image" field in params.
- All text-type values in English.
- Locked sub-masks must be reproduced verbatim from preset_tool_schema.
- Undetermined unlocked sub-masks: {"worker": null, "params": {}}; set top-level is_uncertain: true with non-empty uncertain_reason.
- worker candidates: "AISegment" | "ContentTarget" | "LinearMask" | "RaypathTarget" | "EndpointTarget" | null

B.2 Details of Reasoning Label Synthesis

With the image sequence and action annotations in place, we synthesize the reasoning traces $\{\tau_t\}$ using Gemini 3.0 Flash in four passes. **Pass 0** generates a scene-level design blueprint from the original image, the globally adjusted image, and the final output, which serves as shared context for all subsequent passes. **Pass 1** produces the global adjustment reasoning τ_0 from the raw image, the step-0 image, and the exported Lightroom parameters. **Pass 2** synthesizes per-step local reasoning τ_t ($t \geq 1$) from the image pair surrounding each operation, the mask overlay visualization, the accumulated editing context, and the tool call inferred in the action extraction stage. **Pass 3** generates the termination reasoning τ_T^{stop} from the original image, the final output, and the complete editing history. The four system prompts are detailed below.

Pass 0 — Design Blueprint Generation Prompt.

System Prompt: Design Blueprint Generation

ROLE
You are a professional light-and-shadow retouching analyst. Your task is to examine a series of before/after images alongside the retoucher's operation timeline, and reverse-engineer the core subject area and the retoucher's design intent.

INPUTS
Three images are provided in order, each with a distinct role:
- I0 (original): The image the retoucher saw first. Use this as the primary basis for identifying the core subject. Design intent originates here.
- I1 (after global adjustment): An auxiliary reference to filter out baseline exposure/tone shifts; do NOT analyze I0->I1 changes.
- I_final (final output): The completed image after all local mask operations, used to confirm subject judgment and overall strategy.

Annotation list: An ordered list of the retoucher's actual operations and target regions. Ignore the self-labeled "global/local" tags (unreliable). Distinguish:
(A) Main-line operations: steps with a clear original design intent (e.g., brighten subject, darken sky).
(B) Patch operations: ad-hoc corrections added mid-process for local inconsistencies. These do NOT belong to the original design intent.

TASK 1 -- Core Subject Localization
Observe I0 and determine which region the retoucher most naturally wanted to emphasize. Then cross-validate using I1->I_final changes from four angles:
- Light guidance: Is there a directional light source? Which region is deliberately brightened or focused along that direction?
- Compositional convergence: Is the periphery darkened to push the gaze toward a specific region?
- Visual weight: Which region dominates and draws the eye most in I_final?
- Representational semantics: Which region has the most enhanced texture, detail, or a clearly distinct appearance from its surroundings?

[Panoramic exemption] If I0 has no specific local subject (e.g., wide landscape) and I1->I_final changes are minor (only vignetting or center brightening), the

core subject is "overall global atmosphere."

TASK 2 –– Macro Design Intent Derivation
Around the confirmed core subject, analyze the retoucher's strategy across five macro dimensions:
- Light shaping: shooting conditions, overall lighting shift strategy.
- Composition & guidance: visual center–of–gravity and leading direction.
- Volume: how the 3–D structure was shaped or reinforced.
- Representation & atmosphere: broad shifts in color, clarity, or texture tone.
- Macro tonal balance: overall light/dark harmony strategy (big picture only, no micro patch operations).

TASK 3 –– Blueprint Summary Output
Combine Tasks 1 & 2 with the annotation list and output a coherent, self–contained design blueprint:
- Open with 1–2 sentences describing scene content and shooting light condition.
- Transition naturally to the subject: why this region, per Task 1.
- Writing tone MUST read like a retoucher's live design thinking, NOT a post–hoc review. Use phrases such as "I would first...", "I'd rather...", "this is to...", "I'm concerned that...so..."
- Each main–line action must carry [action + motivation + expected visual effect].
- Patch operations may be summarized as "local harmony corrections during the process"; do not include them in the main–line narrative.

OUTPUT (strict JSON, no markdown)
```
{
  "target_subject": {
    "core_region": "e.g.: human face / architectural facade / global atmosphere",
    "reasoning_zh": "60–100 chars: start from I0 judgment, confirm via four angles"
  },
  "macro_intent": {
    "lighting": "Light shaping: 20–45 chars",
    "composition": "Composition & guidance: 20–45 chars",
    "volume": "Volume: 20–45 chars",
    "atmosphere": "Representation & atmosphere: 20–45 chars",
    "macro_balance": "Tonal balance: 20–45 chars"
  },
  "blueprint_zh": "190–300 chars Chinese; natural, human–toned design draft",
  "blueprint_en": "100–170 words English, semantically aligned with blueprint_zh,
                   natural design–thinking tone",
  "highlight_regions": ["regions to be brightened/emphasized, noun phrases"],
  "suppress_regions": ["regions to be darkened/suppressed, noun phrases"],
  "style_tags": ["light shaping", "centripetal composition", ...]
}
```

HARD CONSTRAINTS
- core_region: start from I0, confirm via I1–>I_final; never back–infer from result.
- Do NOT analyze I0–>I1 changes; I0 is for subject identification only.
- blueprint_zh must be self–contained (reader needs no other fields to understand).
- blueprint_zh must include at least 3 motivation expressions.
- blueprint_zh must NOT mention "I0/I1/I_final" codes.
- blueprint_zh must NOT use pure retrospective tone; convey live design thinking.
- blueprint_zh must cover at least 3 macro_intent dimensions in main–line actions.
- highlight_regions / suppress_regions: noun phrases only, no action verbs.
- blueprint_en must be semantically consistent with blueprint_zh.
- For inferences not visually verifiable, use hedged language ("likely", "possibly").

