CHIARORETOUCH: RECOVERABLE COLOR ACTIONS FOR STAGEWISE LOCAL PHOTO RETOUCHING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reasoning photo retouching translates abstract user intent into precise tone and color adjustments. Generative editors regenerate pixels and reach impressive quality at the cost of slow inference and capped output resolution, whereas driving a differentiable executor with compact editing parameters is efficient and resolution-independent but has largely been limited to global operations. Extending such efficient models to local retouching requires explicitly modeling *where* to adjust and *what* action to apply. From a per-pixel perspective, we factorize region control along chromatic, luminance, and semantic dimensions, and build a six-stage process over Hue, three luminance bands (Shadows, Midtones, and Highlights), and Subject supports, together with a Global stage. Learning such structured local execution, however, requires stagewise trajectories with spatial supervision, which are difficult to collect at scale. We therefore treat recoverability as a design constraint and propose ChiaroRetouch, whose fixed-basis Gaussian executor stays nonlinear in color yet linear in its executable action coefficients, so every masked state transition determines its action by a deterministic support-weighted solve. This property lets MaskGrade synthesize and numerically verify local stagewise trajectories at scale via controlled degradation. Experiments show that ChiaroRetouch achieves a favorable trade-off between reference fidelity and instruction adherence across multiple expert retouching benchmarks; its closed-form recovery is over $94\times$ faster than iterative fitting, and it processes a 1K image in 0.35s while supporting local retouching at original resolution.

1 INTRODUCTION

Photo retouching aims to enhance visual aesthetics through precise tone and color adjustments while preserving fine-grained original textures (Bychkovsky et al., 2011; Gharbi et al., 2017; Moran et al., 2021; Liang et al., 2021; Yang et al., 2022; Liu et al., 2023b). Leveraging the visual understanding and reasoning of vision-language models (VLMs), reasoning photo retouching further translates abstract user intent into executable enhancement operations (Dutt et al., 2025; Lin et al., 2025; YeBin et al., 2025; Lin et al., 2026; Wu et al., 2026b; Yao et al., 2026; Ma et al., 2026). Existing generative editing methods achieve impressive retouching quality (Brooks et al., 2023; Huang et al., 2024; Black Forest Labs et al., 2025; Wu et al., 2025; Chang et al., 2026; Qin et al., 2026), but their limited generation resolution and high computational cost hinder direct scaling to arbitrarily high-resolution images. In contrast, the latest reasoning retouching methods (Guo et al., 2026) drive a differentiable executor with compact editing parameters, enabling efficient, end-to-end, and resolution-independent processing without regenerating image content.

However, extending such efficient models to local retouching requires deciding not only *what* color action to apply but also *where* to apply it (Figure 1(a)). A single support–action pair is insufficient when different image regions require distinct, or even conflicting, adjustments. Making a subject stand out while muting its background, for instance, needs two supports carrying opposite adjustments (Figure 1(a)). From a per-pixel perspective, we factorize region control along three complementary cues, instantiated as Hue (*chromatic*), Shadows/Midtones/Highlights (*luminance*), and Subject (*semantic*) supports. Together with a Global stage, they yield a six-stage process with distinct support slots for composing local photometric adjustments.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

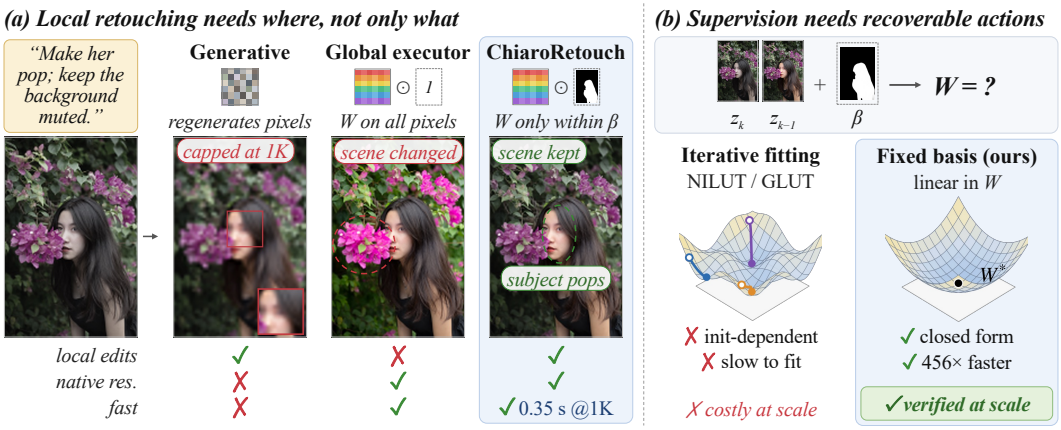


Figure 1: **Local control and recoverable supervision.** Spatial supports localize color actions; a fixed Gaussian basis enables closed-form recovery for trajectory construction.

Learning such structured local execution requires step-by-step trajectories with spatial supports and color actions, which are difficult to collect from experts at scale. Controlled degradation has effectively synthesized global color adjustment data (Park et al., 2018; Dutt et al., 2025; Guo et al., 2026). For local stagewise trajectories, a reference image is degraded stage by stage within known spatial supports. Each stage's supervision target is then the state restored by the executor's reverse color action. Since degradations such as highlight clipping and desaturation discard information, this action cannot be derived analytically and must be recovered numerically from the adjacent states. This recovery is costly: for continuous color representations such as NILUT (Conde et al., 2024) and standard GLUT (Xue et al., 2026), it requires nonlinear iterative optimization, whose cost and sensitivity to initialization limit construction and verification at scale (Figure 1(b)).

To address this issue, we propose ChiaroRetouch. We co-design the executor for recoverability: freezing the basis geometry and blending weights of a Gaussian color executor (Xue et al., 2026) keeps the mapping nonlinear in color yet linear in its action coefficients, so recovering an action from a masked state transition is a deterministic support-weighted linear solve. Building on this recoverable representation, our MaskGrade pipeline constructs local stagewise trajectories at scale via controlled degradation. Supervised by these trajectories, ChiaroRetouch separately predicts spatial supports (*where*) and color actions (*what*) and executes them through the structured six-stage process. Experiments on multiple expert retouching benchmarks (Bychkovsky et al., 2011; Liang et al., 2021; Lin et al., 2026) show that ChiaroRetouch achieves competitive reference fidelity and instruction adherence, with a clear fidelity–adherence trade-off. Its closed-form recovery is over 94× faster than iterative fitting, and the full model processes a 1K image in 0.35 s while retouching locally at the original resolution.

- **Recoverable color actions.** Freezing a learned nonlinear Gaussian basis keeps a flexible color mapping while making the executable action linear in its coefficients, so masked state transitions determine their actions deterministically and can be verified at scale.
- **Recovery-verified local stagewise trajectories.** MaskGrade extends controlled degradation from global color data to local stagewise trajectories, explicitly aligning spatial supports, color actions, and intermediate states through reconstruction and stability checks. On six-step chains, closed-form recovery leaves 75.7% of trajectories free of severe local errors against 54.6% for per-pixel numerical inversion at a comparable mean error.
- **Structured local execution and diagnosis.** ChiaroRetouch instantiates region control as a six-stage factorization over chromatic, luminance, and semantic support families, presented as one practical decomposition rather than a unique one. Trajectory supervision improves intermediate-stage alignment, and interventions show that localization is carried by the spatial supports, whereas color-action prediction dominates current reconstruction error. Replacing the stage markers with a generic word costs 0.76 SC while PSNR does not drop, so reference metrics alone miss this degradation.

2 Related Work

Recoverable color transform representations. Learning-based retouching commonly adopts compact parametric color transforms, including adjustment curves and splines (Moran et al., 2021; Bianco et al., 2020), conditional color mappings (Liu et al., 2023b; Ho & Zhou, 2021), and 3D LUTs (Yang et al., 2022; Zhang et al., 2022; Zhao et al., 2026b). Recent methods further predict discrete LUT tokens (Ma et al., 2026) or model transforms continuously via implicit functions or Gaussian representations (Conde et al., 2024; Xue et al., 2026). While effective for forward rendering, inversely recovering executable actions for these continuous models typically requires costly nonlinear iterative fitting. In contrast, our fixed-basis representation reduces action recovery to a regularized linear problem with a unique global optimum.

Stagewise editing supervision. Existing expert datasets mainly provide input–target image pairs (Bychkovsky et al., 2011; Liang et al., 2021), while controlled degradation enables the automatic synthesis of global color supervision (Dutt et al., 2025; Guo et al., 2026). Sequential editing methods model multi-step execution through reinforcement learning, neural operators, interaction records, or VLM reasoning (Furuta et al., 2020; Wang et al., 2022; Shi et al., 2021; Cheng et al., 2020; Zhao et al., 2026a; Yang et al., 2026). However, such supervision typically does not jointly provide stage-aligned intermediate states, spatial supports, and executable actions. MaskGrade addresses this by extending controlled degradation to explicitly align all three for local stagewise trajectories.

Spatially selective execution. Spatially varying enhancement has been realized through bilateral grids, illumination maps, curves, LUTs, and adaptive filters (Gharbi et al., 2017; Wu et al., 2026a; Wang et al., 2019; Guo et al., 2020; Liu et al., 2023a; Gong et al., 2025; Moran et al., 2020; Ouyang et al., 2023). Meanwhile, agentic and generative methods achieve local control through external masks or region conditioning (Lin et al., 2026; Zheng et al., 2026; Ye et al., 2026; Chang et al., 2026; Qin et al., 2026; Ouyang et al., 2026). In these mechanisms, spatial information serves as internal modulation, a generation condition, or a filter map predicted jointly with its parameters. We instead keep supports and color actions as separate executable quantities, recovered and supervised stage by stage, which makes local trajectories verifiable. We focus on photometric editing, excluding content recomposition.

3 Method

3.1 Overview: Decoupling Local Retouching

ChiaroRetouch factorizes instruction-guided local retouching into semantic intent, spatial supports (*where*), and continuous color actions (*what*). As illustrated in Figure 2, this yields a structured six-stage sequence: **Global**, **Hue** (chromatic), **Shadows**, **Midtones**, **Highlights** (luminance), and **Subject** (semantic). This fixed order provides distinct support slots for local adjustments and mirrors the execution order used at inference.

3.2 Linearized Executor and Closed-Form Recovery

Masked color execution. The editing process spans $K = 6$ stages. Given input Z_K and final output Z_0, the intermediate state update at stage k (indexed in reverse $K \to 1$) applies a color action W_k under spatial mask $M_k \in [0, 1]$:

$$Z_{k-1} = Z_k + M_k \odot [F_{W_k}(Z_k) - Z_k], \tag{1}$$

where $F_{W_k}(\cdot)$ is the RGB-to-RGB color mapping, and $\odot$ denotes element-wise multiplication broadcast across the three color channels. Within a stage, pixels with zero support ($M_k = 0$) stay unchanged; a pixel is preserved along the whole chain only if its support vanishes at every stage. Intermediate states are evaluated without clipping to keep each update affine in its color code; clipping of the executed values is deferred to the final output Z_0.

Fixed-basis Gaussian color executor. To parameterize the color mapping F_W, we adapt the Gaussian LUT (GLUT) (Xue et al., 2026). We freeze the $N = 64$ Gaussian centers, covariances, and

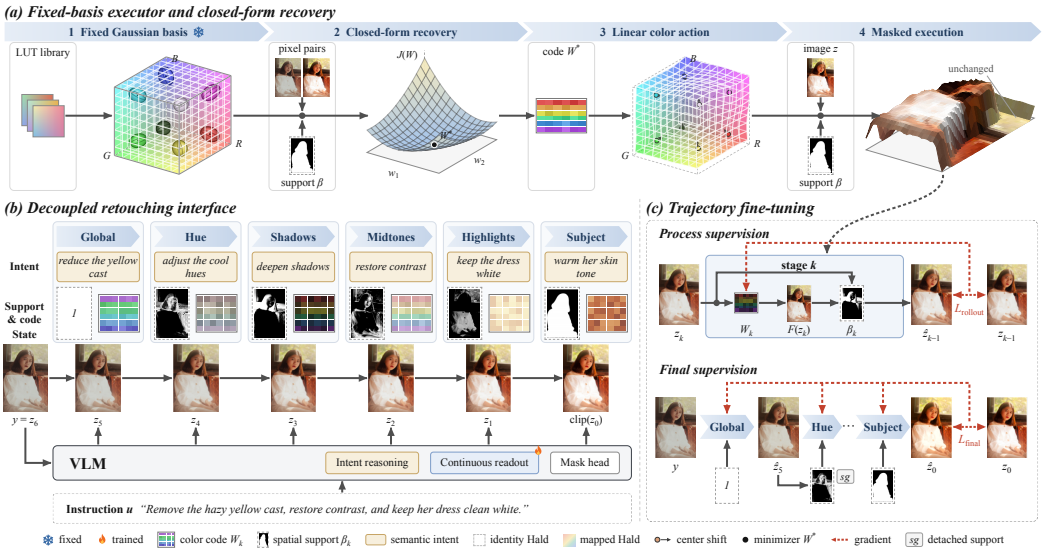


Figure 2: **ChiaroRetouch framework.** The VLM separates intent, spatial support, and continuous color codes, which are executed stage by stage.

opacities after pre-training them on a LUT library (color bank; Appendix F). For any RGB color $C \in \mathbb{R}^3$, these fixed parameters yield a normalized Gaussian response vector $w(C) \in \mathbb{R}^{64}$. The expanded color feature vector $\phi(C) \in \mathbb{R}^{260}$ is constructed as:

$$\phi(C) = \big[w(C) \otimes C,\ w(C),\ C,\ 1\big], \qquad F_W(C) = \phi(C)W^{\mathsf{T}}. \tag{2}$$

The action code $W \in \mathbb{R}^{3\times 260}$ stores channel coefficients, making $F_W(C)$ strictly linear in W despite the non-linear $\phi(C)$.

Regularized closed-form recovery. Let Z and Z^{target} be the flattened pixel matrices of the input and target states for a single transition, and let M be the flattened spatial support. By embedding the mask into a diagonal matrix $B = \mathrm{diag}(M)$ and defining the feature matrix Φ (where each row is $\phi(Z)$), we can isolate W from Equation (1).

Let $H = B\Phi$ be the mask-weighted feature matrix and $Y = (Z^{\text{target}} - Z) + BZ$ be the adjusted fitting target. With a regularization strength $\lambda > 0$, the optimal code W^* minimizes the Ridge regression objective:

$$\begin{aligned} W^* &= \arg\min_W \big\|HW^{\mathsf{T}} - Y\big\|_F^2 + \lambda\|W\|_F^2, \\ (W^*)^{\mathsf{T}} &= \big(H^{\mathsf{T}}H + \lambda I\big)^{-1} H^{\mathsf{T}}Y. \end{aligned} \tag{3}$$

This formulation yields a unique closed-form solution via a direct linear-system solve. Absorbing the support B into the system matrix also handles zero or near-zero mask values without dividing observed residuals by spatial weights. More details on recovery algebra are provided in Appendix A. We retain edits passing strict thresholds for reconstruction error and relative split-half discrepancy, both defined in Appendix A.5.

3.3 MaskGrade and Process Supervision

Controlled local degradation. To synthesize trajectories, we iteratively degrade a reference Z_0 in reverse inference order ($k = 1, \ldots, K$). Using randomized pixel-wise mappings L_k and strength-scaled masks M_k, the state degrades via:

$$Z_k = Z_{k-1} + M_k \odot \left[L_k(Z_{k-1}) - Z_{k-1}\right]. \tag{4}$$

The final degraded state Z_K serves as the unretouched input image.

