RetouchLLM: Image Retouching via a Natural Language Bottleneck

Anonymous ACL submission

Abstract

Image retouching requires identifying subtle photometric differences and translating them into precise adjustments, mirroring how humans reason about visual style in language. We propose RetouchLLM, which treats retouching as a language-grounded reasoning task: photometric differences between source and reference images are verbalized into natural language descriptions, which are then translated into executable retouching codes. By exploring multiple candidate descriptions and selecting the best-matching result via a text-anchored style score, RetouchLLM produces plausible retouching results. The code-based design enables direct editing on high-resolution images, and the language-centric formulation ensures interpretability and naturally extends to user interaction through multi-turn dialogue. Experiments demonstrate favorable performance across eight retouching styles and a strong user preference of 70.75% in the user study.

1 Introduction

Image retouching enhances the aesthetic visual quality of photographs by correcting photographic defects such as improper exposure, poor contrast, or color imbalance. The preferred retouching style varies significantly from person to person (Ouyang et al., 2023; Hu et al., 2018), and even for the same individual, it may differ depending on the subject or scene (Wang et al., 2023). In response to this diversity, various deep learning-based approaches (Duan et al., 2025; Ouyang et al., 2023; Wang et al., 2022; Kosugi, 2024; He et al., 2020) have been proposed.

However, these approaches come with notable limitations. They require large-scale datasets of image pairs retouched in the same target style, limiting adaptability to new styles or user preferences. Their output tends to reflect an average learned from the training distribution (Wang et al., 2023), making it difficult to achieve fine-grained control tailored to individual intentions. Furthermore, these models function largely as black boxes, making it challenging to understand or intervene in the retouching process. Moreover, many of them apply edits to downscaled versions of the input image and later upscale the results, potentially degrading the original image quality.

These challenges call for an alternative paradigm: one where style understanding and editing decisions are mediated through natural language rather than learned implicitly from data. Unlike pixel-level prediction models, vision-language models (VLMs) are capable of comparing images and expressing their differences in natural language, mirroring how retouching is actually practiced, where a photographer identifies what and how much needs to change and then applies the corresponding adjustments. This positions retouching as a language-grounded reasoning task, connecting naturally to the capabilities of language-based models in structured generation and iterative planning.

Building on this intuition, we propose *RetouchLLM*, which places natural language descriptions at the center of the retouching pipeline. Photometric differences are first expressed as a natural language description and translated into executable retouching programs, making the entire process transparent and interpretable. Furthermore, this language-centric design naturally extends to user interaction, where language instructions replace reference images, enabling personalized and user-intended adjustments through multi-turn dialogue.

Our approach explores diverse retouching paths by suggesting multiple candidates and selecting one. A text-anchored selection score captures style cues from reference images despite content differences, ensuring stable convergence. RetouchLLM integrates two complementary modules: a visual critic that identifies photometric differences and provides multiple candidate descriptions, and a code generator that determines the editing sequence

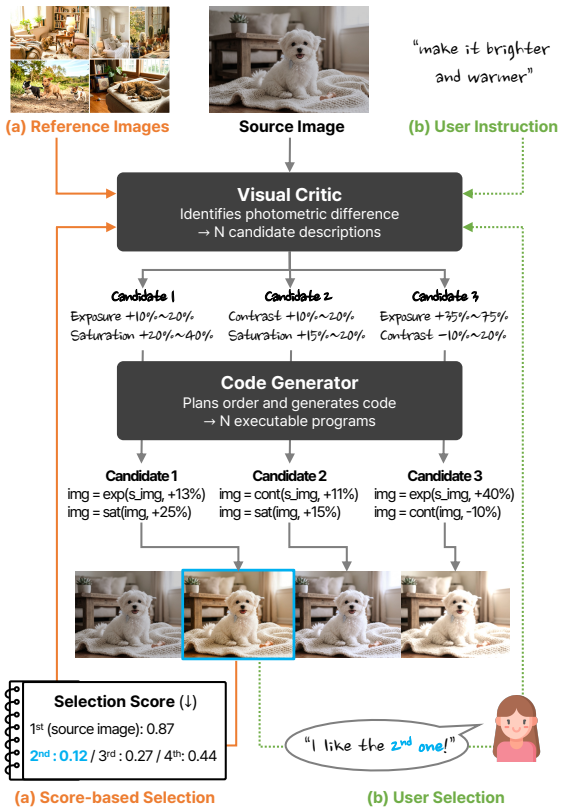

Figure 1: **Overview of RetouchLLM.** Our framework supports two modes: (a) reference-guided, where style reference images drive the retouching process, and (b) user-interactive, where natural language instructions replace reference images. In both modes, the visual critic identifies photometric differences and generates N candidate descriptions in natural language, and the code generator translates each description into an executable retouching program, producing N adjusted candidates. The best candidate is selected via a text-anchored style score in (a) or user preference in (b), set as the new source image, and the process repeats until convergence.

and produces executable retouching code. These components operate in a loop, enabling coarse-to-fine refinement across iterations.

RetouchLLM demonstrates strong performance across diverse retouching styles through qualitative and quantitative experiments. Ablation studies confirm that natural language descriptions serve as an effective semantic bottleneck, and that both the visual critic and code generator contribute to retouching quality. Furthermore, our method supports user-interactive retouching via natural language instructions, enabling flexible and personalized adjustments. A user study with 40 participants demonstrates a strong preference for our method (70.75%), confirmed by a chi-square test ($\chi^2(3) = 1381.70, p < 0.001$). We summarize our key **contributions** as follows:

- RetouchLLM formulates retouching as a language-grounded reasoning task in which language serves as the bottleneck between visual understanding and executable code generation, enabling interpretability and user interaction.
- We introduce a text-anchored style score that measures style similarity through linguistically defined concepts via text-based alignment, reliably capturing style characteristics from reference images regardless of content differences.
- Our method naturally supports interactive refinement via natural language instructions, enabling personalized retouching aligned with user intent.

2 Related Work

Automatic photo retouching. Automatic photo retouching (Yan et al., 2016; Hu et al., 2018; Ke et al., 2022; Wu et al., 2024; Yang et al., 2024b; Duan et al., 2025; Tseng et al., 2022; Kim et al., 2020; Ouyang et al., 2023) has been extensively studied to automate the retouching process by training deep models, allowing non-experts to attempt what was once an exclusive domain of professionals. These can be classified into two categories: Image-to-image translation methods and physics-based modeling methods. Translation methods train a model to generate a retouched image in the target style from an input image. Models such as U-Net (Ronneberger et al., 2015) and GANs (Goodfellow et al., 2014) are commonly used in this approach, but they often come with limitations in image resolution. These models also function as black-box, making it difficult to interpret the retouching process. Physics-based modeling treats the retouching process as a combination of actual retouching filters and reframes the problem as estimating the appropriate filter combinations and their parameters, enabling more interpretable white-box models. However, most existing methods (Dutt et al., 2025; Duan et al., 2025; Kosugi, 2024) require training a model on thousands of paired images, limiting adaptability to new styles and user preferences. Our method addresses these limitations by placing natural language at the center of the retouching pipeline, supporting interactive and interpretable retouching without training.

LLMs and VLMs for structured reasoning and generation. LLMs (Achiam et al., 2023; Chiang et al., 2023; Touvron et al., 2023a,b; Jiang et al., 2023; Abdin et al., 2024; Team et al., 2024b; Yang

et al., 2024a) and VLMs (Achiam et al., 2023; Liu et al., 2024, 2023; Team et al., 2024a; Wang et al., 2024; Hong et al., 2024) have demonstrated strong capabilities in comparing and reasoning about visual content, and generating structured outputs such as code and structured text. Building on these, prior work has addressed complex tasks, including visual programming (Shinn et al., 2024; Surís et al., 2023; Gupta and Kembhavi, 2023; Hu et al., 2024), and decision making (Zhao et al., 2024), in a structured and interpretable manner. Recent work (Dutt et al., 2025; Kosugi, 2024) has also explored applying LLMs and VLMs to image retouching, but typically lacks iterative feedback mechanisms and does not support interactive user participation, limiting flexibility and personalization. In contrast, our method leverages the comparative reasoning and code generation capabilities of LLMs and VLMs to formulate retouching as an iterative process driven by language descriptions, supporting transparent and user-interactive image retouching.

3 Method: RetouchLLM

We propose RetouchLLM, which places natural language descriptions at the center of the retouching pipeline. Photometric differences are first expressed as natural language descriptions, which are then translated into executable retouching code. The use of natural language makes the retouching process interpretable and directly extensible to user interaction, as natural language instructions can replace reference images, allowing users to guide and refine the retouching in a conversational manner. Accordingly, our framework supports two modes as illustrated in Fig. 1: (a) reference-guided, where style reference images drive the process, and (b) user-interactive, where user instructions and explicit selections take their place. The following sections describe each component in detail, with the primary focus on mode (a).

3.1 Visual Critic: Grounding Photometric Differences in Language

At each iteration, the visual critic takes the source image and reference images as input, and generates N candidate descriptions of photometric adjustments in natural language, specifying the direction and degree of change for each filter. We employ a vision-language model (VLM) for this role, as VLMs are capable of comparing images and expressing their differences in natural language. We deliberately design the description format to be linguistically grounded yet attribute-structured, as this enables the code generator to reliably reason about editing order and parameter values while retaining the semantic expressiveness needed for flexible, human-readable interaction. For example, a candidate description may include "*Exposure: The brightness of the target images is 5-10% higher than the source image.*"

However, identifying the correct adjustment amount for each photometric attribute is inherently uncertain, as VLMs may not always reliably estimate the precise degree of change in a single attempt. Sampling N candidate descriptions increases the probability that at least one falls within the correct adjustment range. Table 1 empirically supports this: with a single attempt, the correct range is identified only 53.3% of the time, while this improves to 86.3% with two candidates and 98.0% with three, matching the default setting of our method ($N = 3$). (see Appendix C.1 for details).

Table 1: **Brightness adjustment range prediction.**

Model	N=1	N=2	N=3
Random	16.7	33.3	50.0
Qwen3-VL-8B	53.3	86.3	**98.0**

3.2 Code Generator: from Description to Executable Program

Given a candidate description from the visual critic, the code generator produces an executable retouching program. A key challenge in this translation is that photometric adjustments are order-dependent, *e.g.*, applying exposure before contrast yields different results than applying contrast before exposure. The code generator explicitly plans the sequence of operations as part of the program, allowing it to adapt the ordering to each description.

The retouching program is expressed as an ordered composition of photometric operations, such as exposure, contrast, saturation, and temperature. The code generator selects a subset of operations $h_1, h_2, \ldots$ and their ordering from a pool of photometric filters: $x_t^i = (h_1 \circ h_2 \circ \cdots)(x_t^{\text{src}})$, where x_t^i denotes the i-th adjusted candidate at iteration t. Both the choice of operations and their ordering vary per description. The generated program is then executed by a Python interpreter, which applies the operations to the source image to produce the adjusted candidate x_t^i. Since each operation is applied directly to the image at its original resolution, the procedure is resolution-independent. Furthermore, unlike black-box retouching models, the generated program explicitly specifies all op-

erations and parameters, making the retouching process fully interpretable and reproducible.