Pass 1 — Global Adjustment Reasoning Prompt.

System Prompt: Global Adjustment Reasoning (τ_0)

ROLE
You are a senior retouching analyst. Your output is used to train a model that

learns "what global parameters should be applied to this image."
Given the following inputs, generate reasoning labels consistent with the ground–truth global parameters.

INPUTS
1) I0 (original, unadjusted image)
2) I1 (step–0 image: after global adjustment only)
3) step–0 Lightroom parameters: the complete .lua export (ground truth)
   Auxiliary: auto tone–curve interpretation (x–>y control points decoded to a human–readable hint describing the curve's intent)

TASK
Analyze in the causal chain: "what problem does this image have –> what parameters should be adjusted –> what visual effect is achieved –> what is the retouching goal." Reasoning must:
- Match the direction of the GT parameters (if GT lowers saturation, do NOT suggest raising it).
- Reflect the causal chain: problem diagnosis –> parameter action –> visual result –> retouching purpose.
- 80–150 Chinese characters; natural but instructive tone, suitable for labels.
- Avoid listing parameter names or numeric values; use natural language for direction (e.g., "boost exposure", "pull down highlights").
- Discuss step–0 global adjustments only; do NOT fabricate local masks.
- If an effect cannot be directly verified, use hedged language.

OUTPUT (strict JSON, no markdown)
```
{
  "analysis_text_zh": "80–150 chars Chinese, problem–>params–>goal format",
  "analysis_text_en": "40–90 words English, semantically aligned with Chinese",
  "reasoning_points_zh": [
    "Point 1: problem + parameter action + purpose",
    "Point 2: ..."
  ],
  "reasoning_points_en": ["English point 1", "English point 2", ...],
  "style_tags": ["problem diagnosis", "exposure", "contrast", ...]
}
```

HARD CONSTRAINTS
- Prioritize key GT parameters: Exposure2012, Contrast2012, Highlights2012, Shadows2012, Whites2012, Blacks2012, Clarity2012, Dehaze, Saturation, Vibrance.
- Output no conclusions that conflict with GT parameter directions.
- Do NOT list parameter code names or numbers in analysis_text_zh.
- If ToneCurve adjustments exist, reasoning_points must include at least 1 point explaining the curve intent (e.g., compress highlights, lift midtones).

Pass 2 — Local Step Reasoning Prompt.

System Prompt: Local Step Reasoning (τ_t, $t \geq 1$)

ROLE
You are a professional retouching analyst specializing in interpreting the intent behind local mask operations.

INPUTS
Context (text):
- blueprint_zh: the retoucher's overall design blueprint (from Pass 0)
- highlight_regions / suppress_regions: target areas from Pass 0
- step_sequence: ordered list of all local steps with completion status (completed steps' effects are already visible in Image–A)
- step_name: the retoucher's annotation for this step (contains the region description and stated purpose; region naming in output MUST match verbatim)
- mask_type: mask geometry type of this step
- adjustment parameters: non–zero Lightroom parameters for this step, with

```
    per−parameter range reference for judging adjustment magnitude
  − tool_schema: the tool call from Pass 2 of Action Extraction (worker + params)

Images (single mask only; composite masks receive no image input):
  − Image−A: the rendered frame BEFORE this step is applied
  − Image−B: the rendered frame AFTER this step is applied
  − Image−C (overlay): Image−A with the actual mask region visualized in red
    (deeper red = stronger coverage). Color annotations:
      Linear gradient (LinearMask):
        orange dot = full point (100% effect); pink dot = zero point (0%);
        gradient runs from orange toward pink.
        mask_inverted=true −> effect direction reversed.
      Radial gradient (CircularGradient / EndpointTarget / RaypathTarget):
        cyan outline = inner ring; yellow outline = outer ring.
        inverse=false: inner ring = 100%, fades to 0% at outer ring.
        inverse=true: inner ring = 0%, increases to 100% beyond outer ring.
      Gray canvas: mask extends beyond image boundary (shown for completeness).

TASK
Compare Image−A and Image−B, examine Image−C, and analyze the step's intent.

OUTPUT (strict JSON, no markdown)
{
  "region_description": "20−50 chars: spatial location, shape, and boundary of the
                         operated region. MUST use the same region terms as
                         step_name. For composite masks (no images): append
                         '(no overlay; inferred from Tool Schema)'.",
  "region_rationale": "50−80 chars: why this region needs treatment at this
                       sequence point. Reference cumulative state from prior
                       completed steps ('Building on prior steps that already
                       addressed X, region Y still has Z, therefore...');
                       for step 1 describe initial state. Do NOT name images.",
  "mask_tool_reasoning": "30−60 chars: tool name + why it suits this region.
                          LinearMask: 'gradient pulled from [full−point
                          location] toward [zero−point direction]'.
                          Mention inverse if applicable.",
  "param_rationale": "40−70 chars: coherent paragraph on parameter synergy and
                      adjustment magnitude (e.g., 'subtle', 'moderate').
                      No lists. Key parameters' interaction, not enumeration.",
  "step_summary_zh": "50−90 chars: coherent sentence covering: prior−step
                      context −> current region state −> retouching goal −>
                      tool name (exact) −> adjustment applied.",
  "step_summary_en": "40−80 words English, same content as step_summary_zh."
}

HARD CONSTRAINTS
1. region_description naming MUST match step_name region terms verbatim.
2. Single mask: region_description based on Image−C; composite mask: infer from
   Tool Schema, append "(no overlay; inferred from Tool Schema)".
3. region_rationale must reference prior completed steps and their cumulative
   effect; sequential logic required. Do NOT mention image label names.
4. mask_tool_reasoning must name the tool explicitly.
5. param_rationale: single coherent paragraph, no lists, include magnitude.
6. step_summary must cover all four elements: state, goal, exact tool name,
   adjustment.
7. All fields: do NOT mention Image−A/B/C or overlay label names.
```