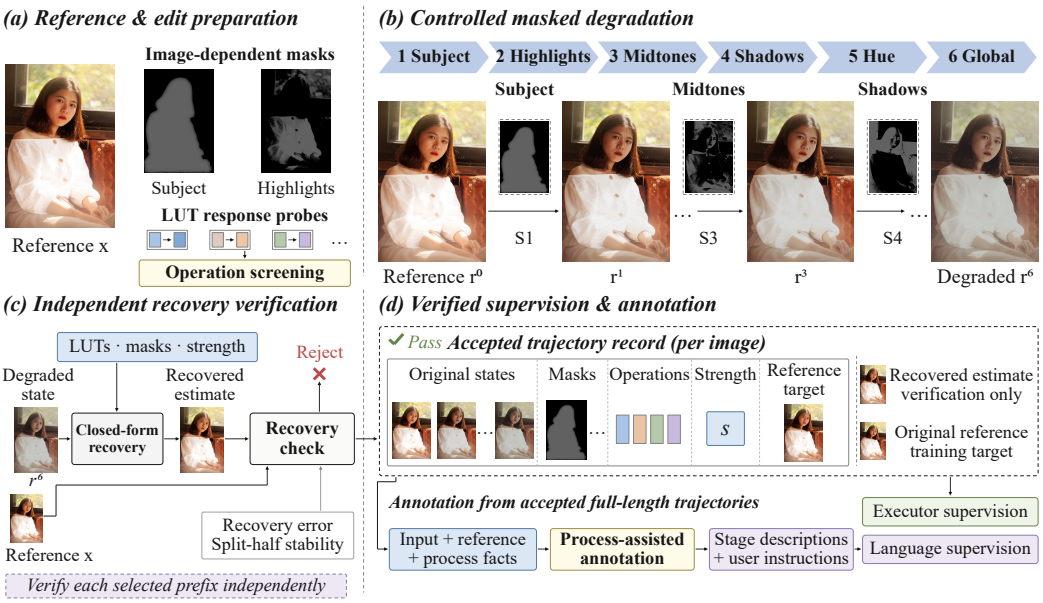


Figure 3: **MaskGrade construction.** Masked degradations of a reference yield recorded states; recovery checks and annotation turn them into supervision.

Action recovery and verification. To construct the forward restoration trajectory, we reverse the sequence. For each inference stage k (traversed $k = K, \ldots, 1$), the input state is Z_k, the target state is Z_{k-1}, and the spatial support is M_k. We apply the closed-form solver (Equation 3) to these known states to recover the optimal continuous color action W_k^* that restores the degraded state.

Since degradations like clipping discard information, we filter recovered actions using the aforementioned verification metrics to ensure high-fidelity alignment. We retain only the trajectories where every intermediate action passes both checks, explicitly aligning spatial supports, color actions, and image states with high fidelity.

Language grounding and endpoint pairs. For each verified trajectory, an annotation model processes the image states alongside the recorded degradation metadata (support types, adjustment directions, and strengths) to generate stage descriptions grounded in the constructed edits. Each record thereby pairs the sequence of executable color codes with an intent-driven language instruction.

Additionally, MaskGrade incorporates standard endpoint-pair supervision to populate the color bank. For style examples (LUT pairs) and real expert pairs, the intermediate steps are bypassed. We set the global target Z^{target} directly to the reference image and recover a single global code ($M_k \equiv 1$). We assign real-pair prototype labels by minimizing the rendered image error, using the original expert image as the pixel-space training target.

3.4 Continuous Readout and Training Objectives

Continuous VLM readout. We append $m = 8$ learnable readout tokens per stage to the VLM's input. The tokens follow the instruction and the stage markers, and no textual stage description is generated at inference.

A linear projection head maps the concatenated hidden states of these readout tokens directly to the predicted continuous color code $\widehat{W}_k \in \mathbb{R}^{3\times 260}$ for each stage k. To ensure stable optimization, the regression targets are standardized using fixed dataset statistics (detailed in Appendix F). During inference, the VLM reads out the six stage codes $\widehat{W}_1, \ldots, \widehat{W}_6$ in a single forward pass, and the executor applies them sequentially.

Training objectives. For standard endpoint-pair data (such as LUT styles), we bypass the intermediate stages and perform a single global edit ($M \equiv 1$). Given the input Z and the predicted global

code $\widehat{W}$, the single-edit loss minimizes the discrepancy with the reference target Z^{target}:

$$\mathcal{L}_{\text{single}} = \left\| \widehat{Z}^{\text{target}} - Z^{\text{target}} \right\|_1, \tag{5}$$

where $\widehat{Z}^{\text{target}}$ is rendered using Equation (1).

For our synthesized local trajectories, we supervise the sequential six-stage rollout. Let Z_k be the recorded ground-truth state, and $\widehat{Z}_k$ be its predicted rollout counterpart, initialized with $\widehat{Z}_K = Z_K$. For $k = K, \ldots, 1$, the model executes its own sequential predictions:

$$\widehat{Z}_{k-1} = \widehat{Z}_k + M_k(\widehat{Z}_k) \odot \left[F_{\widehat{W}_k}(\widehat{Z}_k) - \widehat{Z}_k \right]. \tag{6}$$

The photometric supports are recomputed from the current predicted state $\widehat{Z}_k$ and detached, so gradients flow only through the predicted color actions $\widehat{W}_k$.

We evaluate two trajectory objectives on this rollout:

$$\mathcal{L}_{\text{final}} = \left\| \widehat{Z}_0 - Z_0 \right\|_1, \qquad \mathcal{L}_{\text{rollout}} = \frac{1}{K} \sum_{k=1}^{K} \left\| \widehat{Z}_{k-1} - Z_{k-1} \right\|_1. \tag{7}$$

$\mathcal{L}_{\text{final}}$ supervises only the final output, whereas $\mathcal{L}_{\text{rollout}}$ (our default) aligns every predicted intermediate state with its recorded counterpart. Objective comparisons and rollout diagnostics are reported in Section 4.4.

3.5 Inference

Given an input image Z_K and a language instruction, inference executes two forward passes of the shared VLM using branch-specific adapters. The *spatial pass* predicts the Subject support mask using a language-conditioned mask head. The *color pass* appends the six stage markers and their readout tokens to the instruction, predicting the continuous color actions $\widehat{W}_1, \ldots, \widehat{W}_6$ in a single forward pass without generating textual stage descriptions. Starting from the input Z_K, the executor applies the predicted codes in the fixed descending order ($k = K, \ldots, 1$) using Equation (1). During execution, the Global mask is 1, Hue and Luminance masks are computed analytically from Z_k, and the Subject mask comes from the spatial pass. Crucially, because the closed-form execution is fully decoupled from the VLM's spatial grid, rendering runs natively at the original image resolution (detailed in Appendix J).

4 Experiments

4.1 Experimental Setup

Model. The proposed model uses Qwen3-VL-4B (Qwen Team, 2025) with a frozen visual tower and rollout-stage supervision (Eq. (7); Appendix F). *1-stage* and *6-stage* execute the same checkpoint with one full-image stage and with all six stages (Section 3.5).

Benchmarks and metrics. ArtEdit-Bench-Lr (Lin et al., 2026) provides 400 English editing requests. For MIT-Adobe FiveK (Bychkovsky et al., 2011) (Expert C, 498 pairs) and PPR10K (Liang et al., 2021) (Expert a, 492 pairs), GPT-5.6 Terra writes an editing instruction for each expert pair. All methods receive the same inputs and instructions; references are used only for annotation and scoring (Appendix G.1). We report reference fidelity, VIEScore (SC: instruction adherence; PQ: content consistency; O: their geometric mean), and no-reference quality (Appendix G).

4.2 Main Results

Reference fidelity. Ours (1-stage) leads on all five ArtEdit reference metrics, while 6-stage execution leads on nine of ten expert-benchmark metrics (Tables 1–2).

User study. Ten participants rated 100 ArtEdit images in a blinded five-way comparison, ten ratings per image (Figure 4). Six-stage execution raises the overall share from 9.3% to 23.3% and ranks second in visual quality, behind EyeControl, which can also alter content.

Table 1: **Comparison on ArtEdit-Bench.** Backbone is the principal model size. Bold/underline: best/second.

Method	Backbone (B)	Reference fidelity L1↓ (10^2)	L2↓ (10^3)	PSNR↑ (dB)	SSIM↑	ΔE_{00} ↓	Conditional↑ SC (0–10)	PQ	O	No-reference↑ ArtiMuse (0–100)	Q-Align (1–5)	DeQA (1–5)
Generative editing methods												
EyeControl	12	10.44	23.54	17.96	0.664	11.77	<u>8.18</u>	<u>9.87</u>	**8.96**	**66.35**	**3.77**	**3.94**
FLUX.1 Kontext	12	17.90	67.81	14.09	0.443	18.51	7.92	9.31	8.54	63.59	3.58	3.40
Qwen-Image-Edit	20	14.12	41.57	15.45	0.534	15.85	**8.60**	8.97	8.75	63.71	<u>3.68</u>	3.68
InstructPix2Pix	0.86	13.22	36.81	15.90	0.557	15.73	6.77	7.98	7.24	60.48	3.37	3.46
Retouching and agent-based methods												
JarvisEvo	8	<u>8.44</u>	16.05	<u>20.73</u>	<u>0.789</u>	<u>9.76</u>	7.73	**9.88**	8.68	<u>65.24</u>	3.60	3.83
JarvisArt	7	9.23	18.75	19.92	0.754	10.81	7.80	9.74	8.65	<u>65.24</u>	3.62	3.71
VeraRetouch	0.5	9.52	19.07	19.43	0.777	11.24	7.03	9.84	8.23	64.48	3.47	3.79
MonetGPT	7	10.65	21.93	18.69	0.764	12.49	6.31	9.83	7.69	63.92	3.49	3.92
Ours (1-stage)	4	**7.98**	**13.43**	**20.88**	**0.804**	**9.62**	7.37	9.82	8.43	64.58	3.52	<u>3.93</u>
Ours (6-stage)	4	8.54	<u>14.25</u>	20.27	0.786	10.01	<u>8.18</u>	9.73	<u>8.89</u>	64.81	3.47	3.89

1-stage versus 6-stage execution. 1-stage execution achieves higher reference fidelity, whereas 6-stage execution trades a 0.61 dB PSNR decrease for better instruction adherence, raising SC from 7.37 to 8.18 and O from 8.43 to 8.89 on ArtEdit.

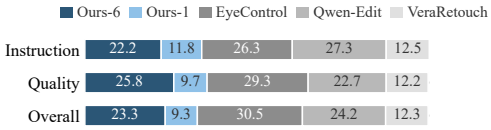


Figure 4: **Human preference.** Share of votes (%) per criterion in a blinded evaluation.

Qualitative results. In Figure 5, the baselines either leave the background nearly unchanged (JarvisEvo, MonetGPT) or darken the subject with it (VeraRetouch). Ours (6-stage) darkens the background while keeping the red flower and the warm skin tones, since each local adjustment is confined to its spatial support.

Table 2: **Comparison on FiveK and PPR10K.** L1 and L2 are scaled by 10^2 and 10^3.

Method	FiveK-Bench L1↓	L2↓	PSNR↑	SSIM↑	ΔE_{00} ↓	PPR10K-Bench L1↓	L2↓	PSNR↑	SSIM↑	ΔE_{00} ↓
MonetGPT	9.86	16.57	19.52	**0.805**	10.42	<u>7.57</u>	<u>10.99</u>	<u>21.94</u>	<u>0.898</u>	<u>8.07</u>
JarvisArt	9.90	17.62	19.28	0.774	11.70	9.76	16.24	19.49	0.835	10.11
JarvisEvo	<u>8.80</u>	<u>13.82</u>	<u>20.25</u>	0.801	<u>10.00</u>	9.12	13.74	20.12	0.851	8.85
VeraRetouch	9.58	16.17	19.95	0.803	10.57	7.80	11.14	21.44	0.893	8.53
Ours (6-stage)	**8.32**	**11.14**	**20.66**	<u>0.804</u>	**9.30**	**6.10**	**6.53**	**23.22**	**0.922**	**6.80**

4.3 Efficiency

Closed-form recovery. Closed-form recovery takes 36.6 ms per pair versus 3.4 s for L-BFGS (94–2,649× faster than the iterative fitters), with similar reconstruction error and no initialization requirement (Figure 6; Table C2).

Inference. Ours-6 edits a 1K image in 0.35 s, 19–350× faster than the baselines, since one VLM pass reads out all codes; the margin narrows to 4.2× at 4K, where rendering dominates (Figure 7).

4.4 Ablation Studies and Analysis

We ablate the supervision objective (Table 3) and probe the trained model with test-time interventions and oracle replacement (Figure 8).

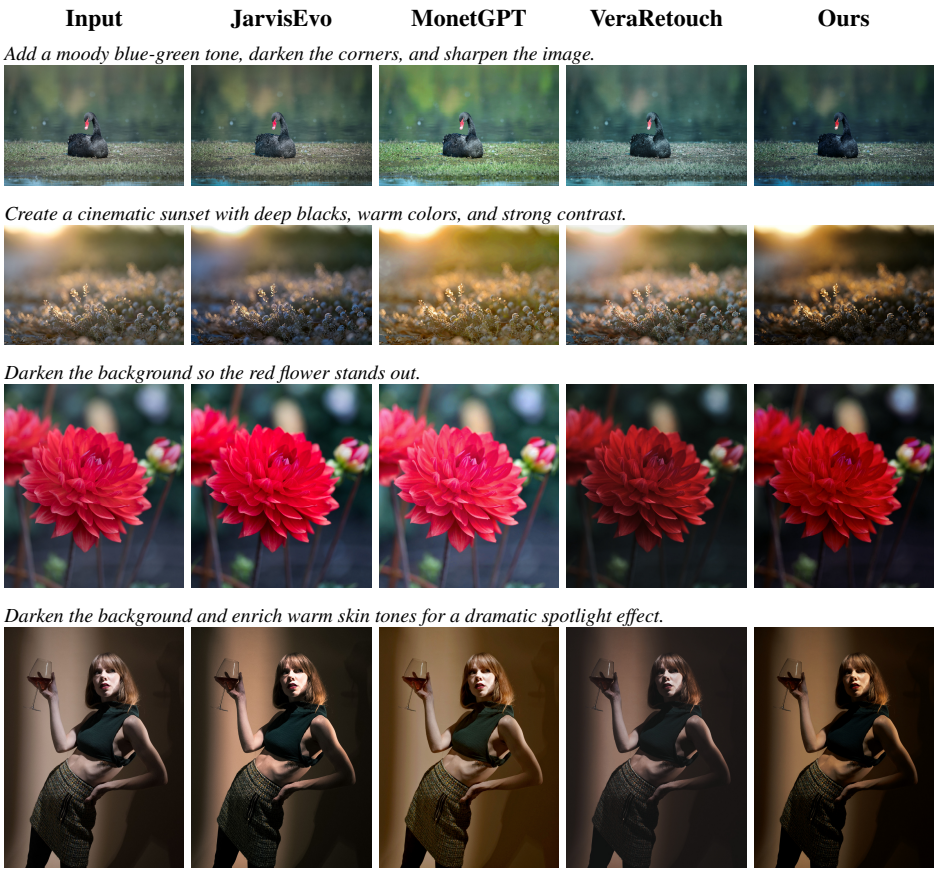


Figure 5: **Qualitative comparison on ArtEdit-Bench.** Ours shows six-stage outputs.

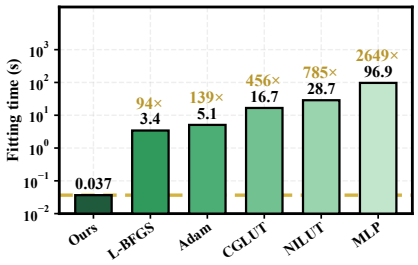


Figure 6: **Color recovery time.** Median per ArtEdit pair (log scale).

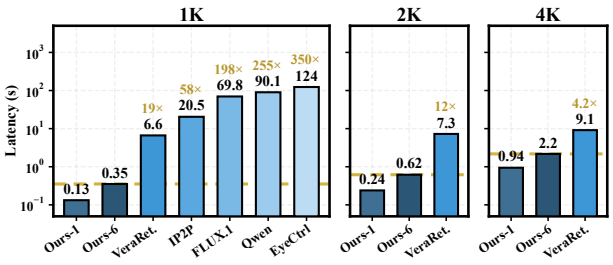


Figure 7: **Inference latency.** Per image (log scale); methods editing through external software are excluded.