3.3 Text-anchored Candidate Selection

After generating N candidates, we select the one that best matches the target style. Since the contents of the source and reference images differ, directly comparing them through image-based similarity measures such as pixel histograms or Gram matrices is unreliable. Instead, we propose a text-anchored style score that measures style similarity via filter-related text prompts implemented in the CLIP space (Radford et al., 2021), making the score content-agnostic.

Specifically, we construct K text prompts $\{z_1, \ldots, z_K\}$ as contrasting pairs for each filter (*e.g.*, "dark" vs. "bright" for exposure). Although our method employs both global and local filters, we find that restricting prompts to global filters yields better style alignment, as CLIP tends to capture global photometric characteristics more reliably, yielding $K = 8$ prompts. For each candidate image $x_t \in \mathcal{C}_t$ in iteration t, and reference image $y \in \mathcal{Y}$, we compute their logits $\ell_k(x_t)$ and $\ell_k(y)$ with respect to the K text embeddings. These logits can be converted to probabilities:

$$\begin{aligned} P_{\text{cand}} = p_k(x_t) &= \frac{\exp(\ell_k(x_t))}{\sum_{r=1}^{K} \exp(\ell_r(x_t))}, \\ q_k(y) &= \frac{\exp(\ell_k(y))}{\sum_{r=1}^{K} \exp(\ell_r(y))}. \end{aligned} \tag{1}$$

Then, we define a selection score $\sigma : \mathcal{X} \to \mathbb{R}_{\geq 0}$ as

$$\begin{aligned} \sigma(x_t, Y) &= D_{\text{KL}}(p(x_t) \,\|\, \bar{q}) \\ &= \sum_{k=1}^{K} p_k(x_t)\big(\log p_k(x_t) - \log \bar{q}_k\big). \end{aligned} \tag{2}$$

where $P_{ref} = \bar{q}_k = \frac{1}{M}\sum_{j=1}^{M} q_k(y^j)$ summarizes all of the reference images with respect to the text prompts. The candidate with the lowest score is selected as the new source x_{t+1}^{src} for the next iteration:

$$\begin{aligned} x_{t+1}^{\text{src}} &\leftarrow \mathcal{C}_t[i^*], \\ \text{where} \quad i^* &= \arg\min_{i \in \{0,\ldots,N\}} \sigma(\mathcal{C}_t[i], \mathcal{Y}), \end{aligned} \tag{3}$$

with the candidate set $\mathcal{C}_t$ for the current iteration t. We include the source image x_t^{src} in the candidate set, *i.e.*, $\mathcal{C}_t = [x_t^{src}, x_t^1, \ldots, x_t^N]$, to ensure the score is non-increasing across iterations (see Appendix B.3 for proof). While σ serves as a proxy for style alignment, the resulting improvement in perceptual quality is empirically confirmed in Fig. 2.

The iterative process in mode (a) continues until one of the stopping conditions holds: the source image is selected for three consecutive steps, the visual critic reports no significant differences across all retouching elements, or a predefined maximum iterations $T = 10$ is reached. In mode (b), the user stops the process when satisfied with the result.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate our method on two public datasets: AdobeFiveK (Bychkovsky et al., 2011) and PPR10K (Liang et al., 2021). To construct source-reference pairs, we utilize CLIP (Radford et al., 2021) to compute pairwise KL divergence across the dataset, selecting reference images based on content similarity to reflect common user behavior in real-world retouching workflows.

Competing methods. We compare against Z-STAR (Deng et al., 2024), a zero-shot style transfer model, and supervised models RSFNet (Ouyang et al., 2023) and PG-IA-NILUT (Kosugi, 2024), which are pretrained and further fine-tuned to each target style. Details of fine-tuned models are provided in Appendix C.2. For plausibility assessment, we additionally compare against Exposure (Hu et al., 2018), Unpaired (Kosugi and Yamasaki, 2020), InstructP2P (Brooks et al., 2023), MGIE (Fu et al., 2024), and MonetGPT (Dutt et al., 2025).

Implementation details. We use five reference images per sample ($M = 5$), and the visual critic generates three candidate descriptions per iteration ($N = 3$) by default. We implement the visual critic and code generator using four LLMs: Qwen2.5-VL-72B (Bai et al., 2025b), InternVL3-14B (Zhu et al., 2025), GPT-5 (OpenAI, 2025), and Gemini-1.5-Pro (Team et al., 2024a). Further implementation details are provided in Appendix C.

Metrics. Following previous work (Ke et al., 2022; Wang et al., 2022; Wu et al., 2024), we report PSNR, SSIM, LPIPS, and ΔE ($\mathcal{L}_2$ distance in the CIELAB color space). Higher PSNR and SSIM values, and lower LPIPS and ΔE values, indicate better performance.

4.2 Experimental Results

Plausibility assessment. We evaluate our method under the setting of (Dutt et al., 2025), where each input image is associated with five expert retouch-

Table 2: **Plausibility assessment.** All comparison methods are trained on PPR10K and evaluated on AdobeFiveK, except MGIE, which requires large-scale training data. Our method requires no task-specific training.

Method	PSNR(↑)	SSIM(↑)	LPIPS(↓)	ΔE(↓)
Exposure	15.12	0.63	0.14	-
Unpaired	21.73	0.83	0.12	-
RSFNet	21.85	0.88	0.08	-
InstructP2P	16.99	0.61	0.22	-
MGIE	22.94	0.74	0.08	-
MonetGPT	23.75	0.90	**0.07**	-
RetouchLLM (Ours)	**25.48**	**0.92**	0.09	**7.57**

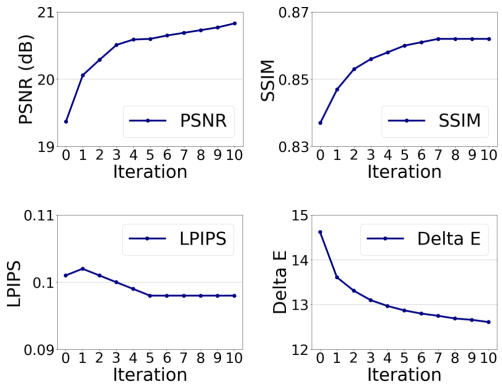

Figure 2: **Quantitative results over iterations.** All metrics improve consistently as iterations progress, indicating that RetouchLLM continues to refine the image in a style-aligned direction. Higher PSNR and SSIM, and lower LPIPS and ΔE, indicate a closer to GT.

ings and the model output is scored against the closest among them. This protocol evaluates how well a model can produce edits that fall within the range of human editing preferences. Following this protocol, we evaluate our method by randomly sampling 400 images from the 500 test images in AdobeFiveK (Bychkovsky et al., 2011) and reporting the corresponding scores in Table 2.

Our method achieves the highest PSNR and SSIM, and a competitive LPIPS. These results suggest that our outputs are closely aligned with the diversity of expert-level adjustments, demonstrating that exploring multiple candidate paths leads to retouching results that better reflect the range of human editing preferences.

Retouching quality over iterations. We evaluate the effectiveness of the iterative retouching process using the GPT-5 implementation of RetouchLLM. Figure 2 reports the metric-wise trend averaged over all eight retouching styles, showing consistent improvement across iterations. The most significant changes occur in the early iterations, while gains diminish in later steps, indicating that RetouchLLM performs retouching in a coarse-to-fine manner. The qualitative results in Figure 3 further illustrate progressive enhancement toward the target style. Notably, all retouching operations are performed directly on the original high-resolution images without any resizing, preserving fine details throughout the process. The qualitative results of high-resolution images are provided in Figs. 7, 8, and 9 of the Appendix.

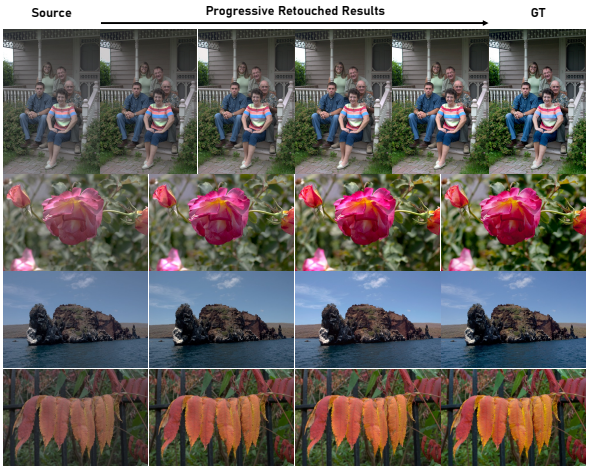

Figure 3: **Qualitative results of progressive retouching.** In each row, the leftmost image is source, the rightmost is GT, and the intermediate images show the progressively retouched results toward the target style.

Retouching performance across diverse styles. Table 3 shows that our method significantly outperforms Z-STAR in all metrics and styles. Furthermore, it outperforms fine-tuning methods across all metrics in AdobeFiveK, while achieving competitive results in PPR10K, demonstrating the adaptability of RetouchLLM across a wide range of retouching styles. This gap on PPR10K may be attributed to its portrait-centric nature (Liang et al., 2021), where supervised models can exploit person-specific patterns learned during fine-tuning. In addition, Table 4 highlights the extensibility of RetouchLLM across different model backbones spanning both proprietary and open-source models, consistently yielding superior performance over existing methods, suggesting that the effectiveness of our framework stems from its design rather than the choice of a specific model. The style-wise full results of Tables 3 and 4 are given in Appendix A.1.

Ablation and comparison of selection scores. In Table 5, we evaluate the effectiveness of the proposed selection score introduced in Sec. 3.3 against alternative methods: (1) RGB-channel histograms, (2) YUV-channel histograms, (3) Gram

Table 3: **Retouching performance across multiple styles.** Gray text indicates models that are pre-trained and further fine-tuned to the target style using reference images. RetouchLLM results are based on GPT-5.

Style	Method	AdobeFiveK				PPR10K			
		PSNR(↑)	SSIM(↑)	LPIPS(↓)	ΔE(↓)	PSNR(↑)	SSIM(↑)	LPIPS(↓)	ΔE(↓)
1	RSFNet	18.03	0.773	0.178	18.34	17.81	**0.819**	**0.106**	19.77
	PG-IA-NILUT	20.54	0.743	0.168	12.13	**19.60**	0.674	0.146	15.24
	Z-STAR	16.01	0.607	0.397	17.70	16.13	0.662	0.325	19.63
	RetouchLLM (Ours)	**21.68**	**0.900**	**0.072**	**10.89**	19.31	0.817	0.150	**14.95**
2	RSFNet	17.89	0.754	0.188	18.09	**21.93**	**0.870**	**0.079**	13.21
	PG-IA-NILUT	18.04	0.632	0.221	17.46	21.42	0.760	0.104	**11.82**
	Z-STAR	16.23	0.592	0.412	19.39	16.89	0.629	0.336	16.62
	RetouchLLM (Ours)	**21.13**	**0.867**	**0.094**	**12.20**	20.91	0.837	0.116	12.07
3	RSFNet	17.60	0.780	0.162	17.36	21.27	0.834	**0.072**	13.50
	PG-IA-NILUT	19.48	0.686	0.232	14.35	21.25	0.710	0.112	12.59
	Z-STAR	15.40	0.597	0.379	20.54	18.38	0.679	0.321	15.50
	RetouchLLM (Ours)	**21.32**	**0.897**	**0.081**	**12.17**	**21.49**	**0.853**	**0.105**	**12.27**

Table 4: **Retouching performance across multiple models.** All values are averaged over eight retouching styles: five from Adobe FiveK and three from PPR10K.