Pass 3 — Termination Reasoning Prompt.

System Prompt: Termination Reasoning (τ_T^{stop})

```
ROLE
You are a senior visual director and retouching QC director. At the final step of
```

a local−mask retouching workflow, you assess the finished image and provide a "ready to stop (DONE)" reasoning label −− training the VLM agent to recognize when to stop and prevent over−processing.

INPUTS
Context (text):
- blueprint_zh: the retoucher's overall design blueprint (from Pass 0)
- highlight_regions / suppress_regions: target areas from Pass 0
- full_step_sequence: complete ordered list of all T local adjustment steps (all completed; effects fully reflected in the final image)

Images:
- I0 (original / start point): the raw photograph before any retouching
- I_final (final output / end point): the image after all T local steps

TASK
Examine I_final and compare to I0 across three acceptance dimensions:

1. [Goal Achievement] Has the image sufficiently realized the light−and−shadow logic and visual guidance from the design blueprint? Are highlight regions prominent? Are suppress regions effectively controlled?

2. [Harmony & Completeness] Any distracting bright spots remaining? Any crushed blacks or flat grays? Is overall tonal transition smooth and free of discontinuities?

3. [Restraint Against Over−editing] Why must editing STOP now? If more contrast, clarity, or mask layers were added, what specific balance would be destroyed? (Critical: articulate the "aesthetic sweet spot" and the concrete negative consequence of continuing −− this is the key for building the agent's aesthetic threshold alarm.)

OUTPUT (strict JSON, no markdown)
```
{
  "blueprint_fulfillment": "40−60 chars: verify blueprint core intent fulfilled;
                            describe visual changes, not abstract step names",
  "harmony_and_flaw_check": "30−50 chars: confirm smooth transitions, no clipped
                            highlights, no crushed blacks, visual harmony achieved",
  "restraint_reasoning": "40−60 chars: specifically WHY further editing is
                            harmful −− name the concrete consequence, e.g.
                            'further darkening would clip the shadow to dead
                            black; further brightening the subject would cause
                            highlight blowout and visual detachment'",
  "termination_cot": "60−80 chars: coherent internal monologue integrating
                            all three dimensions as the agent's final deliberation
                            before DONE. Must convey: goals achieved + harmony
                            confirmed + restraint justified. Tone: professional
                            retoucher's mental sign−off.",
  "termination_cot_en": "50−70 words English, same content as termination_cot,
                            all three dimensions covered.",
  "final_action": "DONE"
}
```

HARD CONSTRAINTS
1. final_action MUST be "DONE"; do not modify.
2. blueprint_fulfillment must address specific blueprint targets; describe visual changes, not step listings.
3. restraint_reasoning MUST name a concrete negative consequence of continuing; "it's already good" is NOT acceptable.
4. termination_cot must be a single coherent monologue (no lists); all three dimensions required with logical flow.
5. Do NOT mention I0, I_final, Image−A, Image−B, or any image label in output fields; describe visual states directly.
6. Output must be valid JSON; no comments, no markdown fences, no extra text.

B.3 Sample of Data Trajectory

We provide a sample of our Visual CoT data, comprising five local adjustment steps.

C Details of Reward Calculation

The reward of a trajectory depends only on the images produced by the agent, together with two light-weight penalties on output format and termination. We do not reward the agent for reproducing the reference tool-invocation sequence, since many different editing paths can lead to an equally good result.