Table 3: **Ablation on supervision objectives.** A: ArtEdit; B: MaskGrade-400 (drift in 0–255). Δ: change from Global endpoint (blue: better; green: worse), whose drift is zero by construction.

	A. ArtEdit-Bench				B. MaskGrade-400	
Supervision	SC↑	PQ↑	O↑	PSNR↑	Align.↑	Drift↓
Global endpoint	7.072	9.875	8.247	20.458	0.076	0.000
Final only	8.019	9.694	8.788	20.381	0.250	0.369
Δ	**+0.95**	−0.18	**+0.54**	−0.08	**+0.174**	+0.369
Stage + final	8.144	9.667	8.845	19.991	0.272	0.476
Δ	**+1.07**	−0.21	**+0.60**	−0.47	**+0.196**	+0.476
Rollout-stage (Ours)	**8.181**	9.732	**8.89**	20.27	**0.281**	0.497
Δ	**+1.11**	−0.14	**+0.64**	−0.19	**+0.205**	+0.497
No-op	1.443	9.895	3.186	19.741	0.000	0.000

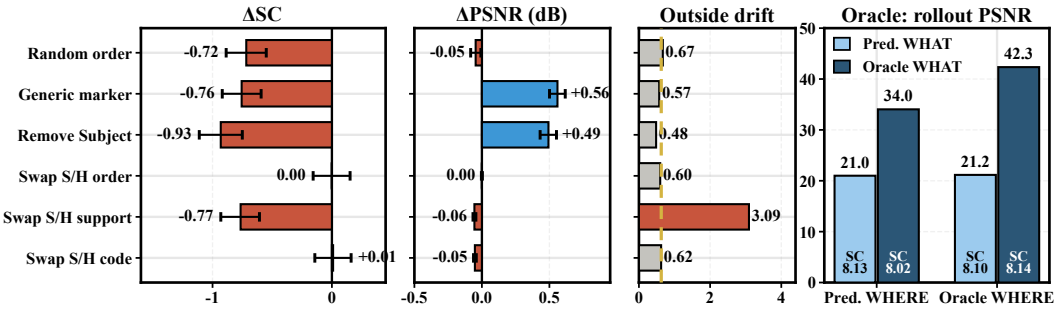


Figure 8: **Ablation by test-time interventions and oracle replacement.** Paired changes from normal execution with 95% intervals (red/blue: significant decrease/increase; gray: not significant); dashed line: normal drift. S/H: Shadows/Highlights.

Trajectory supervision improves intermediate-stage alignment. Global endpoint differs from the structured variants in both execution and supervision, so its SC gap of 0.95–1.11 is not the effect of the intermediate-state loss alone (Table 3; definitions and schedules in Appendix F). Within structured execution, rollout-stage supervision raises stage alignment over Final only from 0.250 to 0.281 (+0.032 paired, 95% CI [0.023, 0.040]), concentrated on the local stages (Figure 9), and SC by 0.16. Endpoint fidelity moves the other way, with ArtEdit PSNR down by up to 0.47 dB, so dense trajectory supervision shows up mainly in intermediate-state alignment.

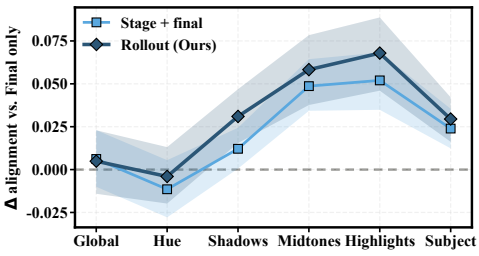


Figure 9: **Per-stage alignment gain over Final only.** Paired differences on MaskGrade-400 with 95% intervals.

The model uses the stage markers. Replacing the stage markers with a generic word lowers SC by 0.76 and permuting the execution order of the predicted codes lowers SC by 0.72, while neither lowers PSNR (-0.05 to $+0.56$ dB; Figure 8; Appendix C.2), so reference metrics alone miss this degradation. Swapping only the Shadows/Highlights order leaves SC and drift unchanged, so the loss comes from non-commuting stages, not absolute position.

Stage roles live in the supports, not in the codes. Exchanging the Shadows/Highlights supports raises outside drift about fivefold and lowers SC by 0.77, whereas exchanging their codes changes neither: a code has no stage semantics on its own, since its effect is set jointly by the current state and its support (Appendix A.4). The factors interact: oracle codes add 13.0 dB of rollout PSNR and oracle supports 0.16 dB, but under oracle codes the supports add a further 8.3 dB (34.0 to 42.3 dB), so color-action prediction is the current bottleneck rather than localization being solved.

5 Conclusion

ChiaroRetouch treats recoverability as a design constraint for scalable process supervision in local retouching. Freezing the executor's geometry and blending weights determines each color action deterministically from a masked state transition, letting MaskGrade construct and verify structured local trajectories. With a 4B backbone, single-stage execution leads every ArtEdit reference metric and six-stage execution nine of ten expert-benchmark metrics, trading fidelity for instruction adherence. Interventions show that localization is carried by the spatial supports, which provide stage-local containment, whereas reconstruction error is dominated by color-action prediction. The method is limited to photometric edits and depends on recoverable color operations and mask quality; content-aware and geometric edits are future work. Instruction scores also saturate, since oracle color actions add 13.0 dB of rollout PSNR without raising SC, so process-level measures are needed to judge local editing accuracy. The chromatic and luminance supports also remain fixed analytic families rather than learned ones. We view the six-stage factorization as one practical instantiation, not a unique decomposition.

AI Use Statement

In preparing this paper, we used generative AI tools for LaTeX template configuration, title suggestions, assistance in drafting and revising the full manuscript, including the abstract, introduction, method, experiment, and appendix sections, tables, and figure captions, translation from Chinese into English, assistance with implementing the data-construction, training, and evaluation code, assistance with designing experiments and providing feedback on them, assistance with interpreting experimental results, reference lookup and formatting, and assistance with scientific-figure design and editable-diagram preparation. We also used GPT-5.6 Terra to annotate the stage descriptions and editing instructions of the MaskGrade training trajectories and the MaskGrade-400 diagnostic set, and to annotate editing instructions from before/after expert-retouching pairs for the FiveK and PPR10K evaluations. Gemini 3.1 Pro serves as the VIEScore judge for the reported instruction-conditioned scores.

Ethics Statement

We have read the ICLR Code of Ethics and adhere to it. The preference study asked volunteer participants to compare anonymized outputs and recorded only the selected output; no personal, demographic, or biometric data were collected or released. All photographs come from public sources—MIT-Adobe FiveK (Bychkovsky et al., 2011), PPR10K (Liang et al., 2021), ArtEdit-Bench-Lr (Lin et al., 2026), and Unsplash (Unsplash, 2026)—used for non-commercial research under their terms, with the native-resolution photographer credited in Appendix J; some images show identifiable people, which is inherent to portrait retouching, and we neither collect new images of people nor infer any attribute about them. The method applies masked per-pixel color maps and cannot add, remove, or geometrically alter content, which bounds misuse to tonal and color manipulation. Even so, localized skin-tone edits can encode narrow aesthetic norms, the annotation and judge models inherit the biases of their vision-language backbones, retouching quality may vary across unevenly represented skin tones and subjects, and altered photographs should not be presented as unmodified records.

Reproducibility Statement

The executor, its closed-form recovery, and the training objectives are specified in Sections 3.2–3.4 and Appendix A; the MaskGrade pipeline with its verification metrics and annotation prompts in Appendices A.5, D, and H; the spatial branch in Appendix B; every training stage with its data, schedules, and supervision variants in Appendix F; and the baselines, metrics, latency protocol, and instruction annotation in Appendix G. Reported intervals are 95% bootstrap intervals over 2,000 resamples of source-image groups (Appendix C), and every evaluation split is held out from training by the image-level split in Appendix G.1. Institutional restrictions prevent releasing source code, trained weights, or the constructed trajectories, so these specifications are reported in full, down to data counts, hyperparameters, verification thresholds, and prompts, so that the work can be reimplemented from the paper.

References

Simone Bianco, Claudio Cusano, Flavio Piccoli, and Raimondo Schettini. Personalized image enhancement using neural spline color transforms. *IEEE Trans. Image Process.*, 2020. 3

Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, Sumith Kulal, Kyle Lacey, Yam Levi, Cheng Li, Dominik Lorenz, Jonas Müller, Dustin Podell, Robin Rombach, Harry Saini, Axel Sauer, and Luke Smith. FLUX.1 Kontext: Flow matching for in-context image generation and editing in latent space. *arXiv*, 2025. 1, 32

Tim Brooks, Aleksander Holynski, and Alexei A. Efros. InstructPix2Pix: Learning to follow image editing instructions. In *CVPR*, 2023. 1, 32

Vladimir Bychkovsky, Sylvain Paris, Eric Chan, and Frédo Durand. Learning photographic global tonal adjustment with a database of input / output image pairs. In *CVPR*, 2011. 1, 2, 3, 6, 10

Shuo Cao, Nan Ma, Jiayang Li, Xiaohui Li, Lihao Shao, Kaiwen Zhu, Yu Zhou, Yuandong Pu, Jiarui Wu, Jiaquan Wang, Bo Qu, Wenhai Wang, Yu Qiao, Dajuin Yao, and Yihao Liu. ArtiMuse: Fine-grained image aesthetics assessment with joint scoring and expert-level understanding. In *CVPR*, pp. 15313–15322, 2026. 32

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris Coll-Vinent, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, Jie Lei, Tengyu Ma, Baishan Guo, Arpit Kalla, Markus Marks, Joseph Greer, Meng Wang, Peize Sun, Roman Rädle, Triantafyllos Afouras, Effrosyni Mavroudi, Katherine Xu, Tsung-Han Wu, Yu Zhou, Liliane Momeni, Rishi Hazra, Shuangrui Ding, Sagar Vaze, Francois Porcher, Feng Li, Siyuan Li, Aishwarya Kamath, Ho Kei Cheng, Piotr Dollár, Nikhila Ravi, Kate Saenko, Pengchuan Zhang, and Christoph Feichtenhofer. SAM 3: Segment anything with concepts. In *ICLR*, pp. 138846–138923, 2026. 21

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 *AAAI*, 2026. 1, 3

Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *CVPR*, pp. 1290–1299, 2022. 18

Yu Cheng, Zhe Gan, Yitong Li, Jingjing Liu, and Jianfeng Gao. Sequential attention gan for interactive image editing. In *ACM MM*, 2020. 3

Marcos V. Conde, Javier Vazquez-Corral, Michael S. Brown, and Radu Timofte. NILUT: Conditional neural implicit 3d lookup tables for image enhancement. In *AAAI*, 2024. 2, 3

Niladri Shekhar Dutt, Duygu Ceylan, and Niloy J. Mitra. MonetGPT: Solving puzzles enhances mllms' image retouching skills. *ACM Trans. Graph.*, 2025. 1, 2, 3, 16, 32

Ryosuke Furuta, Naoto Inoue, and Toshihiko Yamasaki. PixelRL: Fully convolutional network with reinforcement learning for image processing. *IEEE Trans. Multimedia*, 2020. 3

Michaël Gharbi, Jiawen Chen, Jonathan T. Barron, Samuel W. Hasinoff, and Frédo Durand. Deep bilateral learning for real-time image enhancement. *ACM Trans. Graph.*, 2017. 1, 3

Zerui Gong, Zhonghua Wu, Qingyi Tao, Qinyue Li, and Chen Change Loy. SA-LUT: Spatial adaptive 4d look-up table for photorealistic style transfer. In *ICCV*, 2025. 3

Chunle Guo, Chongyi Li, Jichang Guo, Chen Change Loy, Junhui Hou, Sam Kwong, and Runmin Cong. Zero-reference deep curve estimation for low-light image enhancement. In *CVPR*, 2020. 3

Yihong Guo, Youwei Lyu, Jiajun Tang, Yizhuo Zhou, Hongliang Wang, Jinwei Chen, Changqing Zou, and Qingnan Fan. VeraRetouch: A lightweight fully differentiable framework for multi-task reasoning photo retouching. In *SIGGRAPH*, 2026. 1, 2, 3, 16, 32

Man M. Ho and Jinjia Zhou. Deep preset: Blending and retouching photos with color style transfer. In *WACV*, 2021. 3

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022. 18, 31

Yuzhou Huang, Liangbin Xie, Xintao Wang, Ziyang Yuan, Xiaodong Cun, Yixiao Ge, Jiantao Zhou, Chao Dong, Rui Huang, Ruimao Zhang, and Ying Shan. SmartEdit: Exploring complex instruction-based image editing with multimodal large language models. In *CVPR*, 2024. 1

Max Ku, Dongfu Jiang, Cong Wei, Xiang Yue, and Wenhu Chen. VIEScore: Towards explainable metrics for conditional image synthesis evaluation. In *ACL*, 2024. 32

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 *CVPR*, 2021. 1, 2, 3, 6, 10

Yunlong Lin, Zixu Lin, Kunjie Lin, Jinbin Bai, Panwang Pan, Chenxin Li, Haoyu Chen, Zhongdao Wang, Xinghao Ding, Wenbo Li, and Shuicheng Yan. JarvisArt: Liberating human artistic creativity via an intelligent photo retouching agent. In *NeurIPS*, 2025. 1, 32

Yunlong Lin, Linqing Wang, Kunjie Lin, Zixu Lin, Kaixiong Gong, Wenbo Li, Bin Lin, Zhenxi Li, Shiyi Zhang, Yuyang Peng, Wenxun Dai, Xinghao Ding, Chunyu Wang, and Qinglin Lu. JarvisEvo: Towards a self-evolving photo editing agent with synergistic editor-evaluator optimization. In *CVPR*, 2026. 1, 2, 3, 6, 10, 32

Chengxu Liu, Huan Yang, Jianlong Fu, and Xueming Qian. 4d lut: Learnable context-aware 4d lookup table for image enhancement. *IEEE Trans. Image Process.*, 2023a. 3

Yihao Liu, Jingwen He, Xiangyu Chen, Zhengwen Zhang, Hengyuan Zhao, Chao Dong, and Yu Qiao. Very lightweight photo retouching network with conditional sequential modulation. *IEEE Trans. Multimedia*, 2023b. 1, 3

Tianren Ma, Mingxiang Liao, Xijin Zhang, and Qixiang Ye. AceTone: Bridging words and colors for conditional image grading. In *CVPR*, 2026. 1, 3

Sean Moran, Pierre Marza, Steven McDonagh, Sarah Parisot, and Gregory Slabaugh. DeepLPF: Deep local parametric filters for image enhancement. In *CVPR*, 2020. 3

Sean Moran, Steven McDonagh, and Gregory Slabaugh. CURL: Neural curve layers for global image enhancement. In *ICPR*, 2021. 1, 3

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 *ICCV*, 2023. 3

Zhuohan Ouyang, Zhe Qian, Wenhuo Cui, and Chaoqun Wang. Region-constrained group relative policy optimization for flow-based image editing. *arXiv*, 2026. 3

Jongchan Park, Joon-Young Lee, Donggeun Yoo, and In So Kweon. Distort-and-recover: Color enhancement using deep reinforcement learning. In *CVPR*, 2018. 2

Chujie Qin, Zilong Zhang, Zewei Chang, Chun-Le 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 *ECCV*, 2026. 1, 3, 32

Qwen Team. Qwen3-VL technical report. *arXiv*, 2025. 6

Gaurav Sharma, Wencheng Wu, and Edul N. Dalal. The CIEDE2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations. *Color Res. Appl.*, 2005. 32

Jing Shi, Ning Xu, Yihang Xu, Trung Bui, Franck Dernoncourt, and Chenliang Xu. Learning by planning: Language-guided global image editing. In *CVPR*, 2021. 3

Unsplash. Unsplash: Freely usable images. https://unsplash.com, 2026. 10, 35

Ruixing Wang, Qing Zhang, Chi-Wing Fu, Xiaoyong Shen, Wei-Shi Zheng, and Jiaya Jia. Underexposed photo enhancement using deep illumination estimation. In *CVPR*, 2019. 3