Model	PSNR(↑)	SSIM(↑)	LPIPS(↓)	ΔE(↓)
RSFNet	18.69	0.798	0.144	16.82
PG-IA-NILUT	19.73	0.692	0.173	14.05
Z-STAR	16.28	0.623	0.368	18.30
RetouchLLM (Ours)				
w/ GPT-5	20.75	0.858	0.101	12.76
w/ Gemini-1.5-Pro	20.41	0.857	0.102	13.03
w/ Qwen2.5-VL-72B	20.00	0.844	0.103	13.68
w/ InternVL3-14B	20.72	0.857	0.106	12.75

Table 5: **Effectiveness of the proposed selection score.** Results are reported using InternVL3 on FiveK style 1.

	Selection Score	PSNR(↑)	SSIM(↑)	LPIPS(↓)	ΔE(↓)
(1)	RGB hist.	21.62	0.906	0.078	11.51
(2)	YUV hist.	20.94	0.894	0.078	11.66
(3)	Gram matrix	21.49	0.906	0.072	11.15
(4)	KL CLIP all	21.93	0.899	0.072	10.58
(5)	KL CLIP global	**22.19**	**0.909**	**0.070**	**10.07**

matrix similarity commonly used in style transfer, (4) our KL CLIP score using prompts regarding all filters including local ones, and (5) our default KL CLIP score using only global filters. Details of each score are in Appendix C.3. The result shows that our proposed method (global) achieves the best performance. Notably, image-based scores (1-3) consistently underperform our approach, empirically confirming that style similarity is more faithfully captured through linguistically defined concepts than through pixel-level or feature-level statistics. This aligns with the language-centric design of RetouchLLM, where style is characterized via text prompts rather than visual signals.

Ablation of RetouchLLM modules. We design two ablation variants: (a) remove the visual critic and instead feed image-level statistics (*e.g.*, mean, standard deviation) directly to the code generator, which provides insight about how the form of image information representation affects retouching performance; (b) merge the visual critic and code generator into a single VLM that directly generates retouching code by comparing the source and reference images, without explicitly describing their differences, bypassing the natural language bottleneck entirely. We compare variants under paired and unpaired setups. The paired setup directly validates the generated code against expert retouching by serving the target image paired with the source as a reference. In the unpaired setup, which we take as the default and general setting, reference images with different content are used, requiring the model to generalize its retouching logic.

In Table 6, RetouchLLM (c) achieves the best performance across all metrics in the unpaired setup. These results demonstrate that verbalizing photometric differences into explicit language forces structured reasoning, leading to more accurate and generalizable retouching programs, and that both the natural-language bottleneck and the separation of the visual critic and the code generator contribute to retouching quality. The strong performance in the paired setup further suggests that our approach can effectively leverage prior knowledge to handle fine-grained photo retouching. In addition, Fig. 4 illustrates a practical case of the paired setup, showing that the generated program can be reused to achieve a similar style.

Ablation of design choices. We study the effect of two key design choices: the number of candidates and the number of reference images. In Table 7, as the number of candidates increases, the performance improves, which is consistent with the

Table 6: **Ablation of RetouchLLM modules.** (a) The code generator produces the retouching code based on the statistics of images instead of using a textual description. (b) The visual critic directly generates codes, bypassing the natural language bottleneck. (c) Our RetouchLLM. In the paired setup, the ground truth image is available as reference; in the unpaired setup, reference images share style but differ in content.

	Visual Critic	Code Generator	Paired				Unpaired			
			PSNR(↑)	SSIM(↑)	LPIPS(↓)	ΔE(↓)	PSNR(↑)	SSIM(↑)	LPIPS(↓)	ΔE(↓)
	Z-STAR		20.01	0.732	0.196	13.60	16.29	0.595	0.399	18.99
(a)	✗	✓	26.78	0.947	**0.050**	6.69	19.74	0.866	0.096	13.78
(b)	✓	✗	27.58	**0.959**	0.052	5.79	20.71	0.863	0.092	12.81
(c)	✓	✓	**29.21**	0.956	0.053	**5.23**	**22.19**	**0.909**	**0.070**	**10.07**

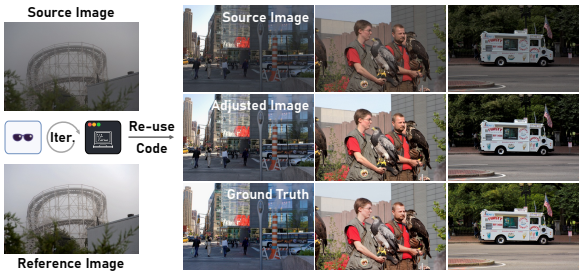

Figure 4: **Applying the extracted filter.** In the paired setup, RetouchLLM extracts a reusable retouching code that can be applied to other images as a preset filter.

Table 7: **Ablation of the number of generated candidates.** N=3 is used as default in all other experiments.

# Cand.	PSNR(↑)	SSIM(↑)	LPIPS(↓)	ΔE(↓)
1	21.27	0.889	0.078	10.91
3	22.19	0.909	0.070	10.07
5	**22.76**	**0.914**	**0.069**	**9.74**

explanation in Sec. 3.1. However, since a larger number also increases computational cost, we use three candidates in practice. Table 8 shows that increasing the number of reference images improves performance, with the best results obtained using five, as the model benefits from richer stylistic cues.

User study. We evaluate perceptual preferences across methods through a user study. We collected responses from 40 participants over 30 samples (1,200 responses in total). We ensure that the participants cover different genders, nationalities, and come from geographically different regions. More details are in Appendix C.4. The results show a strong preference for our method: NILUT: 16.42%, RSFNet: 9.67%, Z-STAR: 3.17%, and Ours: 70.75%. A chi-square test confirms that this preference is statistically significant ($\chi^2(3) = 1381.70$, $p < 0.001$), indicating that RetouchLLM consistently produces outputs that better match the target style compared to all competing methods.

Robustness to changes in reference images.

Table 8: **Ablation of the number of reference images.** M=5 is used as default in all other experiments.

# Ref.	PSNR(↑)	SSIM(↑)	LPIPS(↓)	ΔE(↓)
1	20.35	0.890	0.080	11.37
3	21.80	0.904	**0.070**	10.13
5	**22.19**	**0.909**	**0.070**	**10.07**

Table 9: **Score consistency (mean ± standard deviation) under a similar-content reference setting.**

PSNR(↑)	SSIM(↑)	LPIPS(↓)	ΔE(↓)
22.34± 1.55	0.918± 0.016	0.068± 0.011	9.44± 1.04

Users generally rely on reference images with similar content. To evaluate the score consistency under such conditions, we randomly select five images from the top ten candidates produced by CLIP-based retrieval and measure the standard deviation of their scores. This procedure is repeated seven times, and we report the trimmed mean and standard deviation by excluding the maximum and minimum values. Table 9 shows that the standard deviation remains small, indicating that the method behaves reliably when the reference images contain different but semantically similar content.

4.3 User Interaction Mode

Qualitative results. Figure 5 illustrates how RetouchLLM progressively improves image quality while reflecting the user's intended style. The model effectively composes relevant filters based on user instructions and adjusts their intensity in a controlled and interpretable manner. In the top example, in response to "*make the truck more yellowish*," the model increases both saturation and color temperature. When subsequently asked to "*give the image a slightly cooler tone*," it reduces the temperature by a relatively small amount (*e.g.*, 5%). The bottom example further demonstrates how RetouchLLM leverages interaction history to resolve ambiguous instructions. In the first itera-

Figure 5: **User interactive retouching.** The user provides natural language instructions to guide the retouching process. RetouchLLM interprets each instruction in the context of previous interactions, resolving ambiguous expressions such as "further" and "slightly" based on the history of adjustments.

tion, the model reduces saturation by 30%. When the user then asks to "*reduce the saturation further*," it interprets this as requiring a stronger adjustment and applies an 80% reduction. In the third iteration, "*slightly lower the saturation*" is interpreted as a much smaller change relative to the previous 80%, resulting in a 29% reduction. This demonstrates that RetouchLLM can resolve the ambiguity inherent in relative expressions such as "*further*" and "*slightly*" by grounding them in the history of previous interactions. More qualitative examples can be found in Figs. 15 and 16 in Appendix A.4.

Quantitative results. Since Mode (b) targets no fixed ground-truth image, pixel-level metrics such as PSNR and SSIM are not applicable; instead, we evaluate two complementary properties: *instruction consistency* (reproducibility across repeated calls) and *instruction ambiguity* (sensitivity to intensity modifiers). For *instruction consistency*, we run each (image, instruction) pair 10 times and measure the coefficient of variation (CV) of the primary filter parameter, direction-flip rate, and pairwise LPIPS and ΔE across the resulting images. The mean primary-parameter CV is 0.38, direction-flip rate is only 6%, and pairwise LPIPS stays within 0.003–0.017 and ΔE within 1.56–4.75, confirming that the model produces stable and direction-consistent retouching plans across repeated calls. For *instruction ambiguity*, we issue three modifier levels ("slightly X" / bare "X" / "much more X") as separate single-turn queries and measure how well the executed parameter magnitudes follow the expected ordering. The mean Spearman rank correlation is 0.946 and adjacent-pair accuracy is 0.80, indicating that the model reliably resolves single-turn intensity qualifiers across all aspects. Further details and analysis are provided in Appendix D.

5 Conclusion

We present RetouchLLM, which formulates image retouching as a language-grounded reasoning task, placing natural language descriptions at the center of the pipeline to bridge photometric understanding and executable code generation. Extensive experiments demonstrate that our approach produces more visually plausible results than existing methods, generalizes well across diverse styles, and supports high-resolution editing through transparent, code-based operations. Ablation studies confirm that the natural language bottleneck between the visual critic and code generator is essential to retouching quality. Furthermore, the language-centric design supports both reference-guided and user-interactive retouching, where natural language instructions enable personalized and context-aware adjustments through multi-turn interaction; quantitative evaluation confirms that the model executes instructions consistently across repeated calls and correctly interprets modifiers' intensity.