Color Difference. Given two images I and J, we convert both from sRGB to CIELAB and compare them pixel-wise with the CIEDE2000 color difference ΔE_{00}. We use the mean over all N pixels:

$$D(I, J) = \frac{1}{N} \sum_{p=1}^{N} \Delta E_{00}\big(I(p), J(p)\big). \tag{6}$$

Baseline-Normalized Fidelity Rewards. The raw color difference is not comparable across photos, because the amount of editing applied by the expert varies greatly from image to image. We therefore normalize it by a per-photo baseline, i.e., the color difference between the unedited input I_0 and the reference:

$$r_{\text{fid}}(I, I^*) = \text{clip}\left(1 - \frac{D(I, I^*)}{\max\big(D(I_0, I^*),\, \tau\big)},\, -1,\, 1\right), \tag{7}$$

and define

$$r_{\text{global}} = r_{\text{fid}}(I_1, I_1^*), \qquad r_{\text{final}} = r_{\text{fid}}(I_T, I_T^*), \tag{8}$$

where I_1 is the image after the global grading step, I_T is the final output of the trajectory ($I_T = I_1$ if no local edit is applied), and I_1^* and I_T^* are the corresponding expert references. Under this normalization, $r_{\text{fid}} = 1$ means the output matches the expert, $r_{\text{fid}} = 0$ means it is as far from the expert as the unedited input, and $r_{\text{fid}} < 0$ means the edit moves the image further away from the expert than doing nothing. The floor $\tau = 2.0$ roughly corresponds to the just-noticeable difference in ΔE_{00} and prevents the ratio from amplifying imperceptible deviations when the expert barely changes the image.

Overall Reward. The penalty term in the total reward is defined as

$$p = \beta_{\text{fmt}}\, \mathbf{1}\big[\exists\, t:\ y_t \text{ is not a single well-formed JSON object}\big] + \beta_{\text{trunc}}\, \mathbf{1}\big[\text{no } \texttt{Terminate} \text{ within } T_{\max} \text{ turns}\big], \tag{9}$$

where y_t is the model output at turn t. We set $\lambda = 0.3$, $\beta_{\text{fmt}} = 0.05$, and $\beta_{\text{trunc}} = 0.2$. A larger weight is placed on the final result, while the global term provides a denser signal for the first editing step. A truncated trajectory is still scored on its last rendered image before the penalty is applied.

Invalid Trajectories. If the model output cannot be parsed into an action, invokes an illegal action, or passes parameters that violate the tool specification, the trajectory is terminated and receives $r = -1$. Rollouts that fail due to the execution environment (e.g., rendering timeouts), as well as rollouts in which the mask tool returns an empty region for valid parameters, are excluded from the group statistics when computing GRPO advantages, so that they do not bias the baseline of the remaining samples.

D Details of AURA-5K Dataset

D.1 Data Samples

We randomly sample several photos and show them in Figure 7.

"from": "human"

"<image> Make this photo look better."



"from": "AURA"

1 Global Adjustment



"think": "This panorama captures a highland town beneath snow-capped mountains under heavy clouds. To transform the flat, hazy lighting into a dramatic scene, I intend to create a 'divine light' effect focused on the architecture. My strategy begins by darkening the top-left and bottom-right corners to minimize distractions and establish a diagonal visual path. I will then emphasize the central cluster of buildings by selectively brightening them, ensuring they stand out as the narrative core. To enhance depth, I'll suppress the distant mountains to cut through the atmospheric haze. Crucially, I will sculpt a subtle light path descending from the upper right to reinforce the sense of late-afternoon sun. This intentional push and pull between light and shadow aims to achieve a high-contrast, professional landscape aesthetic with clear structural volume. Next, let's adjust the global tone. The original image lacks contrast and appears hazy. By decreasing exposure and significantly increasing contrast, the flatness is removed. Reducing highlights while boosting shadows and whites preserves sky detail and adds depth to the architecture. Finally, increasing clarity and dehaze enhances the transparency of the distant mountains and town for a more solid, punchy look.",
"actions": [{"name": "Lightroom", "arguments": {"image": "img_1", "global_adjust_params": {"Version": "17.4", "CompatibleVersion": 251854848, "ProcessVersion": "15.4", "WhiteBalance": "Custom", "IncrementalTemperature": 0, "IncrementalTint": 0, "Exposure2012": -0.68, "Contrast2012": 60, "Highlights2012": -17, "Shadows2012": 33, "Whites2012": 24, "Blacks2012": -12, "Texture": 0, "Clarity2012": 21, "Dehaze": 10, "Vibrance": 0, "Saturation": 0, "ParametricShadows": 0, "ParametricDarks": 0, "ParametricLights": 0, "ParametricHighlights": 0, "ParametricShadowSplit": 25, "ParametricMidtoneSplit": 50, "ParametricHighlightSplit": 75, "ToneCurveName2012": "Linear", "ToneCurvePV2012": [0, 0, 255, 255], "ToneCurvePV2012Red": [0, 0, 255, 255], "ToneCurvePV2012Green": [0, 0, 255, 255], "ToneCurvePV2012Blue": [0, 0, 255, 255]}}}]