Yili Wang, Xin Li, Kun Xu, Dongliang He, Qi Zhang, Fu Li, and Errui Ding. Neural color operators for sequential image retouching. In *ECCV*, 2022. 3

Zhou Wang and Alan C. Bovik. Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE Signal Process. Mag.*, 2009. 32

Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Trans. Image Process.*, 2004. 32

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, Yuxiang Chen, Zecheng Tang, Zekai Zhang, Zhengyi Wang, An Yang, Bowen Yu, Chen Cheng, Dayiheng Liu, Deqing Li, Hang Zhang, Hao Meng, Hu Wei, Jingyuan Ni, Kai Chen, Kuan Cao, Liang Peng, Lin Qu, Minggang Wu, Peng Wang, Shuting Yu, Tingkun Wen, Wensen Feng, Xiaoxiao Xu, Yi Wang, Yichang Zhang, Yongqiang Zhu, Yujia Wu, Yuxuan Cai, and Zenan Liu. Qwen-Image technical report. *arXiv*, 2025. 1, 32

Haoning Wu, Zicheng Zhang, Weixia Zhang, Chaofeng Chen, Liang Liao, Chunyi Li, Yixuan Gao, Annan Wang, Erli Zhang, Wenxiu Sun, Qiong Yan, Xiongkuo Min, Guangtao Zhai, and Weisi Lin. Q-align: Teaching lmms for visual scoring via discrete text-defined levels. In *ICML*, 2024. 32

Jiarui Wu, Yujin Wang, Ruikang Li, Fan Zhang, Mingde Yao, and Tianfan Xue. InstantRetouch: Efficient and high-fidelity instruction-guided image retouching with bilateral space. In *CVPR*, 2026a. 3, 32

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. In *CVPR*, pp. 12279–12288, 2026b. 1

Danna Xue, David Serrano-Lozano, Shaolin Su, and Javier Vazquez-Corral. GLUT: 3d gaussian lookup table for continuous color transformation. *arXiv*, 2026. 2, 3, 14

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 *CVPR*, 2022. 1, 3

Hanqing Yang, Qiang Zhou, Yongchao Du, Sashuai Zhou, Zhibin Wang, Jun Song, Tiezheng Ge, Cheng Yu, and Bo Zheng. DDA-Thinker: Decoupled dual-atomic reinforcement learning for reasoning-driven image editing. *arXiv*, 2026. 3

Mingde Yao, Zhiyuan You, King-Man Tam, Menglu Wang, and Tianfan Xue. PhotoAgent: Exploratory visual aesthetic planning with large vision models. In *ICML*, 2026. 1

Ruijie Ye, Jiayi Zhang, Zhuoxin Liu, Zihao Zhu, Siyuan Yang, Li Li, Tianfu Fu, Franck Dernoncourt, Yue Zhao, Jiacheng Zhu, Ryan Rossi, Wenhao Chai, and Zhengzhong Tu. Agent banana: High-fidelity image editing with agentic thinking and tooling. *arXiv*, 2026. 3

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*, 2025. 1

Zhiyuan You, Xin Cai, Jinjin Gu, Tianfan Xue, and Chao Dong. Teaching large language models to regress accurate image quality scores using score distribution. In *CVPR*, 2025. 32

Fengyi Zhang, Hui Zeng, Tianjun Zhang, and Lin Zhang. CLUT-Net: Learning adaptively compressed representations of 3dluts for lightweight image enhancement. In *ACM MM*, 2022. 3

Yiran Zhao, Yaoqi Ye, Xiang Liu, Michael Qizhe Shieh, and Trung Bui. ImageEdit-R1: Boosting multi-agent image editing via reinforcement learning. *arXiv*, 2026a. 3

Ziqi Zhao, Abhijit Mishra, and Shounak Roychowdhury. LoR-LUT: Learning compact 3d lookup tables via low-rank residuals. *arXiv*, 2026b. 3

Huayu Zheng, Guangzhao Li, Baixuan Zhao, Siqi Luo, Hantao Jiang, Guangtao Zhai, and Xiaohong Liu. A^2-edit: Precise reference-guided image editing of arbitrary objects and ambiguous masks. *arXiv*, 2026. 3

A Method Derivations and Implementation Details

This appendix derives the fixed-basis color representation and its support-weighted closed-form recovery, states their properties, and compares closed-form recovery with numerical inversion on degradation chains. Training settings and evaluation metrics are given in Appendices F and G.

A.1 Fixed-Basis Color Representation

We adopt GLUT's combination of Gaussian-weighted local affine transforms and a global affine transform in RGB space (Xue et al., 2026), and fix the Gaussian centers, covariances, and opacities after learning them on a LUT library (Appendix F). Throughout this derivation, colors are row vectors $c \in \mathbb{R}^{1\times 3}$. Let m_i, Σ_i, and o_i denote the center, positive-definite covariance, and effective positive opacity of Gaussian i. Its log response and normalized weight are

$$\begin{aligned} g_i(c) &= \log o_i - \frac{1}{2}\left[(c-m_i)\Sigma_i^{-1}(c-m_i)^\mathsf{T} + \log\det\Sigma_i\right], \\ w_i(c) &= \frac{\exp g_i(c)}{\sum_{j=1}^{N}\exp g_j(c)}. \end{aligned} \tag{A.1}$$

The implementation evaluates these weights with softmax, giving $\sum_i w_i(c) = 1$. With these parameters fixed, the color transformation becomes

$$F_W(c) = \sum_{i=1}^{N} w_i(c)(cA_i + b_i) + cA_g + b_g = \phi(c)W^\mathsf{T}, \tag{A.2}$$

where $A_i, A_g \in \mathbb{R}^{3\times 3}$, $b_i, b_g \in \mathbb{R}^{1\times 3}$, and

$$\phi(c) = \big[w(c)\otimes c,\ w(c),\ c,\ 1\big], \qquad W \in \mathbb{R}^{3\times D}, \quad D = 4N+4. \tag{A.3}$$

We use $N = 64$, so each color code is a 3×260 matrix. The Gaussian responses vary nonlinearly with the input color, while the output remains linear in W. Gaussian weights determine which colors are affected by each affine component; spatial support $\beta(p)$ determines the image locations at which the color transformation acts.

A.2 Support-Weighted Closed-Form Recovery

Recovery objective. Stack the input and target pixels of one stage into $Z, Z^- \in \mathbb{R}^{P\times 3}$, and define $B = \mathrm{diag}(\beta)$ and $\Phi_{p,:} = \phi(Z_{p,:})$. For a given input state and spatial support, the continuous executor is

$$\mathcal{R}_\beta(Z, W) = (I_P - B)Z + B\Phi W^\mathsf{T}. \tag{A.4}$$

Defining $H = B\Phi$ and $T = Z^- - Z + BZ$ gives

$$\mathcal{R}_\beta(Z, W) - Z^- = HW^\mathsf{T} - T. \tag{A.5}$$

The recovery objective is therefore

$$J(W) = \|HW^\mathsf{T} - T\|_F^2 + \lambda\|W\|_F^2, \qquad \lambda > 0. \tag{A.6}$$

Pixels with $\beta(p) = 0$ give zero rows of H and contribute only a constant, and small support weights enter the system directly instead of dividing the observed residual by β. A support that vanishes everywhere is handled by a separate branch that returns the identity code, which is not the minimizer of the zero-centered objective in that degenerate case.

Normal equations and uniqueness. Writing $U = W^\mathsf{T}$, the gradient is

$$\nabla_U J = 2H^\mathsf{T}(HU - T) + 2\lambda U. \tag{A.7}$$

Setting it to zero yields

$$(H^\mathsf{T}H + \lambda I_D)U = H^\mathsf{T}T. \tag{A.8}$$

For every nonzero $v \in \mathbb{R}^D$,

$$v^\mathsf{T}(H^\mathsf{T}H + \lambda I_D)v = \|Hv\|_2^2 + \lambda\|v\|_2^2 > 0. \tag{A.9}$$

Thus the system matrix is positive definite, the objective is strictly convex, and its unique global minimizer is

$$(W^*)^\mathsf{T} = (H^\mathsf{T}H + \lambda I_D)^{-1}H^\mathsf{T}T. \tag{A.10}$$

Numerical implementation. We compute W^* by solving the linear system, with regularization strength

$$\lambda = \max\left(10^{-2}\frac{\mathrm{tr}(H^\mathsf{T}H)}{D},\ 10^{-6}\right). \tag{A.11}$$

Features and within-block matrix products use float32; accumulation across blocks and the linear-system solve use float64, and the returned code is converted to float32. For empty support ($\beta \equiv 0$) the rendered state is always Z, so fitting is skipped and the identity code W_{id}, with $\phi(c)W_{\mathrm{id}}^\mathsf{T} = c$, is returned. The derivation concerns the continuous executor; clipping and quantization are applied only to the output.

A.3 Parameter Redundancy and Recovery Stability

Parameter redundancy. Normalized weights give four independent linear relations among the features:

$$\sum_{i=1}^{N} w_i(c) = 1, \qquad \sum_{i=1}^{N} w_i(c)c_j = c_j, \quad j = 1, 2, 3. \tag{A.12}$$

Consequently,

$$\mathrm{rank}(\Phi) \le D - 4 = 4N, \qquad \mathrm{rank}(H) \le 4N. \tag{A.13}$$

Equivalently, adding the same affine transform $cQ + d$ to every local component and subtracting it from the global component gives

$$A_i' = A_i + Q, \quad b_i' = b_i + d, \qquad A_g' = A_g - Q, \quad b_g' = b_g - d, \tag{A.14}$$

with $F_{W'}(c) = F_W(c)$ for every color. Each of the three output channels has four such redundant directions, so the 780 stored coefficients at $N = 64$ contain at least 12 redundant scalar directions.

Ridge regularization adds positive curvature along these directions, so the recovered code is unique; it lies in the row space of H and therefore has the minimum norm among codes with the same fitted output.

Stability under fixed fitting conditions. For fixed H and $\lambda > 0$, changing the target from T to $T + \Delta T$ changes the recovered code by

$$\Delta W^\mathsf{T} = (H^\mathsf{T}H + \lambda I_D)^{-1}H^\mathsf{T}\Delta T. \tag{A.15}$$

For a singular value s of H, the corresponding singular value of the recovery operator is $s/(s^2 + \lambda)$. Since $s^2 + \lambda \ge 2s\sqrt{\lambda}$,

$$\|\Delta W\|_F \le \frac{1}{2\sqrt{\lambda}}\|\Delta T\|_F. \tag{A.16}$$

Regularization thus bounds how target perturbations propagate into the code for a fixed state, support, and basis. Because multi-stage execution changes the intermediate states, full trajectories are evaluated empirically in Appendix A.6.

A.4 Color Codes and Rendered Outputs

Code distance. For two codes on the same input and support, let $\Delta = W_1 - W_2$. Then

$$\|\mathcal{R}_\beta(Z, W_1) - \mathcal{R}_\beta(Z, W_2)\|_F^2 = \|H\Delta^\mathsf{T}\|_F^2 = \mathrm{tr}(\Delta H^\mathsf{T}H\Delta^\mathsf{T}). \tag{A.17}$$

The right-hand side is a squared seminorm on code differences induced by the current state and support, so the same code difference produces different image changes under different color distributions and supports.

Code interpolation and strength. For fixed Z and β, let $W_\alpha = (1-\alpha)W_1 + \alpha W_2$ with $\alpha \in [0, 1]$. The continuous executor satisfies $\mathcal{R}_\beta(Z, W_\alpha) = (1-\alpha)\mathcal{R}_\beta(Z, W_1) + \alpha\mathcal{R}_\beta(Z, W_2)$. In particular, interpolating from the identity code to an editing code continuously adjusts the strength of the same

color action. If the effective support is $\beta = s\,a$ with $s \in [0,1]$, define $W' = (1-s)W_{\mathrm{id}} + sW$. Then

$$\mathcal{R}_a(Z, W') = \mathcal{R}_{sa}(Z, W). \tag{A.18}$$

A uniform support strength can therefore be absorbed into the color code.

Preservation at zero support. The update equation directly gives $\mathcal{R}_\beta(Z, W)(p) = Z(p)$ wherever $\beta(p) = 0$. A pixel with zero support at every stage therefore retains its continuous value throughout the execution chain. The code determines the color change, and the support determines where it is allowed.

A.5 Verification Metrics

Each recovered action is checked per image by a reconstruction error and a split-half stability distance. The reconstruction error bakes the recovered Gaussian mapping into a 33^3 LUT, renders the input with it, and clips and rounds the result to 8 bits; with $\widehat{Y}_8$ the rendered image, Y_8 the target, and P pixels,

$$\rho = \frac{100}{3P} \sum_{p,c} \left| \frac{\widehat{Y}_{8,p,c}}{255} - \frac{Y_{8,p,c}}{255} \right|. \tag{A.19}$$

The floating-point error of the continuous mapping after output clipping is recorded separately and is a different quantity, as is the squared residual of the fitting objective in Eq. (3). For stability, the pixels of one image are split into halves by a fixed seed derived from the sample key and recovered independently as W_a and W_b, while W is recovered from all pixels. With $v(W) = (\mathrm{vec}(W) - \mu)/\sigma$ the standardized code, using the fixed standardization statistics of the color codes with the standard deviation floored at 10^{-8},

$$\delta_{\mathrm{rel}} = \frac{\|v(W_a) - v(W_b)\|_2}{\max(\|v(W)\|_2, 10^{-12})}. \tag{A.20}$$

An edit passes when $\rho < 3$ and $\delta_{\mathrm{rel}} < 0.2$, both strictly, and a trajectory is retained only when every one of its steps passes.

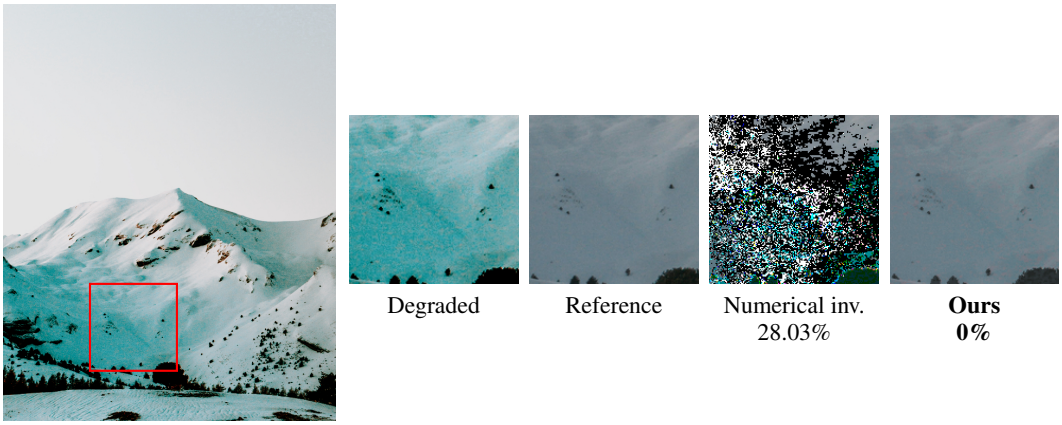
A.6 Degradation–Recovery Study

VeraRetouch and MonetGPT build supervision by degrading high-quality references (Guo et al., 2026; Dutt et al., 2025). Multi-step local supervision additionally needs recovery that is cheap and free of severe local errors, which a low mean reconstruction error can hide. On MaskGrade six-step degradation chains, we compare per-pixel numerical inversion with closed-form color codes fitted from the recorded state pairs.