Limitations

While RetouchLLM demonstrates strong performance across diverse styles, there are several limitations to note. When reference images contain content that is highly dissimilar to the source, VLMs may struggle to consistently abstract the target style, leading to degraded performance. Additionally, the current implementation focuses on photometric operations and does not cover spatial editing such as skin smoothing, object-level adjustments, or content addition and removal. We demonstrate in Appendix A.7 that our framework can be extended to spatially constrained photometric adjustments, and leave broader spatial editing as future work. Finally, as our method relies on multiple rounds of language model inference, it is better suited for offline use cases rather than real-time applications. Runtime can be further reduced through strategies such as caching intermediate representations, adopting lighter-weight vision-language models, or parallelizing candidate generation, making the approach increasingly practical as model efficiency continues to improve.

Ethical Considerations

We design RetouchLLM to avoid content manipulation such as object addition, removal, or identity alteration, and to produce explicit, auditable code as output, thereby minimizing the risk of misuse. The user study was conducted with volunteers who provided informed consent; no personally identifiable information was collected, and diversity across gender, nationality, and geographic region was ensured among participants.

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, and 1 others. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, and 1 others. 2025a. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, and 1 others. 2025b. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*.

Tim Brooks, Aleksander Holynski, and Alexei A Efros. 2023. Instructpix2pix: Learning to follow image editing instructions. In *CVPR*.

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

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Yingying Deng, Xiangyu He, Fan Tang, and Weiming Dong. 2024. Z*: Zero-shot style transfer via attention reweighting. In *CVPR*.

Zheng-Peng Duan, Zheng Lin, Xin Jin, Dongqing Zou, Chunle Guo, Chongyi Li, and 1 others. 2025. Diffretouch: Using diffusion to retouch on the shoulder of experts. In *AAAI*.

Niladri Shekhar Dutt, Duygu Ceylan, and Niloy J Mitra. 2025. Monetgpt: Solving puzzles enhances mllms' image retouching skills. *ACM TOG*, 44(4):1–12.

Tsu-Jui Fu, Wenze Hu, Xianzhi Du, William Yang Wang, Yinfei Yang, and Zhe Gan. 2024. Guiding instruction-based image editing via multimodal large language models. In *ICLR*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *NeurIPS*.

Tanmay Gupta and Aniruddha Kembhavi. 2023. Visual programming: Compositional visual reasoning without training. In *CVPR*.

Jingwen He, Yihao Liu, Yu Qiao, and Chao Dong. 2020. Conditional sequential modulation for efficient global image retouching. In *ECCV*.

Wenyi Hong, Weihan Wang, Ming Ding, Wenmeng Yu, Qingsong Lv, Yan Wang, Yean Cheng, Shiyu Huang, Junhui Ji, Zhao Xue, and 1 others. 2024. Cogvlm2: Visual language models for image and video understanding. *arXiv preprint arXiv:2408.16500*.

Yuanming Hu, Hao He, Chenxi Xu, Baoyuan Wang, and Stephen Lin. 2018. Exposure: A white-box photo post-processing framework. *ACM TOG*.

Yushi Hu, Weijia Shi, Xingyu Fu, Dan Roth, Mari Ostendorf, Luke Zettlemoyer, Noah A Smith, and Ranjay Krishna. 2024. Visual sketchpad: Sketching as a visual chain of thought for multimodal language models. In *NeurIPS*.

AQ Jiang, A Sablayrolles, A Mensch, C Bamford, DS Chaplot, D de las Casas, F Bressand, G Lengyel, G Lample, L Saulnier, and 1 others. 2023. Mistral 7b (2023). *arXiv preprint arXiv:2310.06825*.

Zhanghan Ke, Chunyi Sun, Lei Zhu, Ke Xu, and Rynson WH Lau. 2022. Harmonizer: Learning to perform white-box image and video harmonization. In *ECCV*.

Han-Ul Kim, Young Jun Koh, and Chang-Su Kim. 2020. Pienet: Personalized image enhancement network. In *ECCV*.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, and 1 others. 2023. Segment anything. In *CVPR*.

Satoshi Kosugi. 2024. Prompt-guided image-adaptive neural implicit lookup tables for interpretable image enhancement. In *ACM MM*.

Satoshi Kosugi and Toshihiko Yamasaki. 2020. Unpaired image enhancement featuring reinforcement-learning-controlled image editing software. In *AAAI*.

Jie Liang, Hui Zeng, Miaomiao Cui, Xuansong Xie, and Lei Zhang. 2021. Ppr10k: A large-scale portrait photo retouching dataset with human-region mask and group-level consistency. In *CVPR*.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2024. Improved baselines with visual instruction tuning. In *CVPR*.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. In *NeurIPS*.

OpenAI. 2025. Gpt-5. https://openai.com/index/introducing-gpt-5/.

Wenqi Ouyang, Yi Dong, Xiaoyang Kang, Peiran Ren, Xin Xu, and Xuansong Xie. 2023. Rsfnet: A white-box image retouching approach using region-specific color filters. In *ICCV*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, and 1 others. 2021. Learning transferable visual models from natural language supervision. In *ICML*.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2024. Reflexion: Language agents with verbal reinforcement learning. In *NeurIPS*.

Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *ICLR*.

Dídac Surís, Sachit Menon, and Carl Vondrick. 2023. Vipergpt: Visual inference via python execution for reasoning. In *ICCV*.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, and 1 others. 2024a. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, and 1 others. 2024b. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Ethan Tseng, Yuxuan Zhang, Lars Jebe, Xuaner Zhang, Zhihao Xia, Yifei Fan, Felix Heide, and Jiawen Chen. 2022. Neural photo-finishing. *ACM TOG*.

Haolin Wang, Jiawei Zhang, Ming Liu, Xiaohe Wu, and Wangmeng Zuo. 2023. Learning diverse tone styles for image retouching. *IEEE TIP*.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, and 1 others. 2024. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*.

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

Jiarui Wu, Yujin Wang, Lingen Li, Fan Zhang, and Tianfan Xue. 2024. Goal conditioned reinforcement learning for photo finishing tuning. In *NeurIPS*.

Zhicheng Yan, Hao Zhang, Baoyuan Wang, Sylvain Paris, and Yizhou Yu. 2016. Automatic photo adjustment using deep neural networks. *ACM TOG*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024a. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Sidi Yang, Binxiao Huang, Mingdeng Cao, Yatai Ji, Hanzhong Guo, Ngai Wong, and Yujiu Yang. 2024b. Taming lookup tables for efficient image retouching. In *ECCV*.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024. Expel: Llm agents are experiential learners. In *AAAI*.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, and 1 others. 2025. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*.

Contents

A. Additional Results
- A.1 Extended Results across Multiple Styles
- A.2 High-resolution Retouching Results
- A.3 Applying the Extracted Filter
- A.4 User Interaction Results
- A.5 Descriptions and Codes Samples
- A.6 Failure Cases
- A.7 Applicability for Local Image Editing
- A.8 Test-time Cost

B. Details of RetouchLLM
- B.1 Algorithm Details
- B.2 Prompts
- B.3 Selection Score
- B.4 Reference Image Set Construction

C. Details of Experiments
- C.1 Brightness Range Prediction Test
- C.2 Fine-tuning Supervised Models
- C.3 Other Selection Scores
- C.4 User Study

D. Mode (b) Quantitative Evaluation
- D.1 Experimental Setup
- D.2 Instruction Consistency
- D.3 Instruction Ambiguity

In this Appendix, we include additional qualitative and quantitative results, method details, and experimental details, which are not included in the main paper.

A Additional Results

A.1 Extended Results across Multiple Styles

We complement Table 3 and Table 4 in the main paper by providing evaluations across all retouching styles in Table 10. We evaluate RetouchLLM on eight diverse retouching styles, including five from the MIT-Adobe FiveK dataset (Bychkovsky et al., 2011) and three from PPR10K (Liang et al., 2021). For each style, we randomly select 30 source images for testing, each paired with five reference images. Unlike conventional source–GT pairs that share the same content but differ in style, our setting uses source–reference pairs where the content may differ, and the goal is to adjust the source image to match the reference style. We construct these pairs using CLIP (Radford et al., 2021) by extracting image-level logits and computing pairwise KL divergence across the dataset. The list of test images will be released along with the code.

We compare our method against Z-STAR (Deng et al., 2024), RSFNet (Ouyang et al., 2023), and PG-IA-NILUT (Kosugi, 2024), where RSFNet and PG-IA-NILUT are fine-tuned for each style using the corresponding reference images. The results in Table 10 demonstrate the adaptability of RetouchLLM across a wide range of retouching styles and its extensibility to different model backbones. In addition, Fig. 6 provides a qualitative comparison, where Z-STAR produces distorted results due to its diffusion-based style transfer mechanism, while our method yields results most similar to the ground truth.

A.2 High-resolution Retouching Results

RetouchLLM performs retouching through executable code, it operates independently of image resolution. Figures 7, 8, and 9 present retouching results on high-resolution images from MIT-Adobe FiveK (Bychkovsky et al., 2011). Given five reference images, RetouchLLM extracts the target photometric style and applies the corresponding adjustments directly to the high-resolution source image. The results confirm that our method preserves fine details without degradation, demonstrating its capability for resolution-agnostic retouching.

A.3 Applying the Extracted Filter

Figure 14 presents additional examples of the paired setup introduced in Fig. 4 of the main paper. Given a source image and a corresponding target image, RetouchLLM extracts a retouching program that can be reused as a preset filter, enabling consistent style transfer across multiple photos taken under similar conditions. The extracted program is resolution-independent and can be applied to new images as long as their starting photometric conditions are comparable.

A.4 User Interaction Results

We provide additional examples of user-interactive retouching. In this mode, reference images are replaced with natural language instructions, and the user manually selects the preferred candidate at each iteration. From the second iteration onward, the visual critic receives the history of previous adjustments to better interpret ambiguous relative expressions.

In Fig. 15, the user intends to make the source image significantly brighter with a warmer tone. RetouchLLM generates three candidate images using different adjustment strategies, with exposure increased by approximately 30% to 80% and color temperature raised by 15% to 50%. After the user selects the preferred candidate, the process continues with a new instruction. For simplicity, the three candidate outputs and user selections are omitted in Fig. 5 and Fig. 16; only the final selected output per iteration is shown. In Fig. 16, the model increases temperature by 30% in the first iteration. When the user then asks to increase the contrast a lot," it applies an 80% increase. In the third iteration, make it slightly brighter" is interpreted as a much smaller adjustment relative to the previous 80% change. This demonstrates how RetouchLLM leverages interaction history to resolve ambiguous relative expressions such as a lot" and slightly."

A.5 Generated Descriptions and Codes Samples

Figures 17, 18, and 19 show examples of outputs produced by the visual critic and code generator. The visual critic generates three candidate descriptions, each specifying the photometric differences across all filters. Based on each description, the code generator produces the corresponding retouching code. Rather than adjusting all components simultaneously, RetouchLLM first applies global brightness adjustments, followed by local brightness, color tone, and texture adjustments. Since the effect of each adjustment depends on the sequence of operations, the code generator plans the editing order based on both the difference description and general editing conventions. As illustrated in Fig. 22, our method selectively applies adjustments across iterations and progressively refines the image toward the target style.