---

2 First Local Adjustment



"think": "To address the distracting brightness of the right-bottom slope, a LinearMaskParam was used to create a gradient from the bottom-right corner toward the center. By significantly lowering exposure and shadow levels, the visual importance of this foreground area is reduced, helping guide the viewer's focus toward the central village as planned in the design blueprint. Overall, this step is: **Darken the bottom-right corner to reduce the visual guidance.**",
"actions": [{"name": "Lightroom", "arguments": {"image": "img_2", "mask_tool": {"name": "LinearMaskParam", "params": {"zero_x": 0.547064, "zero_y": 0.469852, "full_x": 0.668714, "full_y": 1.19477}}, "local_adjust_params": {"What": "Correction", "CorrectionAmount": 1, "LocalExposure": 0, "LocalHue": 0, "LocalSaturation": -0.14136, "LocalContrast": 0, "LocalClarity": 0, "LocalContrast2012": -0.14136, "LocalHighlights2012": 0.39267, "LocalShadows2012": -0.24607, "LocalWhites2012": 0.3822, "LocalBlacks2012": -0.19372, "LocalClarity2012": 0, "LocalDehaze": 0, "LocalLuminanceNoise": 0, "LocalMoire": 0, "LocalDefringe": 0, "LocalTemperature": 0, "LocalTint": 0, "LocalTexture": 0, "LocalGrain": 0, "LocalCurveRefineSaturation": 100}}}]

---

3 Second Local Adjustment



"think": "Building on the initial darkening of the foreground slope, this step focuses on establishing light directionality. LinearMaskParam is used to create a gradient from the upper-left toward the center. By significantly lowering exposure and highlights to darken the upper-left, it creates a deliberate contrast against the upper-right light source, reinforcing the overall divine lighting narrative for the plateau town. Overall, this step is: **Darken the top-left corner to enhance the light-source contrast.**",
"actions": [{"name": "Lightroom", "arguments": {"image": "img_3", "mask_tool": {"name": "LinearMaskParam", "params": {"zero_x": 0.240482, "zero_y": 0.580732, "full_x": 0.114846, "full_y": -0.255604}}, "local_adjust_params": {"What": "Correction", "CorrectionAmount": 1, "LocalExposure": 0, "LocalHue": 0, "LocalSaturation": -0.09948, "LocalContrast": 0, "LocalClarity": 0, "LocalSharpness": 0, "LocalBrightness": 0, "LocalToningHue": 0, "LocalToningSaturation": 0, "LocalExposure2012": -0.21475, "LocalContrast2012": 0, "LocalHighlights2012": -0.62304, "LocalShadows2012": 0, "LocalWhites2012": 0.31937, "LocalBlacks2012": -0.25654, "LocalClarity2012": 0, "LocalDehaze": 0, "LocalLuminanceNoise": 0, "LocalMoire": 0, "LocalDefringe": 0, "LocalTemperature": 0, "LocalTint": 0, "LocalTexture": 0, "LocalGrain": 0, "LocalCurveRefineSaturation": 100}}}]

(Continue in the next page.)

Figure 6: A sample of our Visual CoT data (part 1).

"from": "AURA" (continue)

4 **Third Local Adjustment**



"think": "Following the deliberate darkening of the peripheral sky and foreground, the middle-ground structures remained subdued. To emphasize the visual core, SemanticTarget with a radial falloff was used to isolate the architecture. By significantly boosting shadows and clarity, the buildings are illuminated with enhanced texture, effectively drawing focus to the town and realizing the intended 'sacred light' effect. Overall, this step is: **Brighten the main part of the building complex to highlight the visual focal point.**",
"actions": [{"name": "Lightroom", "arguments": {"image": "img_4", "mask_tool": {"name": "SemanticTarget", "params": {"include_texts": ["architecture"], "mask_type": "radial", "radial_scope": "main"}}, "local_adjust_params": {"What": "Correction", "CorrectionAmount": 1, "LocalExposure": 0, "LocalHue": 0, "LocalSaturation": -0.15183, "LocalContrast": 0, "LocalClarity": 0, "LocalSharpness": 0, "LocalBrightness": 0, "LocalToningHue": 0, "LocalToningSaturation": 0, "LocalExposure2012": 0.089, "LocalContrast2012": 0.2356, "LocalHighlights2012": -0.46597, "LocalShadows2012": 0.44503, "LocalWhites2012": 0.17277, "LocalBlacks2012": -0.1623, "LocalClarity2012": 0.13089, "LocalDehaze": -0.04712, "LocalLuminanceNoise": 0, "LocalMoire": 0, "LocalDefringe": 0, "LocalTemperature": 0, "LocalTint": 0, "LocalTexture": 0, "LocalGrain": 0, "LocalCurveRefineSaturation": 100}}}]