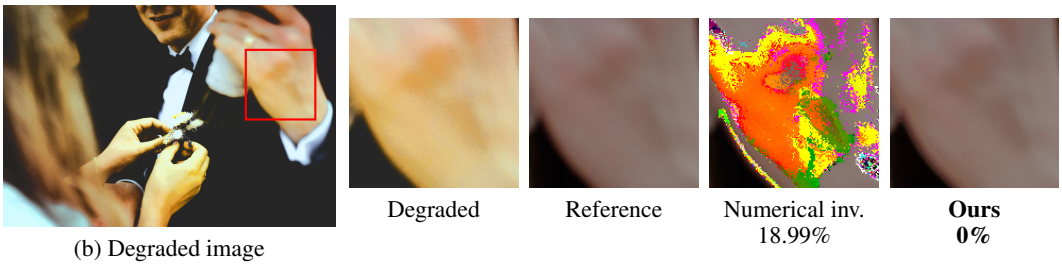
The study uses all 4,051 LUTs in the library to construct 679 six-step chains on 136 source images. Each chain uses six distinct LUTs from the same group and the spatial supports and strength calibration of MaskGrade (Section 3.3). Fitting and evaluation cover the full pixel grid, with the longer image side set to 768 pixels. Numerical inversion uses the known forward LUTs and supports with damped Gauss–Newton, up to 200 iterations, and early stopping at a forward residual of 0.5 intensity levels. Color-code recovery fits known adjacent state pairs with 64 fixed Gaussian bases and the regularization in Appendix A.2. Both methods start from the same degraded endpoint and execute six recovery steps in reverse order.

Outputs are clipped and quantized to 8-bit RGB. A severe-error pixel has a maximum absolute RGB-channel error above 64 intensity levels against the reference, and a chain is free of severe errors if its final reconstruction contains no such pixel. Table A1 also reports the mean severe-error pixel fraction and MAE$\times$100. Paired timings use a fixed sample of 32 chains under the same running conditions, with one warm-up and three repetitions per method; we report the mean of the per-chain median times. Timing covers the six recovery steps, including code fitting and rendering, and output conversion; degradation, image loading, and scoring are excluded.

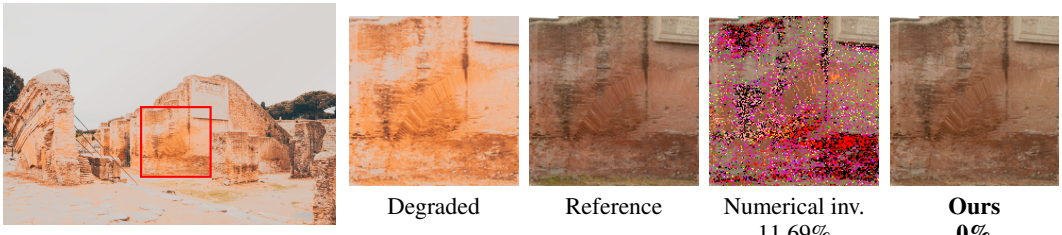
Color-code recovery raises the fraction of chains without severe errors from 54.6% to 75.7% (+21.1 points, 95% CI [16.3, 25.8]) and cuts severe-error pixels from 1.42% to 0.06% at similar MAE, while running 15$\times$ faster (0.128 s versus 1.921 s per chain). Numerical inversion matches the mean



(a) Degraded image



(b) Degraded image



(c) Degraded image



(d) Degraded image

Figure 10: **Severe artifacts after six-step recovery.** Snow, hands, brickwork, and a coastal sky: degraded image with the crop marked (left) and matched crops (right); percentages are full-image severe-error fractions.

Table A1: **Recovery reliability over 679 degradation chains.** Severe-error fractions are per-image means; bold marks the best value per column.

Method	No severe error↑ (%)	Severe pixels↓ (%)	MAE×100 ↓
Degraded input	26.51	4.76	7.56
Numerical inverse	54.64	1.42	**0.64**
Color code (Ours)	**75.70**	**0.06**	0.65

error but leaves concentrated artifacts, such as anomalous patches or bands on snow, hands, brickwork, and a coastal sky, where color-code recovery keeps natural colors and continuous structure (Figure 10).

B Mask Generation and Spatial-Branch Selection

WHERE maps images and instructions to spatial support; WHAT predicts continuous color codes. Their masked execution controls where and how colors change. We describe VLM-based mask generation, architecture selection, and deployment.

B.1 Language-Conditioned Mask Generation

Reading editing regions from VLM representations. The VLM's vision encoder and language backbone learn rich spatial features and semantic representations during pretraining, providing a basis for localizing regions from editing instructions. We freeze these base weights and adapt them through language LoRA (Hu et al., 2022) and learnable queries. Eight WHERE queries follow the image–instruction prompt; their final hidden states guide spatial-feature readout. The 1.433M-parameter LIGHT head reads the final pre-merger vision grid. The adapted Mask2Former head (Cheng et al., 2022) has 21.438M parameters and integrates features from vision layers 6, 12, 18, and 24 through six deformable pixel-encoder layers and nine masked-attention decoder layers.

Continuous spatial support. For Mask2Former, let $\pi_{q,c}$ and $m_q(p)$ denote query q's class probability and mask logit. Then

$$s_c(p) = \sum_{q=1}^{8} \pi_{q,c}\, \sigma(m_q(p)), \qquad a(p) = \frac{s_1(p)}{\max\{s_0(p) + s_1(p), 10^{-8}\}}, \tag{B.1}$$

where $c \in \{0, 1\}$ denotes background or foreground, class probabilities use a softmax including no-object, and σ is sigmoid. We bilinearly resize the foreground probability to the original image dimensions as continuous support for the subject stage. Global support remains one, and the photometric supports are computed analytically from the image they are evaluated on.

Photometric supports. All photometric supports are computed from the stored sRGB values through the luminance proxy

$$\ell = 0.2126R + 0.7152G + 0.0722B, \tag{B.2}$$

which is applied to the stored values directly rather than to linearized light, so it is a coding-space proxy and not a physical luminance. Let $S(t) = u^2(3 - 2u)$ with $u = \mathrm{clip}(t, 0, 1)$, and let $t_l = Q_{q_l}(\ell)$ and $t_h = Q_{q_h}(\ell)$ be quantiles of the image's own luminance-proxy distribution. With

$$a = S\left(\frac{\ell - (t_l - w)}{2w}\right), \qquad b = S\left(\frac{\ell - (t_h - w)}{2w}\right), \tag{B.3}$$

the Shadows, Midtones, and Highlights supports are $1 - a$, $a(1 - b)$, and b. They overlap inside the transition bands of width $2w$ and are neither mutually exclusive nor constrained to sum to one. The Hue support gates a hue weight by saturation,

$$M_{\mathrm{hue}} = h(H)\, S\left(\frac{s - 0.08}{0.25 - 0.08}\right), \tag{B.4}$$

where H and s are the HSV hue and saturation of the same stored values, and h interpolates linearly on the hue circle between the anchors $(0°, 0.15)$, $(60°, 0.15)$, $(120°, 0.45)$, $(180°, 0.70)$, $(230°, 1.0)$, and $(300°, 0.45)$, with the last segment wrapping back to $360°$. The saturation gate excludes near-gray pixels, and the anchors weight cool hues more strongly than warm ones.

The same functions run on different inputs. Trajectory construction evaluates the photometric supports on the original reference Z_0 and draws $q_l \in [0.30, 0.40]$, $q_h \in [0.60, 0.70]$, and $w \in [0.12, 0.24]$ once per chain, and replay multiplies the calibrated stage strength into the support. Local training and inference instead evaluate them on the current stage state, which during training is the predicted state, with the fixed values $q_l = 0.35$, $q_h = 0.65$, and $w = 0.18$; the quantiles are

still taken from the image at hand, and the supports are computed from a detached, clamped copy of that state. The Subject support is the recorded construction mask during construction and local training and the predicted mask at inference. A recorded support therefore need not coincide with the support executed during a rollout, the Global stage being the exception since its support is one in both cases; Section 4.4 measures what replacing predicted supports with recorded ones changes.

B.2 Spatial Representation Reuse and Mask Quality

Effective lightweight readout. We select approximately 20,000 subject-mask examples from MaskGrade, group records by shared source images or masks, and form disjoint training, validation, and test groups. Both recipes share 17,059 training examples, initialization, and training budget; each trains language LoRA, queries, and a segmentation head while freezing base language weights and the vision encoder. LIGHT uses binary cross-entropy and soft Dice; Mask2Former uses set-prediction and auxiliary losses, so results compare complete segmentation recipes. Agreement uses validation-selected checkpoints and a 0.5 threshold, averaged within source groups and then equally across groups. LIGHT achieves higher IoU and Dice on both sets (Table B1). With language and query adaptation, VLM features thus support a lightweight head with high mask-reference agreement, illustrating the value of reusing pretrained representations for spatial readout.

Table B1: **Subject-mask agreement.** IoU and Dice (%) with pseudo-labels on the validation and test splits.

Evaluation set	Source groups	Head	IoU↑	Dice↑
MaskGrade validation	968	LIGHT	**90.74**	**94.72**
	968	Mask2Former	89.13	93.66
MaskGrade test	980	LIGHT	**89.92**	**94.09**
	980	Mask2Former	88.08	92.91

Selection for local editing. Masks weight color residuals spatially, so boundary predictions directly affect the extent of local color changes. We therefore consider both agreement with supervision and how mask coverage and contours correspond to image content. Table B1 uses automatically generated pseudo-labels; higher overlap does not directly imply more accurate image boundaries. Our qualitative review of 50 fixed validation source groups favored Mask2Former. We therefore use Mask2Former's multiscale decoder to read subject masks from VLM representations.

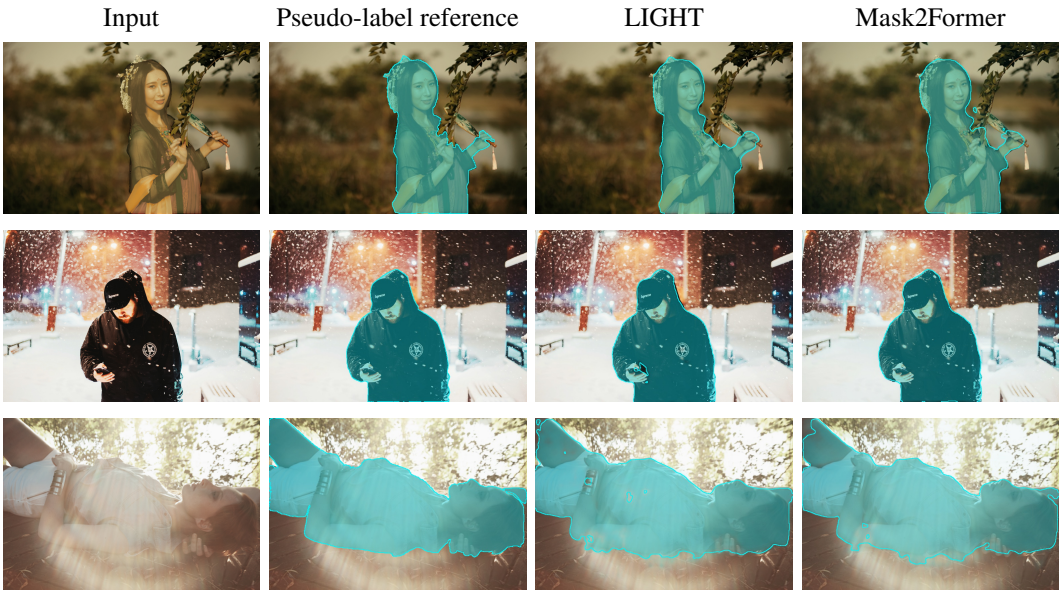


Figure 11: **Subject-mask predictions.** Cyan shows foreground probability; contours mark the 0.5 threshold.

B.3 Separate Spatial and Color Deployment

Preserving branch-specific readout. Spatial localization and color prediction share VLM base weights but use their own learned adapters and queries. We compare merging both adapters for a single forward pass against separate passes that activate each branch's adapters and queries independently. Merging counts the shared initialization increment once, and we compare both query orders. Table B2 measures reconstruction quality and execution cost on the same six ArtEdit examples. All configurations share the same color and spatial models, resize VLM inputs to short side 512, and render at the original size; each is timed with two warm-ups and five repetitions in alternating order, and end-to-end time covers preprocessing, prediction, mask upsampling, rendering, and output transfer.

Table B2: **Branch deployment quality and latency.** Six-image mean PSNR (dB) and latency (ms, mean $\pm$ s.d. over five repetitions).

Deployment / query order	PSNR↑ (dB)	Inference (ms)	Rendering (ms)	End-to-end (ms)
Merged: WHERE→WHAT	20.324	131.85 ± 2.40	259.57 ± 3.21	391.42 ± 5.32
Merged: WHAT→WHERE	21.204	132.86 ± 3.21	259.23 ± 1.87	392.09 ± 2.93
Separate WHERE / WHAT	**25.082**	271.58 ± 14.25	260.71 ± 1.58	532.29 ± 13.95

Quality–cost trade-off. Separate passes improve mean PSNR by 3.9 dB over the better merged configuration at 140 ms more end-to-end latency, so keeping each branch's own adapters and queries is worth one extra forward pass. We therefore run the branches separately: the spatial branch uses Mask2Former with eight queries, and the color branch uses six groups of eight queries.

C Additional Experimental Results

This appendix defines the MaskGrade-400 diagnostic set (C.1) and reports results not shown in Section 4: additional interventions (C.2) and recovery accuracy and stability (C.3). Training and evaluation protocols are in Appendices F and G. Unless stated otherwise, intervals are 95% bootstrap intervals over 2,000 resamples of source-image groups, and pixel errors and drift are in 0–255 levels.

C.1 MaskGrade-400 Diagnostic Set

Diagnostic set. We build one six-stage degradation trajectory from each of the 400 ArtEdit targets (391 source images) with the MaskGrade primitives at a maximum side of 512 and re-annotate its overall instruction with GPT-5.6 Terra. Subject regions for construction come from SAM 3 masks (Carion et al., 2026); at test time the subject support is predicted by the Mask2Former head, which sees neither. All chains are kept, including 29 empty Subject supports and three zero-effect Hue stages.

Metrics. With recorded input x_{k-1}, target x_k, support a_k, and prediction $\widehat{x}_k$ obtained by applying the predicted code to x_{k-1},

$$T_k = 255\,\overline{|\widehat{x}_k - x_k|}, \quad D_k = 255\,\overline{|\widehat{x}_k - x_{k-1}|}_{\,a_k \le 0.05}, \quad A_k = \cos\big(\widehat{x}_k - x_{k-1},\, x_k - x_{k-1}\big).$$

Global has no outside region; zero reference effects are excluded from alignment, and a zero prediction against a nonzero effect scores zero. Values are averaged within each chain and then across chains; rollout PSNR compares the clipped final output with the recorded target. Instruction scores use the benchmark judge and prompt; the judge sees the degraded input, one output, and the instruction, but never the target. Table 3 and Figure 9 report alignment and drift for the supervision objectives.

C.2 Test-Time Interventions

Conditions. All conditions use the 400 chains and fixed checkpoints; the normal order is Global, Hue, Shadows, Midtones, Highlights, Subject. Random order applies one fixed non-original permutation per chain; the Shadows/Highlights (S/H) order swap moves both operations with their codes, while the support and code swaps exchange only the named factor in place. Removal skips a stage, and Subject only applies the Subject operation to the degraded image. Oracle replacement instead substitutes the recorded construction supports, or codes fitted once per chain on the recorded transitions by support-weighted ridge regression, for the predicted ones. The generic marker replaces every stage marker with one word, keeping token positions, and requires a new color-readout pass; all other conditions reuse the predicted codes and supports. Drift and alignment are measured along each intervened rollout, so their absolute values are not directly comparable with the single-step values in Table 3, which apply each predicted code to the recorded preceding state.

Results. Figure 8 reports the main conditions for Ours; Table C1 adds two conditions and the Stage + final model. Stage + final supervision reproduces every effect in Figure 8: randomizing the order, replacing the markers, or removing the Subject stage lowers SC by 0.93–1.04, and swapping the S/H supports raises drift from 0.58 to 2.87. Removing Hue barely changes SC but raises PSNR, and Subject-only execution reaches the highest PSNR with the lowest SC, so agreement with the constructed target and completion of the instruction can move in opposite directions. Individual stages therefore need not improve reference fidelity in isolation: the structured execution is optimized for instruction-conditioned decomposition rather than for maximal fidelity at every stage.