A.6 Failure Cases

While our method performs robustly in most cases, we observe several types of failure cases during the process. First, the visual critic occasionally omits the decision for further editing. If any per-filter analysis is present but the final decision omits further editing, we proceed as if additional retouching is still required. Second, the code generator may occasionally output code that is not directly executable. For example, including placeholder comments such as "`source_image = ... # assume source_img is already defined`" can lead to execution errors. In such cases, we re-query the model with a different temperature, allowing up to three attempts. If all three retries fail, the process skips the current retouching and proceeds to the

Table 10: **Retouching performance across multiple styles (full results of Table 2 in the main paper).** Gray text indicates models fine-tuned on the target style using reference images.

Style	Method	PSNR(↑)	SSIM(↑)	LPIPS(↓)	ΔE(↓)
FiveK A	RSFNet	17.55	0.747	0.204	17.34
	PG-IA-NILUT	20.00	0.713	0.189	13.46
	Z-STAR	16.61	0.622	0.383	16.77
	RetouchLLM (GPT-5)	21.24	0.860	0.086	12.43
	RetouchLLM (Gemini-1.5-Pro)	21.08	0.862	**0.082**	12.27
	RetouchLLM (Qwen2.5-VL)	20.04	0.856	0.084	12.69
	RetouchLLM (InternVL3)	**21.55**	**0.871**	0.090	**11.95**
FiveK B	RSFNet	18.03	0.773	0.178	18.34
	PG-IA-NILUT	20.54	0.743	0.168	12.13
	Z-STAR	16.01	0.607	0.397	17.70
	RetouchLLM (GPT-5)	21.68	0.900	0.072	10.89
	RetouchLLM (Gemini-1.5-Pro)	21.55	0.905	0.074	10.99
	RetouchLLM (Qwen2.5-VL)	21.39	0.889	0.075	11.42
	RetouchLLM (InternVL3)	**22.19**	**0.909**	**0.070**	**10.07**
FiveK C	RSFNet	17.89	0.754	0.188	18.09
	PG-IA-NILUT	18.04	0.632	0.221	17.46
	Z-STAR	16.23	0.592	0.412	19.39
	RetouchLLM (GPT-5)	**21.13**	0.867	0.094	**12.20**
	RetouchLLM (Gemini-1.5-Pro)	21.06	**0.872**	**0.092**	12.44
	RetouchLLM (Qwen2.5-VL)	20.24	0.843	0.101	13.95
	RetouchLLM (InternVL3)	20.69	0.871	0.097	12.46
FiveK D	RSFNet	17.40	0.765	0.160	16.97
	PG-IA-NILUT	17.68	0.615	0.224	**14.95**
	Z-STAR	14.60	0.593	0.391	20.27
	RetouchLLM (GPT-5)	**18.93**	**0.834**	**0.105**	15.08
	RetouchLLM (Gemini-1.5-Pro)	17.94	0.809	0.109	16.16
	RetouchLLM (Qwen2.5-VL)	17.33	0.818	0.112	16.32
	RetouchLLM (InternVL3)	18.30	0.812	0.114	15.95
FiveK E	RSFNet	17.60	0.780	0.162	17.36
	PG-IA-NILUT	19.28	0.687	0.222	14.79
	Z-STAR	15.40	0.597	0.379	20.54
	RetouchLLM (GPT-5)	**21.32**	**0.897**	**0.081**	12.17
	RetouchLLM (Gemini-1.5-Pro)	20.22	0.882	0.084	13.22
	RetouchLLM (Qwen2.5-VL)	19.06	0.864	0.092	14.71
	RetouchLLM (InternVL3)	21.24	**0.897**	0.089	**11.64**
PPR10K A	RSFNet	17.81	**0.819**	**0.106**	19.77
	PG-IA-NILUT	**19.60**	0.674	0.146	15.24
	Z-STAR	16.13	0.662	0.325	19.63
	RetouchLLM (GPT-5)	19.31	0.817	0.150	14.95
	RetouchLLM (Gemini-1.5-Pro)	19.62	0.828	0.144	**14.67**
	RetouchLLM (Qwen2.5-VL)	18.98	0.804	0.148	16.15
	RetouchLLM (InternVL3)	19.13	0.807	0.156	15.23
PPR10K B	RSFNet	**21.93**	**0.870**	**0.079**	13.21
	PG-IA-NILUT	21.42	0.760	0.104	**11.82**
	Z-STAR	16.89	0.629	0.336	16.62
	RetouchLLM (GPT-5)	20.91	0.837	0.116	12.07
	RetouchLLM (Gemini-1.5-Pro)	20.82	0.855	0.117	12.10
	RetouchLLM (Qwen2.5-VL)	21.17	0.845	0.113	12.26
	RetouchLLM (InternVL3)	21.19	0.842	0.119	12.03
PPR10K C	RSFNet	21.27	**0.872**	**0.072**	13.50
	PG-IA-NILUT	21.25	0.710	0.112	12.59
	Z-STAR	18.38	0.679	0.321	15.50
	RetouchLLM (GPT-5)	21.49	0.853	0.105	**12.27**
	RetouchLLM (Gemini-1.5-Pro)	20.96	0.842	0.116	12.39
	RetouchLLM (Qwen2.5-VL)	21.78	0.831	0.101	11.93
	RetouchLLM (InternVL3)	**21.50**	0.845	0.111	12.71

next iteration. In practice, these cases are rare and often resolved by the subsequent iteration. Both failure cases stem from the dependency on external VLM and LLM modules, and are expected to diminish as these models improve.

Figure 20 illustrates failure cases in the final results. In the first row, although the dark stone tomb is successfully brightened, this also leads to an over-exposed sky region. In the second row, while the sky is stylized well, the floor area becomes

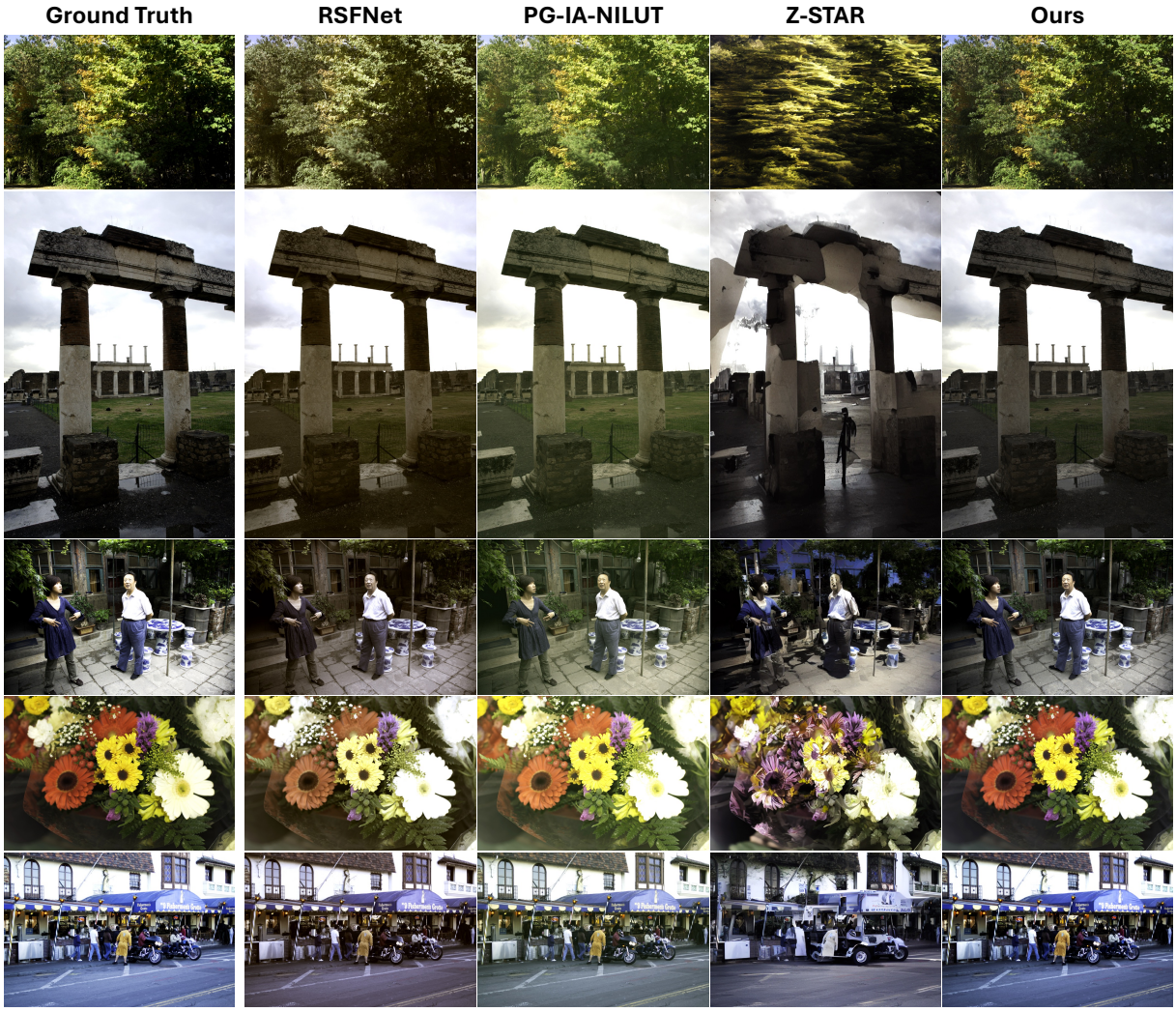

Figure 6: **Qualitative comparison results**

slightly less yellow than desired. These errors are expected to be alleviated once mask-based local editing is incorporated. The last row shows cases where the overall adjustment does not perfectly match the ground truth, resulting in either insufficient brightness or a slightly cool tone. Since our method infers the style from reference images that do not share the same content, the result may not be identical to the ground truth. Nevertheless, it still produces plausible adjustments compared to the original image.

A.7 Applicability for Local Image Editing

To evaluate whether our method can be extended toward localized retouching, we additionally incorporated local editing using segmentation-based masking. Specifically, a target region mask was obtained via SAM (Kirillov et al., 2023) and applied after the global photometric adjustment. The results in Fig. 21 demonstrate that local editing further improves similarity to the ground truth compared to global operations alone (*e.g.*, the first sample's PSNR increases from 23.79 to 29.33), confirming that our framework can be naturally extended to mask-guided local refinements.