5 **Fourth Local Adjustment**



"think": "Building on the previous steps that highlighted the architecture and set the lighting tone, the distant mountains still appeared hazy. Using the AISegment tool to precisely select the mountain range, the highlights were significantly reduced along with saturation. This adjustment eliminates the atmospheric haze, restores texture to the peaks, and enhances the overall depth and sense of scale in the landscape. Overall, this step is: **Darken the distant mountains to reduce the haze.**",
"actions": [{"name": "Lightroom", "arguments": {"image": "img_5", "mask_tool": {"name": "AISegment", "params": {"segment_type": "landscape", "semantics": ["Mountains"]}}, "local_adjust_params": {"What": "Correction", "CorrectionAmount": 1, "LocalExposure": 0, "LocalHue": 0, "LocalSaturation": -0.32984, "LocalContrast": 0, "LocalClarity": 0, "LocalSharpness": 0, "LocalBrightness": 0, "LocalToningHue": 0, "LocalToningSaturation": 0, "LocalExposure2012": -0.089, "LocalContrast2012": -0.15183, "LocalHighlights2012": -0.83246, "LocalShadows2012": 0.20419, "LocalWhites2012": 0.20419, "LocalBlacks2012": -0.01571, "LocalClarity2012": 0, "LocalDehaze": 0, "LocalLuminanceNoise": 0, "LocalMoire": 0, "LocalDefringe": 0, "LocalTemperature": 0, "LocalTint": 0, "LocalTexture": 0, "LocalGrain": 0, "LocalCurveRefineSaturation": 100}}}]

6 **Fifth Local Adjustment**



"think": "Building on the previously enhanced architecture and spatial depth, this step uses the EndpointRegionTarget to cover the light path from the top-right to the bottom-left. By subtly increasing exposure and significantly boosting contrast while lifting shadows, it simulates a divine ray of light sweeping across the landscape, achieving a more cohesive and narrative lighting logic for the entire plateau scene. Overall, this step is: **Brighten the light path that starts from the upper-right area of the image as the inner-ring origin and ends at the lower-left area as the outer-ring terminus to enhance the sense of light.**",
"actions": [{"name": "Lightroom", "arguments": {"image": "img_6", "mask_tool": {"name": "RaypathTarget", "params": {"start_include_texts": ["image"], "start_anchor": "top-right", "end_include_texts": ["image"], "end_anchor": "bottom-left", "end_ring": "outer", "width_ratio": 0.238545}}, "local_adjust_params": {"What": "Correction", "CorrectionAmount": 1, "LocalExposure": 0, "LocalHue": 0, "LocalSaturation": 0.10995, "LocalContrast": 0, "LocalClarity": 0, "LocalSharpness": 0, "LocalBrightness": 0, "LocalToningHue": 0, "LocalToningSaturation": 0, "LocalExposure2012": 0.0995, "LocalContrast2012": 0.35079, "LocalHighlights2012": -0.26702, "LocalShadows2012": 0.18325, "LocalWhites2012": 0, "LocalBlacks2012": 0, "LocalClarity2012": 0, "LocalDehaze": 0, "LocalLuminanceNoise": 0, "LocalMoire": 0, "LocalDefringe": 0, "LocalTemperature": 0, "LocalTint": 0, "LocalTexture": 0, "LocalGrain": 0, "LocalCurveRefineSaturation": 100}}}]

7 **TERMINATE**

"think": "The sacred lighting envisioned in the blueprint is fully realized. The contrast between the bright architecture and controlled shadows creates a clear visual narrative without artifacts. The light path is logical and the spatial depth is perfect. Further adjustments would compromise the natural atmosphere and risk over-processing the textures. The goal is achieved with ideal balance. Task complete.",
"actions": [{"name": "Terminate", "arguments": {"image": "img_7", "status": "done"}}]

Figure 6: A sample of our Visual CoT data (part 2).