C.3 Color Recovery

Setup. Every fitter estimates the color transform of one image pair from both images. The frozen-basis fitters share 64 Gaussians and the per-pair ridge coefficient and differ only in the solver; CGLUT-style fitting also optimizes opacity and per-Gaussian scale; NILUT and the VeraRetouch MLP minimize an unregularized L1 objective with 33,923 and 2,580,483 parameters. Adam runs 500 updates, and L-BFGS stops at convergence or after 500 iterations. The ArtEdit set has 100 fixed pairs; the MMArt-FiveK set has 500 prefiltered pairs, of which CGLUT-style fitting uses the first 200 and the MLP fitters 114. Images are resized to short side 512, and timing covers per-pair fitting only; closed-form timing includes feature computation, which the frozen iterative fitters precompute. The split-half distance δ_{rel} and the reconstruction error ρ are defined in Appendix A.5, and Init. compares two initializations (repeated solves for the closed form).

Results. Fitting times are shown in Figure 6. The closed-form solve reaches the L-BFGS optimum and is the only fitter whose coefficients do not depend on initialization (Table C2). Jointly fitting the Gaussian geometry makes the objective nonconvex: CGLUT-style codes move by 0.82–0.84 between initializations, and only 21–42% of pairs pass the recovery check, against 60–100%

Table C1: **Additional test-time interventions.** Paired changes from normal execution on MaskGrade-400; Figure 8 shows the remaining conditions for Ours.

Condition	ΔSC	ΔO	ΔPSNR	Drift↓	Align.↑
Rollout-stage (Ours): conditions not shown in Figure 8					
Remove Hue	−0.02 [−0.17, 0.13]	+0.00	+0.74 [0.64, 0.84]	0.51	0.289
Subject only	−3.07 [−3.37, −2.78]	−2.05	+1.62 [1.15, 2.09]	1.11	0.204
Stage + final: normal SC 8.16, PSNR 21.34					
Normal	—	—	—	0.58	0.268
Random order	−0.93 [−1.10, −0.76]	−0.76	−0.08 [−0.11, −0.05]	0.62	0.269
Generic marker	−1.00 [−1.17, −0.81]	−0.78	+0.43 [0.38, 0.48]	0.53	0.277
Remove Subject	−1.04 [−1.22, −0.87]	−0.79	+0.44 [0.38, 0.50]	0.46	0.281
Remove Hue	−0.16 [−0.29, −0.01]	−0.13	+0.69 [0.60, 0.80]	0.47	0.282
Subject only	−3.32 [−3.62, −3.04]	−2.28	+1.24 [0.80, 1.69]	1.04	0.202
Swap S/H order	−0.05 [−0.17, 0.07]	−0.03	+0.00 [−0.00, 0.01]	0.55	0.268
Swap S/H support	−0.80 [−0.96, −0.67]	−0.65	−0.07 [−0.08, −0.05]	2.87	0.191
Swap S/H code	−0.02 [−0.14, 0.09]	−0.02	−0.07 [−0.08, −0.05]	0.57	0.268

for frozen-basis fitters. Uniqueness refers to the regularized objective; Appendix A derives it and discusses parameter redundancy.

Table C2: **Color recovery accuracy and stability.** Medians at short side 512; Pass: fraction with $\rho < 3$ and $\delta_{\text{rel}} < 0.2$.

	ArtEdit ($n = 100$)			MMArt-FiveK ($n \leq 500$)		
Fitter	MAE×100↓	Init.↓	Pass↑	MAE×100↓	Init.↓	Pass↑
Closed-form ridge (Ours)	2.413	**0.000**	**0.63**	1.795	**0.000**	0.98
Frozen basis, L-BFGS	2.413	0.010	**0.63**	1.795	0.013	**1.00**
Frozen basis, Adam	2.435	0.048	0.60	1.832	0.057	0.98
CGLUT-style, Adam	3.636	0.819	0.21	2.109	0.839	0.42
NILUT, Adam	2.531	1.404	—	1.667	1.418	—
VeraRetouch MLP, Adam	2.631	1.261	—	2.306	1.252	—

D MaskGrade Data and Annotation

We describe the language of the training annotations and the annotation protocol.

D.1 Scene and Editing Vocabulary

The 73,853 training trajectories contain 24,474 distinct source-image IDs. For each source image, we select the record with the lexicographically smallest training key. We extract scene and subject terms from its six observation fields and editing intents from its medium-length user instruction. Documented lexical families merge singular, plural, and related word forms; each family is counted at most once per source image. Figure 12 summarizes these overlapping, annotation-derived vocabularies.

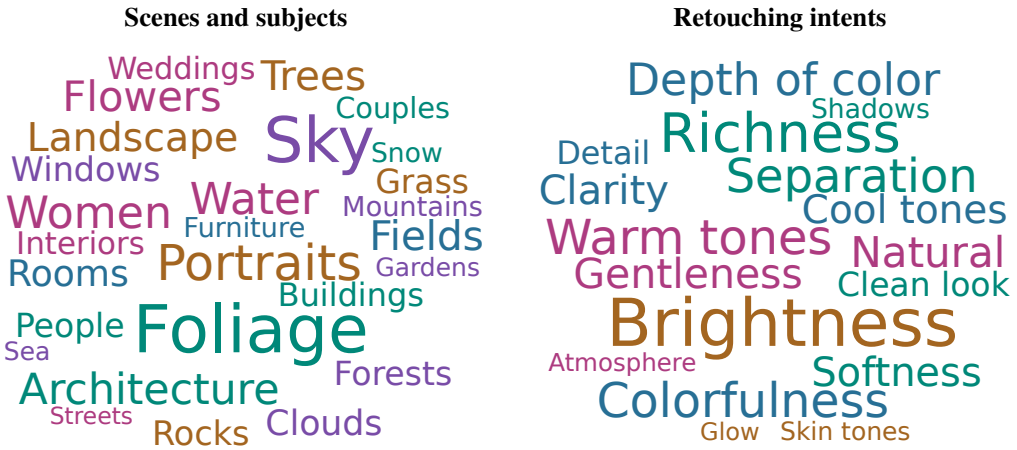


Figure 12: **MaskGrade vocabulary.** Frequent scene and subject terms (left) and retouching intents (right); word size reflects source-image frequency.

D.2 Annotation Inputs and Output Contract

The annotator is GPT-5.6 Terra. It receives three images in order: the current image, its target appearance, and a two-row, seven-column stage breakdown, together with a facts sheet extracted from the construction record. The current image is the degraded state, and the target appearance is the original reference. The annotator returns six ordered records with observation, mask, and adjustment fields, followed by long (60–110 words), medium (20–35 words), and short (5–12 words) user instructions. The parser checks required fields, nonempty text, stage count, and the English-language format. The editing model receives only the current image and a user request; the target and breakdown images are used for annotation only. The complete annotation prompts are provided in Appendix H.

E Local Editing Trajectories

Each example shows (1) the degraded input, the recovered output, and the reference; (2) the recovered states after each of the six stages; (3) two local stages with their supports, residuals, and matched crops inside and outside the support; and (4) excerpts of the stagewise annotation.

Coastal sky

Training example (Unsplash).

1. Before / After / GT



2. Iterative recovery



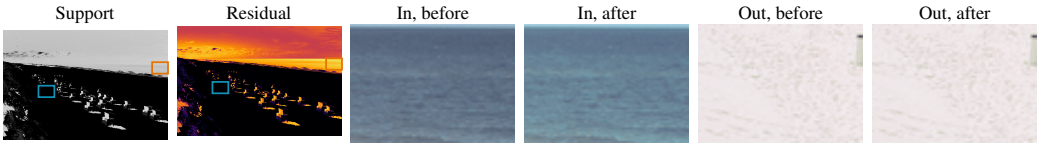


Instruction. Make the current beach photo brighter and more colourful, with a clearer blue sea and sky, richer shadows, controlled pale sand, and chairs that stand out gently in the centre.

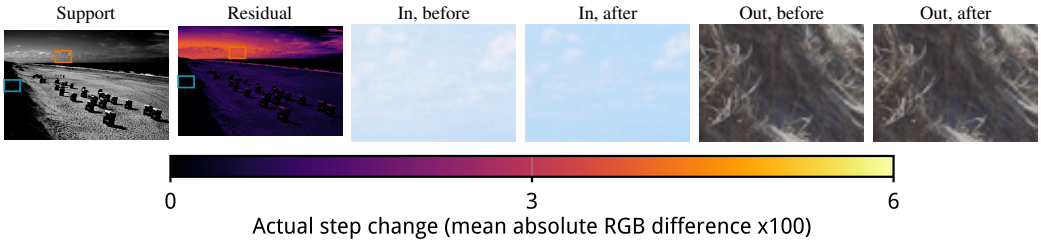
3. Local changes and unchanged controls

Orange box: mask-in ($\beta > 0$). Blue box: mask-out ($\beta = 0$).

Step 2 / Hue



Step 5 / Highlights



The matched mask-out crops are unchanged. Residuals show the actual adjacent-state RGB change; the two steps share one color scale.

4. Stagewise reasoning

Excerpts from the trajectory annotation.

Step 1 / Global. The current image is pale and slightly cool, with restrained colour separation across the hazy sky, dark sea, dune vegetation, and chairs. *Support:* The global grade covers the whole frame evenly, with no mask: sky, sea, sand, dune, and beach chairs all receive the same broad colour foundation.

Step 2 / Hue (detail shown above). After the global pass, the beach is cleaner but the saturated coastal colours still lack lift and separation. *Support:* Use the colour-range mask only on reasonably saturated pixels, weighted mostly toward the blues, cyans, and greens; it should favour the ocean, stronger sky colour, and saturated dune or chair details while leaving pale sand comparatively quiet.

Step 3 / Shadows. The darker dune, chair interiors, and sea shadows remain comparatively neutral and soft after the colour-range pass. *Support:* Apply a luminosity mask to the shadows, with a soft roll-off into neighbouring tones.

Step 4 / Midtones. The broad beach and chair faces now carry colour, but the middle tonal band is still subdued and slightly muddy, making the scene feel flatter than the clear seaside light calls for. *Support:* Use a luminosity mask holding the midtones, with a soft roll-off into both shadows and highlights.

Step 5 / Highlights (detail shown above). The brightest sand, cloud, and foam areas are airy but washed out, with too little hue distinction. *Support:* Use a luminosity mask holding the highlights, again with a soft roll-off into neighbouring tones.

Step 6 / Subject. The chair grouping is readable but still sits too similarly to the surrounding sand. *Support:* Use a radial mask: a broad oval over the centre of the frame, acting inside it and fading outward through a very soft, wide falloff.

Deer in the shade

Training example (Unsplash).

1. Before / After / GT



2. Iterative recovery

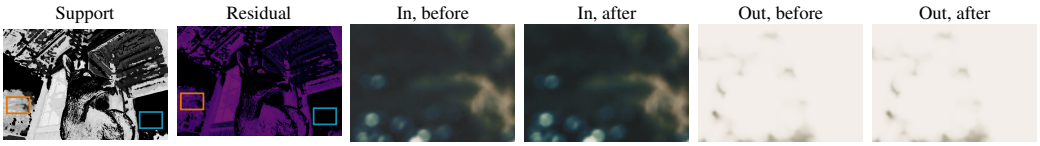


Instruction. Make the current image feel cooler and more dramatic, with richer blue and green surroundings, brighter detail in the dark bridge, and a clearer, gently brighter deer.

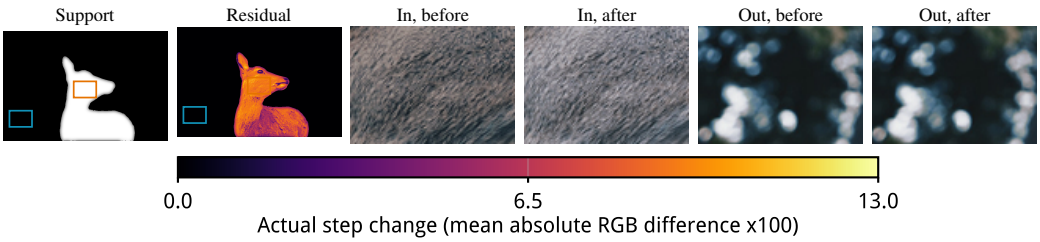
3. Local changes and unchanged controls

Orange box: mask-in ($\beta > 0$). Blue box: mask-out ($\beta = 0$).

Step 2 / Hue



Step 6 / Subject



The matched mask-out crops are unchanged. Residuals show the actual adjacent-state RGB change; the two steps share one color scale.

4. Stagewise reasoning

Excerpts from the trajectory annotation.

Step 1 / Global. The current image is generally muted and warm-grey, so the bridge, foliage, and deer lack the cool, richly separated colour foundation needed for a more atmospheric frame. *Support:* Use the global mask over the whole frame, carrying the foundation equally through the dark bridge structure, bright openings, foliage, and deer.

Step 2 / Hue (detail shown above). After the foundation, the naturally coloured exterior still needs more presence: the blue-cyan openings and green foliage remain comparatively quiet beside the heavy, neutral bridge shadows. *Support:* Use the colour-range mask only on reasonably saturated pixels, weighted most strongly through the blue, cyan, and green material in the foliage, open sky, and coloured bridge detail.

Step 3 / Shadows. The deepest tonal range is still too closed and brownish; under the bridge, structure is getting swallowed, and the deer's shadow-side texture needs a more coloured, breathable lift. *Support:* Work in the luminosity mask that holds only the shadows, with a soft roll-off into neighbouring tones, concentrated in the under-bridge ceiling, deep supports, and the deer's shaded coat.

Step 4 / Midtones. The central working tones need cleaner separation: the deer's fur and the bridge planes are still a little flat, while the yellow-brown midtone content feels too heavy against the emerging cool blue notes. *Support:* Use the midtone luminosity mask with a soft transition into adjacent values, affecting the deer's face and neck, the bridge planes, and the foliage between the deepest shadows and bright sky.

Step 5 / Highlights. The bright openings and rim-lit details need to feel more deliberate rather than simply pale; the highlights need cooler chromatic energy and clearer colour contrast without disturbing the dark architectural mood. *Support:* Restrict this to the highlight luminosity mask, rolling softly into the neighbouring tones across the white sky gaps, bright edge light on the deer, and reflective bridge trim.

Step 6 / Subject (detail shown above). The deer still sits slightly too heavily in the scene after the environmental work, especially through the dark body and cool-grey fur, so it needs a final controlled lift and a touch more colour separation. *Support:* Use the segmented main-subject mask over the deer, reaching 0.20 of the frame with a 0.15 full-strength core; keep the correction contained to the animal's head, neck, and body outline.

Outdoor portrait

Training example (PPR10K).

1. Before / After / GT

Before | After | GT



2. Iterative recovery

Input (z_0) | z_1: Global | z_2: Hue | z_3: Shadows





Instruction. Make the current image warmer and richer, deepen the darker areas, add colour to the background, brighten the light, and make the woman stand out naturally.

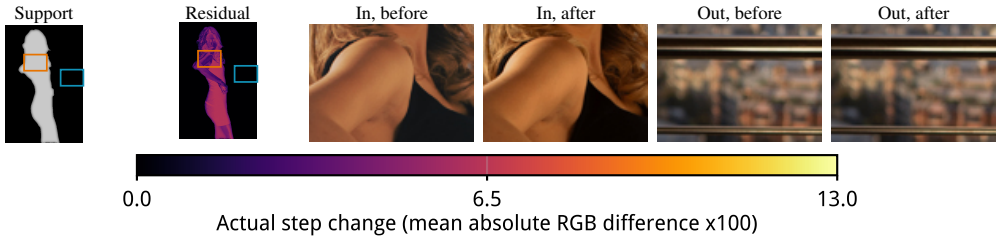
3. Local changes and unchanged controls

Orange box: mask-in ($\beta > 0$). Blue box: mask-out ($\beta = 0$).

Step 5 / Highlights



Step 6 / Subject



The matched mask-out crops are unchanged. Residuals show the actual adjacent-state RGB change; the two steps share one color scale.

4. Stagewise reasoning

Excerpts from the trajectory annotation.