A.8 Test-time Cost

We analyze the test-time computational cost of our method using InternVL. When generating three candidates per iteration, the average inference time without optimization is 117.31 seconds per image. By applying candidate batch processing and early stopping, we reduce the average inference time to 54.45 seconds. While the inference runtime is higher than that of conventional feed-forward retouching methods, our approach serves a fundamentally different purpose. Rather than a single forward pass, RetouchLLM performs iterative exploration in a program space, enabling personalized and interpretable retouching through multi-step re-

Algorithm 1: RetouchLLM. The algorithm iteratively adjusts the source image based on the given reference images. The process stops early if (i) the same image is selected for three consecutive steps, (ii) the generated descriptions suggest no additional edits are needed, or (iii) the algorithm reaches maximum iteration.

Require: Visual critic $f(\cdot)$, code generator $g(\cdot)$
Input: Initial source image x_0^{src}, reference set $\mathcal{Y} = \{y^j\}_{j=1}^{M}$, maximum iterations T, number of candidates N
Output: Adjusted image x^*

```
for t = 0 to T − 1 do
    Generate descriptions:
      (d_1, ..., d_N) = f(x_t^src, Y);
    for i = 1 to N do
        Generate program: g(d_i);
        Adjust source image:
          x_t^i = Execute(g(d_i), x_t^src);
    Construct candidate list:
      C_t = [ x_t^src, x_t^1, ..., x_t^N ]
    Best candidate selection:
    i* =
      arg min_{i∈{0,...,N}} SelectionScore(C_t[i], Y);
    Update source image: x_{t+1}^src ← C_t[i*];
    if stopping condition is met then
        break;
return x* = x_{t+1}^src
```

finement. In practice, the method is intended for offline or semi-interactive scenarios where flexibility and interpretability are prioritized over real-time throughput. Further runtime reductions are feasible through standard acceleration strategies, such as caching intermediate representations or adopting lighter-weight vision-language models.

B Details of RetouchLLM

B.1 Algorithm Details

We summarize the full procedure of RetouchLLM in Algorithm 1. The visual critic is based on a VLM, and the code generator is based on an LLM. Our method performs iterative retouching until one of three stopping conditions is met: (1) the source image is selected as the best candidate for three consecutive iterations; (2) the visual critic reports no significant differences across all retouching elements; or (3) the maximum number of iterations T is reached.

During each iteration, the source image is included in the candidate set to ensure reversibility in case the model outputs an unsatisfactory result. In the first iteration, a rule-based adjusted image is included as a warm-start candidate. When the visual critic produces a description, we provide image-level statistics of both the source and reference images to enable fine-grained quantitative comparison. Specifically, we compute pixel-level mean, median, and standard deviation; top and bottom 10% intensities; RGB channel-wise means; Laplacian variance (for sharpness); saturation mean, standard deviation, minimum, and maximum; and the mean values of the L and b channels.

B.2 Prompts

We provide the prompts used in our method. The system prompt for the visual critic is in Fig. 10, the user prompt for the visual critic in Fig. 11, the system prompt for the code generator in Fig. 12, and the user prompt for the code generator in Fig. 13.

System Prompt for Visual Critic.

Task:
You are an advanced image analysis assistant. Multiple images will be provided along with their color statistics. The first image is the source image, and the rest of the images are the target images. The content and the photometric style of the source and target images differ. The photometric styles of all the target images are the same. Your task is to compare the source and target images in terms of the photometric style and identify how the target images differ from the source image in the specific photometric aspects: Exposure, Contrast, Highlight, Shadow, Saturation, Temperature, Texture.

Definition:
- Exposure refers to the overall brightness of the image. A higher factor results in a brighter image, while a lower factor makes the image darker.
- Contrast refers to the difference in brightness between light and dark areas of an image. A higher factor increases the difference, making the image more vivid but losing detail, while a lower factor reduces the difference, retaining more detail but making the image look softer.
- Highlight refers to the brightest areas in an image. A higher factor brightens

these regions further, which can lead to loss of detail in overexposed areas, while a lower factor reduces brightness, helping to recover details lost in the highlights.
- Shadow refers to the darkest areas in an image. A higher factor brightens these regions, revealing details hidden in underexposed areas, while a lower factor darkens the shadows, enhancing contrast and creating a more dramatic effect, which can result in a loss of detail in the darkest areas.
- Saturation refers to the intensity of colors in an image. A higher factor enhances the vibrancy of colors, making them more intense, while a lower factor reduces the intensity, eventually leading to a grayscale image, where all color is removed.
- Temperature refers to the balance between warm and cool tones in an image. A higher factor adds warmth with reddish tones, while a lower factor introduces coolness with bluish tones.
- Texture refers to the level of detail and high-frequency variations in an image, influencing its perceived sharpness and surface characteristics. A higher factor enhances fine details and edges, while a lower factor softens the image by reducing these variations.

Instructions:
1. Choose whether to increase, decrease, or maintain the factor for each aspect. If adjusting, select the appropriate adjustment range from the given options, and if maintaining, write 'N/A' for that aspect.
2. If adjustments are needed for one or more aspects, write 'go' for the Overall part, while no adjustments are needed for any aspect, write 'stop'.

Output Format:
- Exposure: [description of exposure difference, e.g., the brightness of the target image is 10-20% higher than the one of the source image. or N/A.]
- Contrast: [description of contrast difference, e.g., the contrast of the target image is 10-20% higher than the one of the source image. or N/A.]
- Highlight: [description of highlight difference, e.g., the highlight of the target image is 10-20% higher than the one of the source image. or N/A.]
- Shadow: [description of shadow difference, e.g., the shadow of the target image is 10-20% higher than the one of the source image. or N/A.]
- Saturation: [description of saturation difference, e.g., the saturation of the target image is 10-20% higher than the one of the source image. or N/A.]
- Temperature: [description of temperature difference, e.g., the temperature of the target image is 10-20% higher than the one of the source image. or N/A.]
- Texture: [description of texture difference, e.g., the texture of the target image is 10-20% higher than the one of the source image. or N/A.]
- Overall: Write 'Stop' if there is an N/A for all aspects, and 'Go' if one or more aspects have differences.

User Prompt for Visual Critic.

Task:
You should describe the similar parts between the source image and the target images and generate 3 candidate descriptions. Each candidate should include the difference of all the aspects. Compare the source image and the target images in terms of the photometric adjustments made to the image, and describe the difference in each aspect. You can choose the range from the following list: `{range_list}`%. Do not exceed the range. You can use the color statistics and the scores between the source and target image as a guide.

Color Statistics:
- Source: `{source image statistics}`.
- Targets (averaged): `{average of target images statistics}`.

Averaged scores (PSNR, SSIM, LPIPS, Delta E):
`{Scores between source and reference`

images}

Output Format:
Similar parts
[Description of the similar parts]

Candidate 1
[Description of the first candidate]

Candidate 2
[Description of the second candidate]

Candidate 3
[Description of the third candidate]

System Prompt for Code Generator.

Task:
You are an expert Python programmer. Your task is to generate Python code that sets the appropriate filters and parameter values based on the given photometric aspect-wise description of the color tone difference between the source image and the target image, and arranges the sequence of those steps to make the source image resemble the target image.

Based on the given description, choose one of the following three options and proceed with the corresponding photometric adjustments:
- Global Brightness Adjustment (exposure, contrast): If global brightness adjustments are needed more than 1%, focus on modifying elements that affect overall brightness. Do not adjust local brightness, color tone, and texture elements at this stage, only global brightness-related factors.
- Local Brightness Adjustment (highlight, shadow): If the global brightness adjustments are completed with less than 1% differences, focus on modifying elements that affect local brightness. Do not adjust global brightness, color tone, and texture elements at this stage, only local brightness-related factors.
- Color Tone and Texture Adjustment (saturation, temperature, texture): If both the global and local brightness adjustments are completed with less than 1% differences, focus on modifying elements that affect color tone and texture. Do not adjust global brightness and local brightness elements at this stage, only color tone and texture-related factors.

Code Generation Instructions

Instructions:
1. Examine the given photometric difference description to determine which option to choose, and select only one option from the three options. Ensure that no other options are executed in the code.
2. Select the appropriate filters for the selected adjustment option, and arrange filters in the correct order.
3. The filter parameters can be chosen randomly within the range specified in the description.
4. The variable name of the adjusted image is "{save_adj_img_name}".

Difference Description:
{Difference description from Visual Critic}.

Available Functions:
- "filter.exposure(f_exp: float) -> np.ndarray": Adjusts the exposure (overall brightness) of an image. The f_exp parameter is an exposure adjustment factor, ranging from -1 to 1. The positive f_exp values brighten the overall image, while negative values darken it.
- "filter.contrast(f_cont: float) -> np.ndarray": Adjusts the contrast of an image by scaling its pixel values relative to the mean brightness of the image. The f_cont parameter is a contrast adjustment factor, ranging from -1 to 1. Positive f_cont values increase the contrast, making the image more vivid but potentially losing detail in bright and dark areas, while negative values reduce the contrast, retaining more detail but making the image look softer.
- "filter.highlight(f_high: float) -> np.ndarray": Adjusts the brightness of the bright areas of an image. The f_high

parameter is a highlight adjustment factor, ranging from -1 to 1. The positive f_high values intensify the highlights, and negative values reduce them to recover details.
- "filter.shadow(f_shad: float) -> np.ndarray": Adjusts the brightness of the dark areas of an image. The f_shad parameter is a shadow adjustment factor, ranging from -1 to 1. The positive f_shad values brighten the shadows and negative values deepen them.
- "filter.saturation(f_sat: float) -> np.ndarray": Adjusts the saturation of an image. The f_sat parameter is a saturation adjustment factor, ranging from -1 to 1. The positive f_sat values increase color vibrancy, while negative values desaturate the image towards grayscale.
- "filter.temperature(f_temp: float) -> np.ndarray": Adjusts the color temperature of an image by modifying the balance between warm and cool tones in the RGB color space. The f_temp parameter is a temperature adjustment factor, ranging from -1 to 1. The positive f_temp values shift colors toward warmer tones by increasing red, while negative values shift colors toward cooler tones by enhancing blue.
- "filter.texture(f_text: float) -> np.ndarray": Adjusts the texture of an image by modifying its high-frequency details using Gaussian blur. The f_text parameter is a texture adjustment parameter, ranging from -1 to 1. The positive f_text values enhance texture by amplifying high-frequency details, while negative values soften texture.

Please return the code directly without any imports or additional explanations.
Ensure the code is clear, correct, and follows the steps logically.

B.3 Selection Score

At each iteration, we generate three candidate images and select one as the source image for the next iteration. For the selection process, we employ a CLIP (Radford et al., 2021)-based scoring method. Specifically, we compute the probabilities of alignment between each image in the candidate set and the reference image with respect to eight textual prompts from four global filters: "`a dark light photo`" and "`a bright light photo`" from the exposure, "`a low-contrast photo`" and "`a high-contrast photo`" from the contrast, "`a desaturated colours photo`" and "`a vivid colours photo`" from the saturation, and "`a cool-toned photo`" and "`a warm-toned photo`" from the temperature. We then compute the KL Divergence between the probability distribution of each candidate image and that of the reference image. The candidate with the lowest score is selected. When multiple reference images are provided, we average their distributions before computing the divergence.