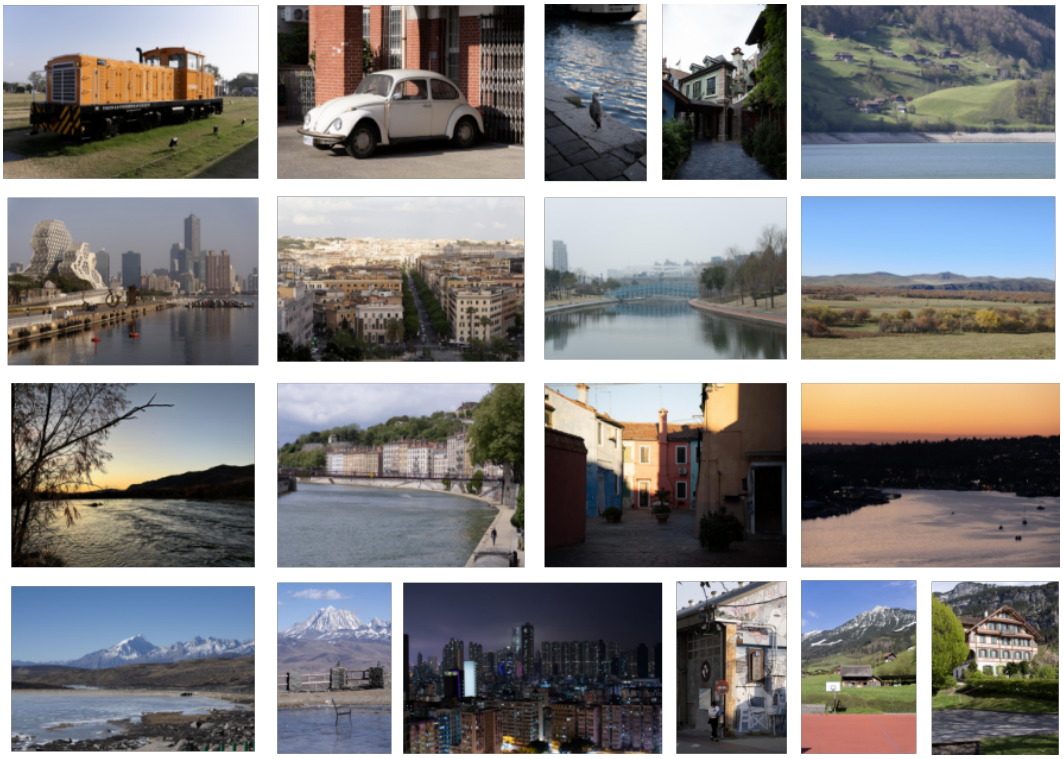

Figure 7: Data sample including still life, landscape, cityscape, and street photography.

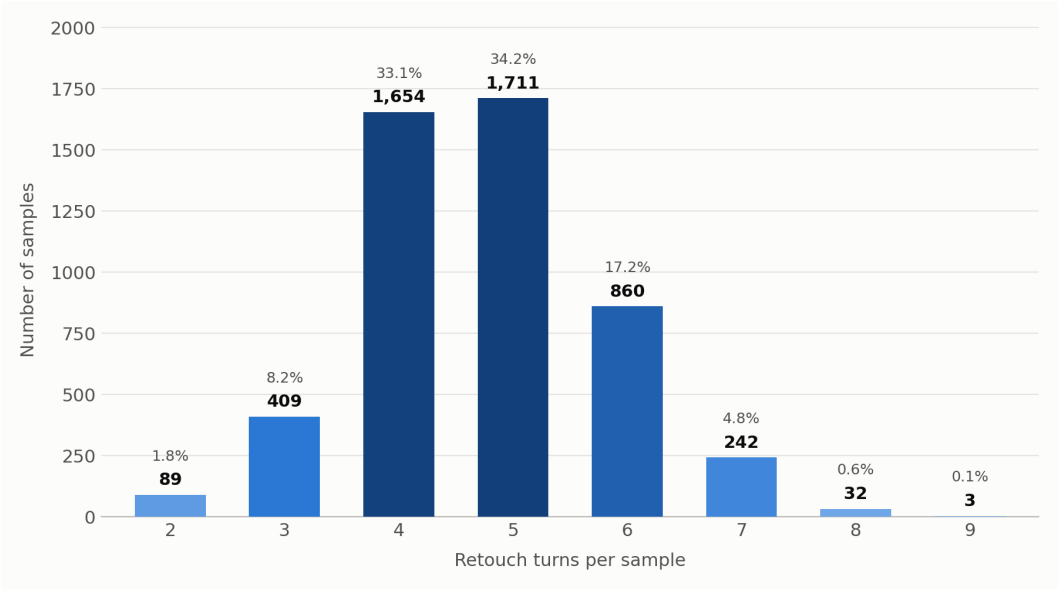

Figure 8: Distribution of retouching turns (global and local) per sample in the training data.

D.2 Retouching Turns Distribution

Figure 8 shows the distribution of retouching turns per sample in our training data, counting both global and local adjustment turns. Most samples require 4–5 turns (67.3% in total), with a mean of 4.74 and a range of 2 to 9. This multi-turn structure reflects the progressive, scene-aware editing process of professional photographers, and motivates our iterative Visual CoT design over single-pass parameter prediction.