Step 1 / Global

Observation. The current image has a cool, somewhat muted overall balance, with the background and dress lacking the warmer, denser colour separation needed for a polished portrait grade. Establish the base palette first, while slightly lowering luminance through most colour families to contain the frame.

Support. No mask: apply this global move evenly across the whole frame, including the sky, city, railings, dress, and subject.

Step 2 / Hue

Observation. The saturated environmental colour is still restrained and slightly cool, especially in the sky, distant hills, and city tones. This move needs to give those selected colours more presence without broadly colouring the neutral subject or the metalwork.

Support. Use the colour-range mask over reasonably saturated pixels, weighted mostly toward the blues, cyans, and greens; let it fall away from less-saturated skin, black fabric, and neutral railings.

Step 3 / Shadows
Observation. The lower values remain cool and colourless, making the dress and shadowed architecture merge into a heavy block. Enrich the shadows with warmer, stronger colour while reducing their luminance so the image keeps a controlled, cinematic low end.
Support. Use the luminosity mask holding only the shadows, with a soft roll-off into neighbouring tones; concentrate on the dark dress, hair recesses, lower deck, and deep city detail.

Step 4 / Midtones
Observation. The middle tonal band is still a little flat and hazy, particularly through the distant city and the transition between the subject and background. Shape it darker and more chromatically separated, keeping the strongest colour emphasis in yellow, green, and blue areas.
Support. Use the luminosity mask holding only the midtones, with a soft roll-off into neighbouring tones; work across the city, hills, skin transitions, balcony surfaces, and dress mid-values.

Step 5 / Highlights (detail shown above)
Observation. The bright areas need more lift and colour sparkle to counter the denser shadows and midtones. Open the red and orange highlights, add saturation across the warm and green families, and keep the sky and specular detail luminous without making them harsh.
Support. Use the luminosity mask holding only the highlights, with a soft roll-off into neighbouring tones; include the bright sky, hair highlights, sunlit skin, rail reflections, and bright city points.

Step 6 / Subject (detail shown above)
Observation. After the environmental grade, the woman still needs to sit forward from the balcony and city while her skin and hair retain a convincing warm presence. Strengthen subject colour separation, slightly deepen the yellow and orange skin values, and add controlled lift to selected cooler details.
Support. Use the segmented main-subject mask over the woman, covering 0.26 of the frame with a core at full strength over 0.20 of the frame; keep the edge transition natural around hair, arms, hands, dress, and legs.

Low-key portrait
Training example (PPR10K).

1. Before / After / GT



2. Iterative recovery

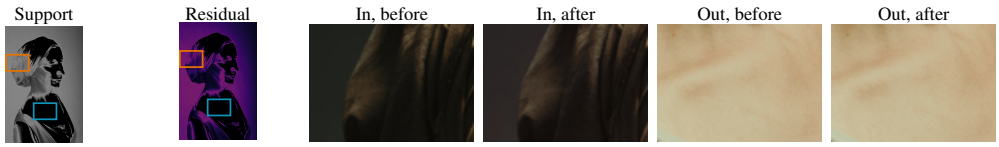


Instruction. Make the portrait less yellow-green, brighten the face and clothing, add richer colour separation, and keep the shadows deep but clearer.

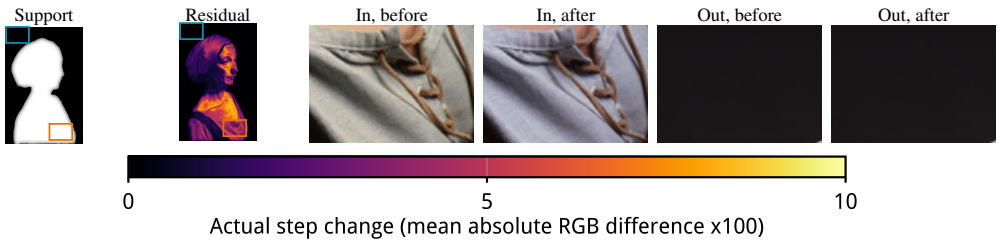
3. Local changes and unchanged controls

Orange box: mask-in ($\beta > 0$). Blue box: mask-out ($\beta = 0$).

Step 4 / Midtones



Step 6 / Subject



The matched mask-out crops are unchanged. Residuals show the actual adjacent-state RGB change; the two steps share one color scale.

4. Stagewise reasoning

Excerpts from the trajectory annotation.

Step 1 / Global
Observation. The current image has a strongly yellow-green overall cast: the skin, turban and dress all lean warm and olive, while the colour separation is restrained. The global pass needs to establish a richer, more dimensional palette before local tonal work.
Support. Global grade over the whole frame, with no mask.
Step 2 / Hue
Observation. After the global colour lift, the saturated cool and green areas still need a more deliberate shift, while some orange and yellow regions remain too bright and warm. This pass concentrates the chromatic separation without disturbing neutral, low-saturation areas.
Support. Colour-range mask over reasonably saturated pixels, weighted mostly toward the blues, cyans and greens.
Step 3 / Shadows
Observation. The darker folds of the clothing, the shadow side of the turban and the recesses around the face are still dense and olive, so the low-end colour is not yet separating cleanly. Open those shadows with a cooler, more articulate chromatic lift while retaining the portrait's depth.
Support. Luminosity mask holding only the shadows, with a soft roll-off into neighbouring tones.
Step 4 / Midtones (detail shown above)
Observation. The midtone skin, fabric and illuminated folds still read somewhat muddy and monochromatic. Build colour separation through the main working tones, especially the greens, blues and purples, while giving the face and clothing a cleaner, brighter presence.
Support. Luminosity mask holding only the midtones, with a soft roll-off into neighbouring tones.
Step 5 / Highlights
Observation. The brightest turban, cheek and neckline highlights are carrying too much yellow-green brightness and lack the intended controlled, softly coloured finish. Tame their luminance slightly while increasing highlight saturation and refining each hue so the light remains luminous rather than harsh.
Support. Luminosity mask holding only the highlights, with a soft roll-off into neighbouring tones.
Step 6 / Subject (detail shown above)
Observation. The scene now has the intended tonal and colour structure, but the subject still carries too much yellow-green warmth and the portrait does not separate with enough refined skin and fabric colour. Finish by tailoring the subject independently from the background.
Support. Segmented main-subject mask covering 0.52 of the frame, with the core at full strength covering 0.40; feather the falloff naturally around the figure.

F Training Data and Optimization

The Gaussian basis of the color executor is learned first and then frozen; the color model is then trained in three stages on a Qwen3-VL-4B-Instruct backbone whose visual tower stays frozen throughout. Table F1 summarizes the data and update counts; the spatial branch is described in Appendix B.

Table F1: **Training stages.** Updates of the evaluated checkpoint, with schedule length in parentheses.

Stage	Training data	Updates
Gaussian basis	3,939 preset 3D LUTs	20,000
Color-language SFT	73,853 annotated trajectories	9,232
Readout initialization	49,963 style records, 16,163 expert pairs	800 (6,250)
Local training	75,311 trajectories, 31,163 global replay examples	6,000 (6,250)
Subject-support head	17,059 subject-mask examples	2,800 (3,200)

Gaussian basis. The centers, covariances, and opacities of the $N = 64$ Gaussians in Eq. (2) are learned once and then frozen for data construction, training, and inference. They are fitted on a library of 4,051 3D LUTs (33^3) drawn from two collections of retouching presets, of which 112 are held out. Each update samples 64 LUTs and two sets of 8,192 colors, half uniform in the RGB cube and half from the pixels of 2,000 training images; each LUT's code is solved in closed form (Eq. 3, $\lambda = 10^{-4}$) on the first set, and the L1 error of the resulting mapping on the second set is back-propagated through the solve to the geometry. The geometry is therefore optimized for the error that remains after the best code rather than for any single code. Adam runs for 20,000 updates with learning rate 3×10^{-3}, 200 warmup updates, and cosine decay; the held-out LUTs are reproduced with a mean L1 error of 0.0044.

Color-language SFT. We fine-tune the full language branch for two epochs on 73,853 annotated MaskGrade trajectories, with 3,786 validation records; one of the 73,854 training records is excluded by the image-geometry check. Six stage tokens are added to the vocabulary, and the loss is token cross-entropy on the assistant response. Optimization uses AdamW with learning rate 10^{-5}, no weight decay or warmup, gradient clipping at 1.0, and effective batch size 16.

Readout initialization. With the SFT weights frozen, we attach rank-32 LoRA (Hu et al., 2022) ($\alpha = 32$) to the query, key, value, output, gate, up, and down projections of all 36 language blocks, eight learnable readout tokens, and a layer-normalized linear head over the concatenated readout states. Training alternates batches of 49,963 single-edit style records, whose targets are rendered from known LUTs, and 16,163 real expert pairs (12,512 from MIT-Adobe FiveK experts A–E and 3,651 from PPR10K), with effective batch size 16. The loss combines pixel L1 with a mean squared error to the standardized closed-form code and a prototype InfoNCE term ($\tau = 0.07$, 5,075 prototypes), each auxiliary term weighted by 0.1. AdamW uses weight decay 0.01, learning rate 10^{-4} for LoRA and 10^{-3} for the readout tokens and head, 100 warmup updates followed by cosine decay over 6,250 updates, and gradient clipping at 1.0.

Local training. Starting from this initialization with a fresh optimizer, training alternates global replay and local trajectory batches. Global batches draw from 15,000 style records and the 16,163 expert pairs and supervise one code with full-image support against the endpoint. Local batches draw from 75,311 six-stage MaskGrade trajectories and supervise the rollout with the objectives in Eq. (7). The subject stage uses the recorded construction support, and photometric supports are recomputed on each predicted state with stopped gradients. All objectives use image MAE only. The prototype-selection head is frozen; LoRA, readout tokens, and the code head are trained with AdamW (weight decay 0.01), learning rate 10^{-4} for LoRA and 10^{-3} for the readout tokens and head, 100 warmup updates, cosine decay over 6,250 updates, gradient clipping at 1.0, and effective batch size 16. We evaluate the checkpoint at update 6,000.

Supervision variants. All four supervision variants start from the same initialization and use the same batch order and seed, and all are evaluated at update 6,000. *Final only* supervises the last state of the six-stage rollout ($\mathcal{L}_{\text{final}}$), *Stage + final* uses $\frac{1}{2}(\mathcal{L}_{\text{rollout}} + \mathcal{L}_{\text{final}})$, and *Rollout-stage* (Ours) uses $\mathcal{L}_{\text{rollout}}$ (Eq. (7)). Global endpoint sees the same batches but reduces each local trajectory to its degraded-input and final-target pair, supervising a single code with full-image support.

Subject-support head. The Mask2Former subject-support head is trained for 3,200 updates and selected at update 2,800 by hard IoU on a held-out validation split that is disjoint from all evaluation benchmarks (Appendix B).

G Evaluation Protocols and Metrics

Baseline configurations. The generative editors in Table 1 are EyeControl (Qin et al., 2026), FLUX.1 Kontext [dev] (Black Forest Labs et al., 2025), Qwen-Image-Edit-2511 (Wu et al., 2025), and InstructPix2Pix (Brooks et al., 2023). The retouching and agent-based baselines are JarvisEvo (Lin et al., 2026), JarvisArt (Lin et al., 2025), VeraRetouch in instruction mode (Guo et al., 2026), and MonetGPT (Dutt et al., 2025).

Baseline inclusion. We quantitatively evaluate methods whose inference assets are publicly available and can be run under our common input–instruction protocol. Each baseline is evaluated as released, and we do not control what it was trained on; FiveK and PPR10K are common sources for retouching models, so the expert benchmarks may be in-domain for several baselines as well as for our expert-pair training data. InstantRetouch (Wu et al., 2026a) is closely related in its focus on efficient instruction-guided retouching, but the checkpoint and evaluation assets needed to reproduce its setting were not publicly available at submission time, so we discuss it as related work instead of reporting cross-paper numbers obtained under a different protocol.

Backbone scale. Table 1 reports the nominal parameter scale of the principal language backbone for agentic retouching methods and the denoising backbone for generative editors, rather than full-pipeline or trainable-only parameter counts. MonetGPT, JarvisArt, JarvisEvo, and VeraRetouch use the 7B, 7B, 8B, and 0.5B language-backbone classes, respectively. EyeControl and FLUX.1 Kontext use a 12B flow backbone; Qwen-Image-Edit uses a 20B generation backbone; the InstructPix2Pix denoiser has approximately 0.86B parameters. Auxiliary image/text encoders, autoencoders, control modules, adapters, and task-specific heads are excluded from this column. In particular, adding the nominal 7B text encoder to the Qwen-Image-Edit generator would give a different, 27B accounting scope. Both Ours rows use the same Qwen3-VL-4B checkpoint with learned readout queries; six-stage execution runs the WHERE and WHAT adapters in separate passes (Appendix B). The expert-benchmark table uses the six-stage configuration.

Reference fidelity. L1 and L2 are mean absolute and mean squared RGB errors against the reference, with L2 penalizing large deviations more strongly. PSNR (Wang & Bovik, 2009) measures pixel reconstruction fidelity on a logarithmic scale and is computed as $10 \log_{10}(255^2/\mathrm{MSE})$ over the stored 8-bit RGB images. SSIM (Wang et al., 2004) compares local luminance, contrast, and structure using an 11×11 Gaussian window with standard deviation 1.5, constants $K_1 = 0.01$ and $K_2 = 0.03$, and an average over RGB channels. CIEDE2000 (ΔE_{00}) (Sharma et al., 2005) measures perceptual color differences in CIELAB: sRGB is converted using the D65 white point, with $k_L = k_C = k_H = 1$, and color differences are averaged over pixels. Expert-reference evaluation requires matching image dimensions and uses the designated expert target. The ArtEdit L1/L2 protocol resizes predictions to the target dimensions when needed and scales values by 10^2 and 10^3, respectively. Metrics are computed per image and averaged over valid outputs.

Instruction conditioning and no-reference quality. We use the ArtEdit VIEScore prompt (Ku et al., 2024) with Gemini 3.1 Pro as the judge: SC measures instruction adherence, PQ measures content consistency as defined by that prompt rather than the perceptual quality of the original VIEScore formulation, and O is their per-image geometric mean. No-reference quality is assessed with ArtiMuse (Cao et al., 2026), Q-Align in aesthetics mode (Wu et al., 2024), and DeQA-Score (You et al., 2025).

ArtEdit valid-output counts. The baseline rows in Table 1 use 398 outputs for JarvisEvo and 396 for JarvisArt, both rendered with Lightroom, 399 for VeraRetouch in instruction mode, and 400 for MonetGPT. InstructPix2Pix, Qwen-Image-Edit-2511, FLUX.1 Kontext [dev], and EyeControl each use 400 outputs. Both Ours rows supply all eleven metrics on 400 outputs.

Expert-benchmark output counts. Table 2 uses 498/492 outputs for MonetGPT and JarvisEvo, 497/490 for JarvisArt, 496/492 for VeraRetouch, and 498/492 for Ours on FiveK/PPR10K, respectively. In both tables, bold and underlined values mark the best and second-best result for each metric.

Latency. Figure 7 times each method on the same 50 source images, resized to the stated short edge with aspect ratio preserved and padded to a multiple of 16 pixels. All methods run on one datacenter NPU in BF16 with batch size 1; each image is timed three times after two warm-up calls, and we report the mean end-to-end time, including preprocessing, model inference or denoising, and rendering or decoding, but excluding image loading. JarvisArt, JarvisEvo, and MonetGPT execute their edits in external software (Lightroom Classic or GIMP) and are not timed; generative editors are reported up to short edge 1024.

G.1 Expert-Pair Instruction Annotation

FiveK and PPR10K provide source images and expert-retouched targets without native editing instructions. The evaluation pairs are held out from all training stages by an image-level split fixed before data construction: for MIT-Adobe FiveK the expert A–E renditions of 4,500 source images form the training pool and the remaining source images are reserved for evaluation, so no evaluation image appears under any expert during training, and PPR10K uses the official test split. We annotate each before/after pair with GPT-5.6 Terra, recorded as `gpt-5.6-terra`, at temperature 0.2. The annotator generates three alternative descriptions of the edit; the reported evaluations use the fixed primary instruction, not an average or best score over the alternatives. Annotation filtering retains all 498 FiveK pairs and 492 of 500 PPR10K pairs. The eight excluded PPR10K pairs comprise one missing annotation, three parsing failures, and four instructions without a clear color-adjustment direction.