Convergence. By always including the current source image in the candidate list, the process ensures that the selected image cannot be strictly worse than in the previous iteration. Moreover, if the same source image is selected for three consecutive iterations, we deem the process saturated and terminate the optimization early.

Formally, let $\sigma_t = \sigma(x_t, \mathcal{Y})$ denote the selection score at iteration t.

$$\sigma_{t+1} = \sigma(x_{t+1}, \mathcal{Y}) \;\leq\; \sigma(x_t, \mathcal{Y}), \qquad (4)$$

showing that the sequence (σ_t) is nonincreasing. Since

$$\sigma_t = D_{\mathrm{KL}}(p(x_t) \,\|\, \bar{q}) \;\geq\; 0, \qquad (5)$$

the sequence (σ_t) is bounded below by zero. By the properties of bounded monotone sequences, (σ_t) therefore converges to some finite limit. Although this convergence does not imply global optimality (since the candidate set $\mathcal{C}_t$ may only partially cover the search space), we observe that further improvements beyond 10 iterations are small. We therefore fix the iteration budget to $T = 10$. Note that σ is a proxy for style alignment and does not directly measure ground-truth quality. Nevertheless, Figure 2 in the main paper empirically shows that standard image quality metrics (PSNR, SSIM, LPIPS, ΔE) improve consistently alongside the selection score across iterations, supporting the effectiveness of this proxy in guiding the retouching process toward higher-quality results.

B.4 Reference Image Set Construction

We construct source–reference image pairs that simulate realistic user behavior in image retouching. In practice, users often choose reference images with semantically similar content. To mimic this behavior, we use CLIP (Radford et al., 2021) to extract

image-level embeddings and compute pairwise KL divergence across the dataset. Reference images are selected based on similarity in the embedding space. Examples of such source–reference pairs are shown in Fig. 23.

C Details of Experiments

Both AdobeFiveK (Bychkovsky et al., 2011) and PPR10K (Liang et al., 2021) are publicly available datasets released for research purposes, and we use them accordingly. Our released code and test-image list are intended for research use only, consistent with the original access conditions of these datasets.

C.1 Brightness Range Prediction Test

In Table 1 of the main paper, Qwen3-VL-8B (Bai et al., 2025a) is evaluated on its ability to infer the brightness adjustment range given a pair of images: the original and a manually brightened version. The task is framed as a classification problem over six predefined discrete intervals: (0–5), (5–10), (10–20), (20–40), (40–60), and (60–100). Given this setup, the accuracy of random guessing is approximately $\frac{N}{6} \times 100\%$ for each N. The prompts used are provided below, with separate variants for each value of N.

Prompts for range prediction test (N=1)

System Prompt: You are an image comparison model. Given two images, determine the brightness difference between them and choose the appropriate difference range from the following list: [(0,5), (5,10), (10,20), (20,40), (40,60), (60,100)]. For example, if the brightness difference is approximately 15%, respond with "(10,20)". Do not provide any additional explanations or details.
User Prompt: Choose the appropriate brightness difference range between the two images. The pixel means of the first image is `{The mean pixel value of the original image}` and the second image is `{The mean pixel value of the manually brightened image}`.

Prompts for range prediction test (N=2)

System Prompt: You are an image comparison model. Given two images, determine the brightness difference between them and choose the two most appropriate difference ranges from the following list: [(0,5), (5,10), (10,20), (20,40), (40,60), (60,100)]. For example, if the brightness difference is approximately 15%, respond with "(10,20), (20,40)". Do not provide any additional explanations or details.
User Prompt: Choose the two most appropriate brightness difference ranges between the two images. The pixel means of the first image is `{The mean pixel value of the original image}` and the second image is `{The mean pixel value of the manually brightened image}`.

Prompts for range prediction test (N=3)

System Prompt: You are an image comparison model. Given two images, determine the brightness difference between them and choose the three most appropriate difference ranges from the following list: [(0,5), (5,10), (10,20), (20,40), (40,60), (60,100)]. For example, if the brightness difference is approximately 15%, respond with "(10,20), (20,40), (40,60)". Do not provide any additional explanations or details.
User Prompt: Choose the three most appropriate brightness difference ranges between the two images. The pixel means of the first image is `{The mean pixel value of the original image}` and the second image is `{The mean pixel value of the manually brightened image}`.

C.2 Fine-tuning Supervised Models

To provide a fair and comprehensive comparison, we fine-tune the RSFNet (Ouyang et al., 2023) and PG-IA-NILUT (Kosugi, 2024) models on the same set of reference images used in our method.

Importantly, the supervised baselines are not trained from scratch using only five examples. Instead, they are fully trained in the standard manner on other styles, and the five target-style images are used solely for adaptation. For the MIT-Adobe FiveK evaluation, we initialize the base-

lines from weights trained on the PPRK10K dataset and adapt them using the five reference images from the target style of MIT-Adobe FiveK; conversely, for the PPRK10K evaluation, we start from weights trained on MIT-Adobe FiveK and adapt using the five reference images from the target style of PPRK10K.

We use the training parameters provided by the authors' code. For PG-IA-NILUT this involves three training stages, and for RSFNet a single stage. Due to the smaller training set, we determined the number of fine-tuning iterations through experimentation to avoid overfitting: 100 iterations for RSFNet and 200 for PG-IA-NILUT. The results are shown in Table 10.

C.3 Other Selection Scores

In Table 5 in the main text, we compare our selection score with alternative methods: (1) RGB-channel histograms, (2) YUV-channel histograms, (3) Gram matrix similarity commonly used in style transfer, (4) our KL CLIP score using prompts regarding all filters including local ones, and (5) our default KL CLIP score using only global filters.

For (1) RGB-channel and (2) YUV-channel histogram-based scores, we compute the channel-wise histogram matching loss L between the images from the candidate set $x^i \in \mathcal{C}$ and the reference images $\mathcal{Y} = \{y^j\}_{j=1}^M$, and add all of them to get the distance

$$\begin{aligned} D_i^{RGB} &= \frac{1}{M}\sum_{j=1}^{M}\left(L_{ij}^R + L_{ij}^G + L_{ij}^B\right), \\ D_i^{YUV} &= \frac{1}{M}\sum_{j=1}^{M}\left(L_{ij}^Y + L_{ij}^{UV}\right). \end{aligned} \tag{6}$$

For (3) gram matrix similarity, for each layer $\ell \in \mathcal{L}$, where $\mathcal{L}$ is a set of layers in VGG (Simonyan and Zisserman, 2015), the feature map of image I is

$$\begin{aligned} F_\ell(I) &\in \mathbb{R}^{C_\ell \times (H_\ell W_\ell)}, \\ G_\ell(I) &= \frac{1}{C_\ell H_\ell W_\ell} F_\ell(I) F_\ell(I)^\top. \end{aligned} \tag{7}$$

The Gram-based style distance between source x_i and target y_j is

$$D_i^{\text{Gram}} = \frac{1}{M}\sum_{j=1}^{M}\sum_{\ell\in\mathcal{L}} \left\|G_\ell(x^i) - G_\ell(y^j)\right\|_F^2. \tag{8}$$

Finally, the candidate with the smallest distance is selected:

$$i^* = \arg\min_i D_i. \tag{9}$$

For (4), we additionally incorporate six prompts corresponding to three local filters, *e.g.*, highlight, shadow, and texture, increasing K from 8 to 14. The remaining equations follow Sec. B.3.

C.4 User Study

In each question, participants were shown five reference style images and four retouched results produced by different methods, presented in randomly shuffled order to avoid positional bias. A ground-truth retouched image was additionally provided as a style reference only, not as an answer option. Participants were asked: "Which image best reflects the retouching style of the given reference images?" Each participant selected the single best result from the four options.

D Mode (b) Quantitative Evaluation

D.1 Experimental Setup

We conduct a controlled quantitative evaluation of Mode (b) instruction-following across two properties: *instruction consistency* and *instruction ambiguity*. Experiments use InternVL3-14B as the underlying VLM, evaluated on 15 test images (sampled from AdobeFiveK) across four photometric aspects, *i.e.*, brightness, contrast, saturation, and warmth, each tested in both positive and negative directions.

Instruction consistency. Each {image, instruction} pair is executed 10 times under stochastic decoding (T=0.7) to measure reproducibility. We report the coefficient of variation (CV) of the primary filter parameter across runs, the direction-flip rate (fraction of runs where the inferred adjustment direction disagrees with the majority), and pairwise perceptual distances LPIPS and ΔE computed across the 10 resulting images.

Instruction ambiguity. We design two scenarios. *Scenario A* (single-turn modifier swap): three modifier levels, *i.e.*,"slightly X," bare "X," and "much more X," are issued in separate single-turn queries; the expected parameter magnitude ranks are 1, 2, 3, respectively. We report Spearman ρ, Kendall τ, and adjacent-pair accuracy (fraction of consecutive modifier pairs where the weaker modifier yields a strictly smaller parameter change). *Scenario B* (iterative refinement): Turn 1 applies the base instruction; Turns 2 and 3 issue "a bit more" / "even further" on the previous result. *Per-turn* ρ measures rank correlation between turn index and the magnitude of the incremental change applied in

Table 11: **Mode (b) instruction consistency.** Coefficient of variation (CV) is computed on the primary filter parameter across 10 runs; flip rate counts the fraction of runs where the inferred direction disagrees with the per-pair majority; LPIPS and ΔE measure pairwise image-level variability.

Aspect	Dir	CV (cand 0)	CV (all 3)	Flip rate	LPIPS	ΔE
brightness	neg	0.32	0.53	0.03	0.014	4.75
brightness	pos	0.31	0.51	0.03	0.017	4.20
contrast	neg	0.41	0.59	0.07	0.003	1.56
contrast	pos	0.30	0.51	0.03	0.010	3.19
saturation	neg	0.48	0.57	0.21	0.007	2.60
saturation	pos	0.30	0.53	0.01	0.011	3.02
warmth	neg	0.48	0.58	0.05	0.010	2.86
warmth	pos	0.44	0.52	0.04	0.009	2.71
mean		0.38	0.54	0.06	0.009	3.11

that turn (expected to shrink each turn); *cumulative* ρ measures rank correlation between turn index and the total change accumulated from the source image (expected to grow monotonically toward the commanded direction).

D.2 Instruction Consistency

Table 11 reports per-aspect consistency results. Overall, the mean primary-parameter CV is 0.38, the direction-flip rate is 6%, and pairwise LPIPS stays in the 0.003–0.017 range and pairwise ΔE is in 1.56–4.75, confirming that RetouchLLM Mode (b) produces stable, direction-consistent retouching plans across repeated calls.

D.3 Instruction Ambiguity

Scenario A: single-turn modifier swap. Table 12 reports per-aspect results. The mean Spearman ρ is 0.946, and the mean adjacent-pair accuracy is 0.80, indicating that the model reliably interprets single-turn qualifiers in the expected intensity order across all four aspects and directions.