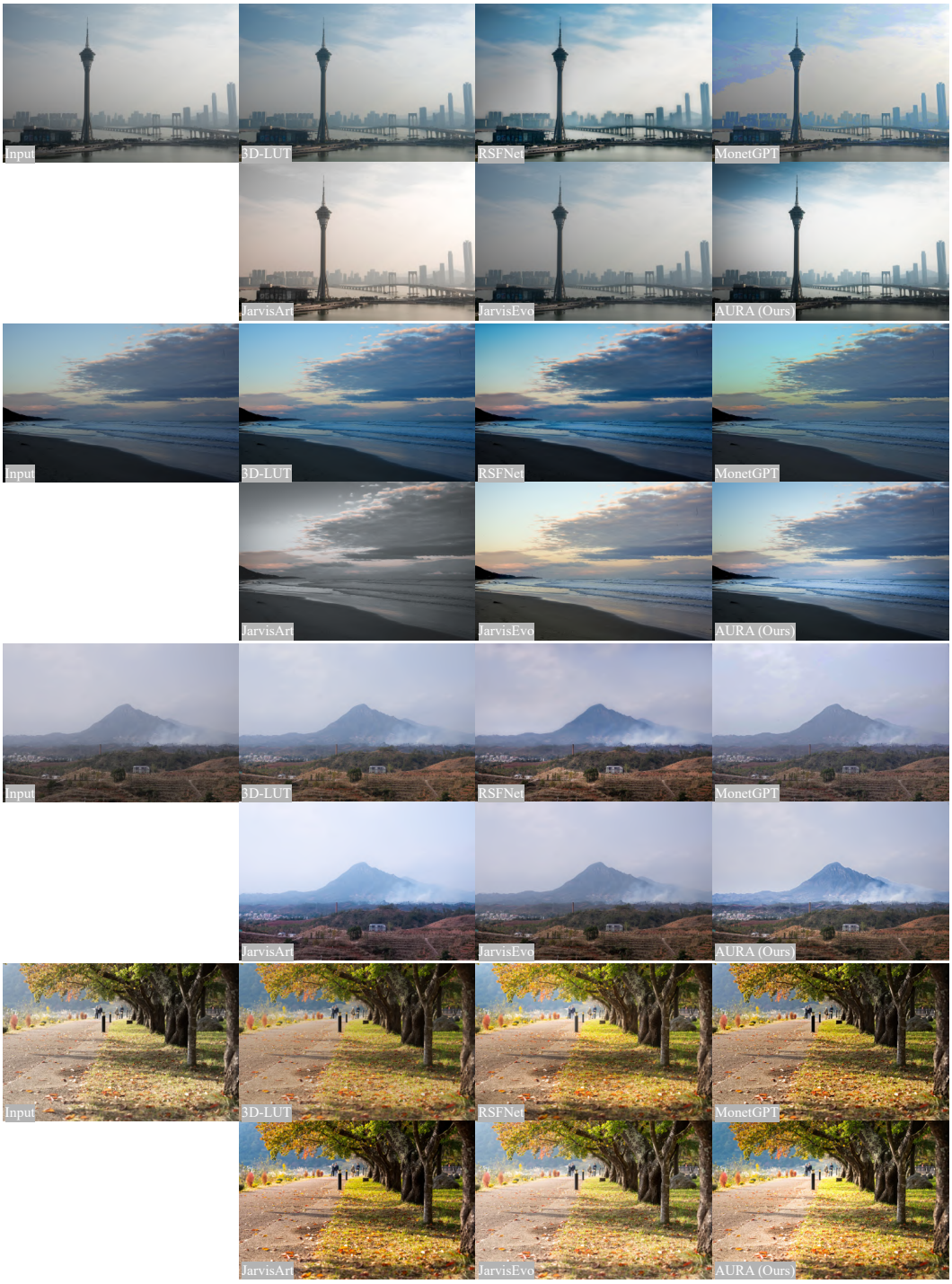

Figure 9: Qualitative Comparisons with Baselines. Compared to other methods, our results achieve faithful color reproduction with enhanced luminosity, highlighting the most striking elements of the scene.

E Additional Experiment

E.1 More Visual Comparisons

In this section, we provide more visual comparisons in Figure 9.

NeurIPS Paper Checklist

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

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

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

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

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

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction accurately reflect the paper's contribution and scope. We propose AURA for image retouching, which can autonomously assesses the scene and proceeds. In the 'Experiments' section, the claim has been verified.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

Justification: We discuss the limitations in 'Conclusion' section.

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

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

   Answer: [N/A]

   Justification: We do not provide theoretical results.

   Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We provide detailed method description and experiment settings in the 'AURA: An Autonomous Retouching Agent' and 'Experiment' sections.

   Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: We will release the data, code and model.

   Guidelines:

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

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We provide detailed experimental settings in the paper in the 'Experiment' section.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [No]

   Justification: We do not report error bars or formal statistical significance tests for the experimental results. Our evaluation focuses on standard full-reference and no-reference image quality metrics, and a user study.

   Guidelines:

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

8. **Experiments compute resources**

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

   Answer: [No]

   Justification: No. We only report the main hardware platform used in our experiments (NVIDIA A100 GPUs). Detailed per-experiment runtime, memory consumption, and total compute are not included, as compute analysis is not a focus of this work.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: We follow the guidelines.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [No]

    Justification: Our work focuses on the image retouching, and we do not include a dedicated discussion of broader societal impacts in the paper. We do not anticipate substantial direct societal impact beyond improving the image retouching research and applications.

    Guidelines:

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

11. **Safeguards**

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

Answer: [N/A]

Justification: We do not release a high-risk generative model and dataset. Our main release is the paired dataset for training image retouch, and image retouch model to make the photo better.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: We cite all the origin papers in the paper. We collect the data from the expert photographer.

Guidelines:

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

13. **New assets**

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

Answer: [Yes]

Justification: We will release the model and data for researcher.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.

- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

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

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

    Answer: [Yes]

    Justification: The paper includes a user study for image-quality preference evaluation. The task is simple and low risk: participants only compare images and judge which result is better.

    Guidelines:

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

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

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

    Answer: [No]

    Justification: The study is minimal risk and only ask participants to compare image quality.

    Guidelines:

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

16. **Declaration of LLM usage**

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

    Answer: [Yes]

    Justification: The paper describes the use of multimodal foundation models and VLMs as core components of the proposed methodology, including generating the training data and finetuning for improving the image retouch performance.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.

- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.