All instruction-conditioned methods receive the same retained requests and primary instructions. The after image is withheld from the retouching models at inference and used only for instruction annotation and reference scoring. Scores use a fixed expert target, so they are not directly comparable to automatic-retouching evaluations or protocols that select the best score across several experts. Means are computed over each method's valid outputs.

H MaskGrade Annotation Prompts

The following system and task prompts are transcribed from the frozen annotation configuration.
System prompt.

> You are a professional photographer and colourist annotating a training set for an instruction-driven photo-editing model. Each sample pairs a CURRENT photo with a TARGET version of the same photograph carrying a finished colour grade. Your job: study the two images and the grade's breakdown sheet, reconstruct how a photographer would think through this exact grade, and write two things , (1) a move-by-move chain of thought that walks the grade from the CURRENT photo to the TARGET look, each move stating the problem seen, the mask to work in, and the correction applied; and (2) three natural-language editing instructions, at three levels of voice, any one of which could have commissioned this edit. Your text teaches the model to read an instruction, look at a photo, and perform the grade , so the reasoning must be visually grounded in what the images actually show, voiced like a working professional, and numerically faithful to the facts sheet.
>
> You are given three images. The first is the CURRENT photo , the image to be graded. The second is the TARGET , the same photograph carrying the look the grade must reach. The third is a breakdown sheet of the grade, 2 rows by 7 columns: the top row shows the masks the grade uses, one per move, in working order; the bottom row starts with the CURRENT photo at the far left, shows the image after each successive move, and ends with the TARGET at the far right. The columns are grouped into the three stages of the grade , the global grade, the masked tonal and hue moves, and the subject move.
>
> A FACTS SHEET follows the task text. It is machine-extracted ground truth and it is authoritative: every mask, every hue band, every sign and every magnitude you state must come from it. Never invent a number, a mask family, a hue band or a direction that is not in the sheet, and never contradict it.
>
> Write the way a working photographer talks through a grade with an assistant: tonal range, colour cast, skin tones, separation, mood , never like a parameter table being read out. Be numerically precise the way a colourist is precise: the figures sit inside the sentences, they are not the sentences. Vary your sentence structure from move to move, and let what you actually see in the images choose the words. English only.

Vocabulary discipline: call the image being graded ”the current image” , never ”the starting frame”, ”the starting image”, ”the original”, ”the source photo” or anything similar. Express the goal only as intent , the look you are building , and never cite the TARGET as an existing picture: nothing may read like ”as seen in the finished image”, ”matching the final result” or ”as the target shows”. The chain of thought must read as a grade being planned from the current image alone; the TARGET and the breakdown sheet are your private reference and must never be mentioned as images the reader can see.

Task prompt.

Task: you are the photographer in charge of this grade. Lay out, move by move, how the CURRENT photo is graded to the TARGET look, then write three instructions of different lengths and voices that could each have commissioned exactly this edit.

The grade runs in three stages, in this order: first the global grade over the whole frame, then the masked tonal and hue moves, and finally the subject move. Keep this arc in mind so the six moves read as one deliberate grade, not six unrelated edits.

Return exactly {n_steps} entries in ‘cot‘, in working order, matching the GRADING STEP numbering of the facts sheet one for one (‘step‘ = that number). Each entry has three parts, and together they must read as one connected thought , problem, place, fix: - ‘observation‘: what is wrong at this point of the grade that this move addresses. Name the tonal range or region and the problem as you actually see it in the image , a cast, a muddy midtone band, flat highlights, a subject that sits apart from the scene , the way you would call it out on a calibrated monitor. Use the breakdown sheet to see what each move changed. - ‘mask‘: where the correction is applied. Name the mask family from the sheet and place it in scene terms , what content it sits over, which side or end is strong, how the falloff behaves. Give coverage or area figures ONLY for the subject mask; for every other mask stay strictly qualitative , no percentages, no fractions of the frame, no pixel distances. - ‘adjustment‘: the correction itself , the hue bands touched and the direction and size of each move, taken from the sheet, phrased the way you would brief an assistant working the tools.

Then: - ‘instruction_long‘: 60-110 words, written the way a professional photographer or colourist would brief the grade , trade vocabulary, precise about ranges, bands and intent, reads like a colourist’s brief. - ‘instruction_medium‘: 20-35 words, written the way an ordinary user of a photo-editing app would ask for this look , plain everyday words, what they see and what they want, no trade jargon, no numbers. - ‘instruction_short‘: 5-12 words, one imperative sentence. All three speak about the CURRENT photo and the look to reach , forward-looking editing language only. Never mention restoring, undoing, reverting or going back to anything, and never let any of them read like a parameter dump.

I TRAINING-DOMAIN INSTRUCTION-GUIDED EDITING

In-domain style editing / Unsplash source photographs

Input | Model output | Input (zoom) | Output (zoom) | Actual residual

1. Make it feel warmer and richer with bold red-orange tones, boost contrast for depth, and bring out the fox's fur and blanket while keeping the scene's mood intact and composition unchanged.



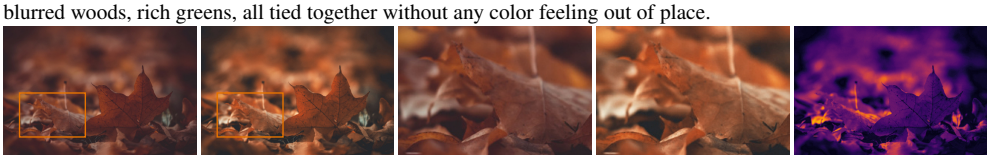
2. Make it feel crisp and cinematic with cooler tones, higher contrast, and bold highlights—especially in the shadows and sky—while pulling out the greens and deepening the blues.



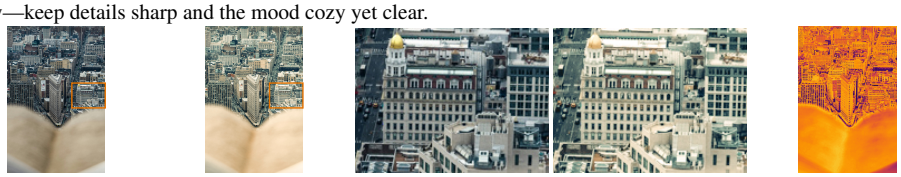
3. Make it feel cozier and brighter with richer yellows and oranges—especially in the wood, pastries, and lighting—while keeping the warmth natural and the colors vibrant but not oversaturated.



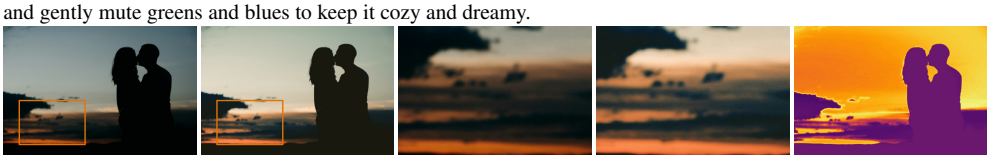
4. Make it feel warm and cinematic, brighter with strong contrast—especially in the leaves and background—softly blurred woods, rich greens, all tied together without any color feeling out of place.



5. Make the whole cityscape brighter and warmer, lifting the buildings, streets, and clouds with a soft, luminous glow—keep details sharp and the mood cozy yet clear.



6. Make the sunset brighter and warmer with soft pastel tones, keep the couple in silhouette, boost contrast subtly, and gently mute greens and blues to keep it cozy and dreamy.



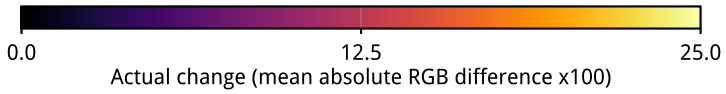
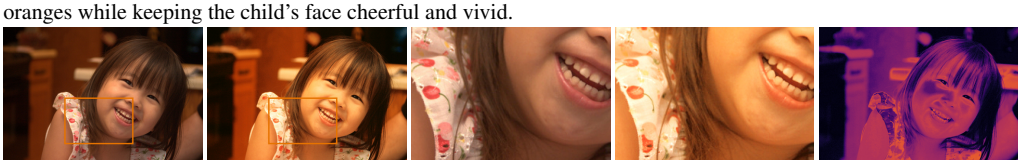


Figure 13: **Training-domain edits on Unsplash images (Unsplash, 2026).** Matched crops and output changes share one color scale per page.

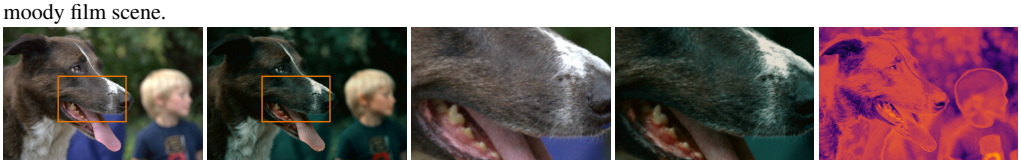
In-domain style editing / MIT-Adobe FiveK source photographs

Input | Model output | Input (zoom) | Output (zoom) | Actual residual

1. Make it feel warmer and brighter like a sunny day—boost the reds and magentas, deepen the browns, and pop the oranges while keeping the child's face cheerful and vivid.



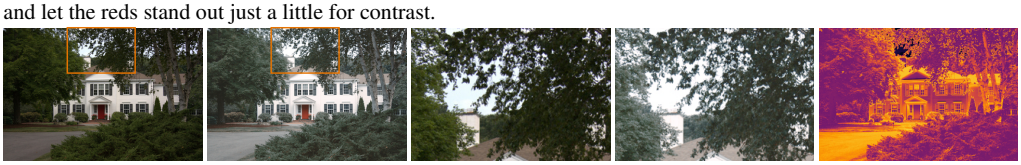
2. Make it darker and cooler overall, with deeper greens and muted purples and blues—subdued, cinematic, like a moody film scene.



3. Make it brighter and cooler overall, strip away the yellow tones from sky, water, and buildings, keep details sharp, and let the cool shadows contrast with warm skin tones for a crisp, clean look.



4. Make it bright and cool, like a soft, pale gray daylight—mute the greens but keep them visible, lift the shadows, and let the reds stand out just a little for contrast.



5. Make it feel crisp and cool—brighten everything, deepen the blues in the water, soften the warm stone, keep the gull sharp and natural, and boost the overall color pop.



6. Make it brighter and cooler, with soft, airy pastel tones—especially on the child and background—keeping it luminous but not too saturated, so it feels fresh and gently dreamy.



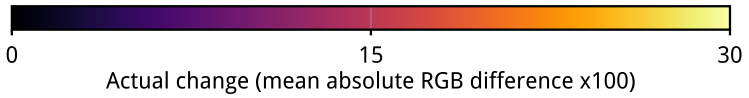


Figure 14: **Training-domain edits on FiveK images.**

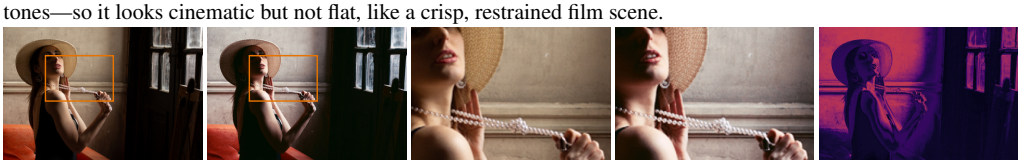
In-domain style editing / PPR10K source photographs

Input | Model output | Input (zoom) | Output (zoom) | Actual residual

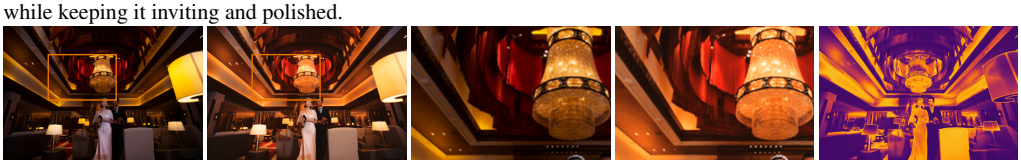
1. Make it feel like a soft, vintage film shot—gentler tones, cooler mood, faded greens, and muted contrast, especially on her face and the bouquet, while keeping everything cohesive and nostalgic.



2. I want the photo to feel cooler and sharper, with high contrast and muted colors—especially the greens and warm tones—so it looks cinematic but not flat, like a crisp, restrained film scene.



3. Make it brighter and warmer with rich reds and magentas, boost the wood tones and skin glow, soften yellow lights while keeping it inviting and polished.



4. Make it brighter and punchier with a cool, vivid green glow—especially on the foliage, umbrella, and water—while deepening contrast and pulling out details in the shadows for a crisp, lively feel.



5. Make it feel cooler and brighter with a soft pastel glow—especially on the woman, her dress, and the forest greens—while keeping everything crisp but gentle and airy.



6. Make it feel warm and nostalgic, like old film—brighten everything, soften harsh edges, and let the colors in the stairwell, railings, and clothes glow with a cozy, faded richness while keeping details sharp.



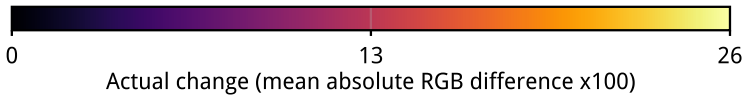


Figure 15: **Training-domain edits on PPR10K images.**

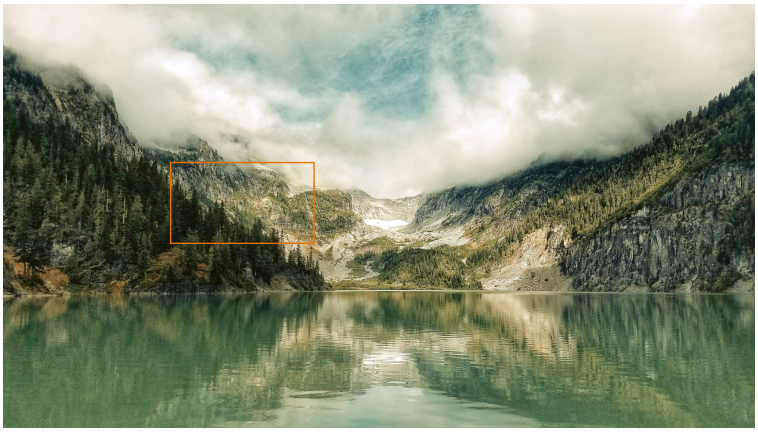
J NATIVE-RESOLUTION RETOUCHING

Native-resolution retouching: 4032×2268

Request. Give the lake a warmer, brighter late-afternoon mood, lift the dark trees gently, and preserve the cloud and mountain detail.



Input (4032×2268)



Model output (4032×2268)

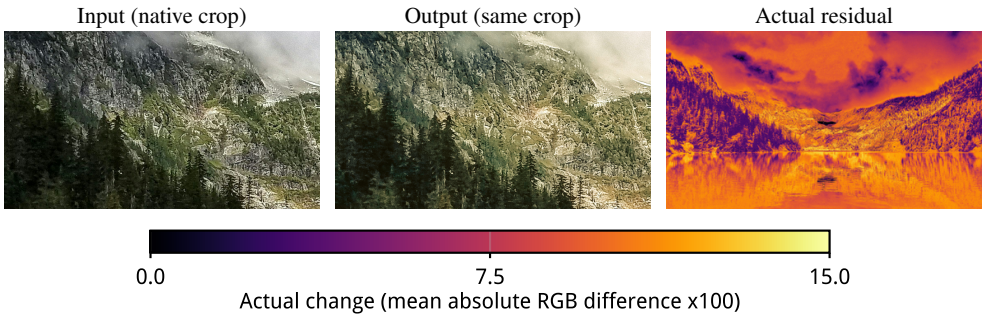


Figure 16: **Retouching at native** 4032×2268 **resolution.** Matched crops keep original pixels. Photo: Yoshi Takekawa, Unsplash.