Scenario B: iterative refinement. Table 13 reports per-turn and cumulative rank correlations.

Positive directions (brighter, more contrasty, more saturated, warmer): the cumulative change grows monotonically toward the commanded direction (mean cumulative $\rho = +0.939$, adj-pair acc $= 0.95$), confirming that the iterative chain accumulates correctly across all three turns.

Negative directions (darker, less contrasty, less saturated, cooler): the cumulative ρ is -0.866 but adj-pair acc drops to 0.08, indicating that the chain reverses after Turn 1. Inspection of per-turn parameter values reveals the underlying cause: when a follow-up phrase such as "a bit more" or "even further" is issued after an image has been pushed into a darker or cooler state, the underlying VLM defaults to interpreting the underspecified modifier in the canonical positive direction of the aspect, effectively undoing the previous adjustment. This is a semantic ambiguity failure mode of the VLM rather than a pipeline issue; multi-turn prompts that explicitly carry the intended direction, or VLMs fine-tuned on negation-consistent iterative instruction data, would be needed to resolve it.

Table 12: **Instruction ambiguity – Scenario A: single-turn modifier swap.** Three modifier levels "slightly X" / bare "X" / "much more X" are tested with expected magnitude ranks 1 / 2 / 3. Spearman ρ and Kendall τ measure rank correlation with the expected order; adj-pair accuracy is the fraction of consecutive modifier pairs where the weaker qualifier produces a strictly smaller parameter change.

Aspect	Dir	Spearman ρ	Kendall τ	Adj-pair acc	Off-target
brightness	neg	1.000	1.000	1.00	0.059
brightness	pos	0.982	0.976	0.93	0.084
contrast	neg	0.866	0.816	0.50	0.000
contrast	pos	1.000	1.000	1.00	0.000
saturation	neg	0.866	0.816	0.50	0.000
saturation	pos	0.937	0.914	0.77	0.000
warmth	neg	0.964	0.951	0.87	0.000
warmth	pos	0.955	0.939	0.83	0.000
mean		0.946	0.927	0.80	0.018

Table 13: **Instruction ambiguity – Scenario B: iterative refinement (3 turns).** Per-turn ρ: rank correlation between turn index and per-turn incremental change magnitude (expected to shrink each turn, i.e., negative ρ). Cumulative ρ: rank correlation between turn index and total accumulated change from source (expected to grow monotonically in the commanded direction).

Aspect	Dir	Per-turn ρ	Per-turn acc	Cum. ρ	Cum. acc
brightness	neg	-0.842	0.03	-1.000	0.00
brightness	pos	-0.751	0.00	$+1.000$	1.00
contrast	neg	-0.866	0.00	-0.933	0.07
contrast	pos	-0.866	0.00	$+0.933$	0.93
saturation	neg	-0.866	0.00	-1.000	0.00
saturation	pos	-0.693	0.00	$+1.000$	1.00
warmth	neg	-0.904	0.03	-0.533	0.27
warmth	pos	-0.562	0.20	$+0.824$	0.87
mean (positive dirs.)		-0.718	0.05	$+0.939$	0.95
mean (negative dirs.)		-0.869	0.02	-0.866	0.08

Figure 7: **High resolution qualitative samples 1**

Figure 8: **High resolution qualitative samples 2**

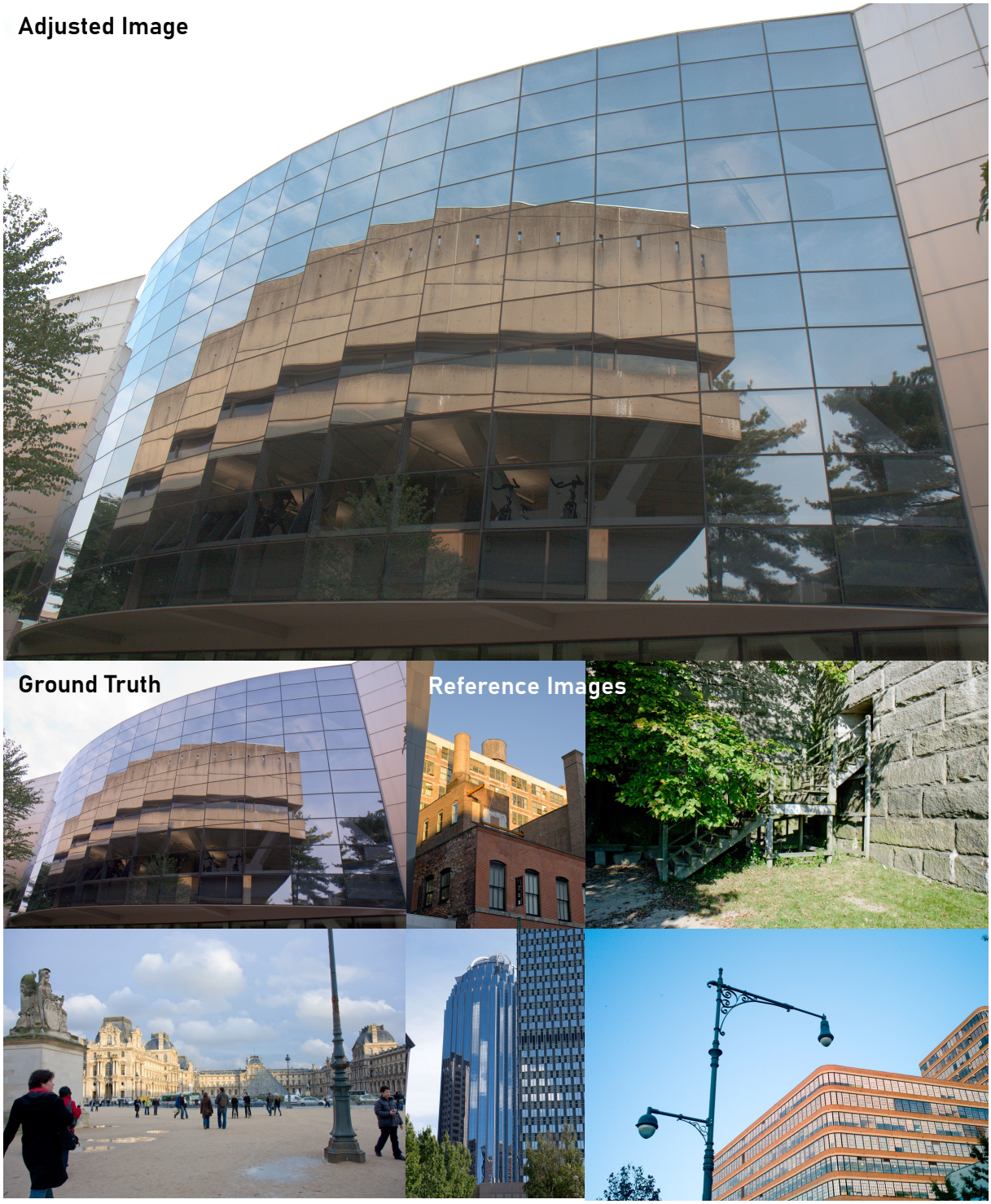

Figure 9: **High resolution qualitative samples 3**

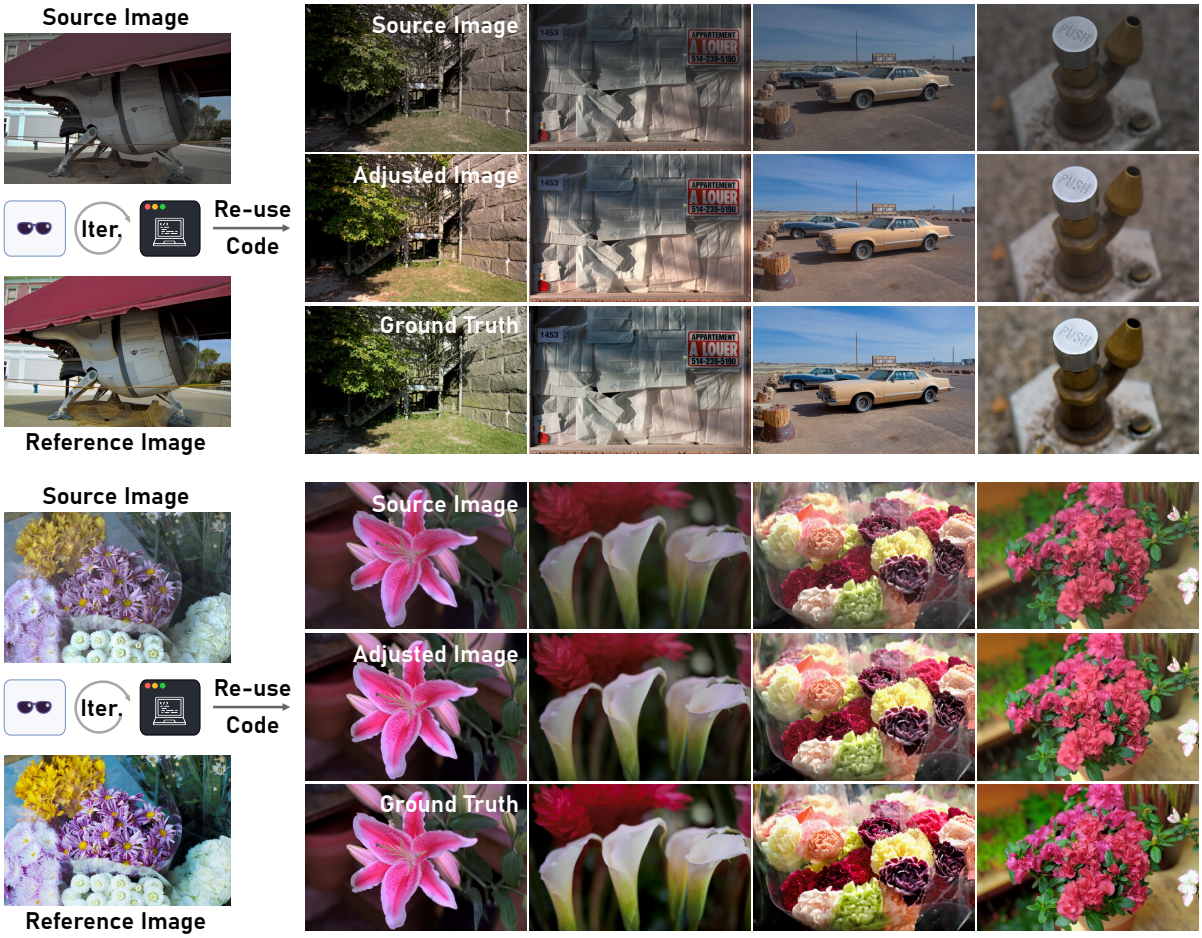

Figure 14: **Additional qualitative results for applying the extracted filter, corresponding to Fig. 4 in the main paper.** The paired setup enables extracting a reusable retouching program that can be applied to other images as a preset filter. The code extracted from the left image pair is reused to retouch other images, achieving a consistent style similar to the ground truth.

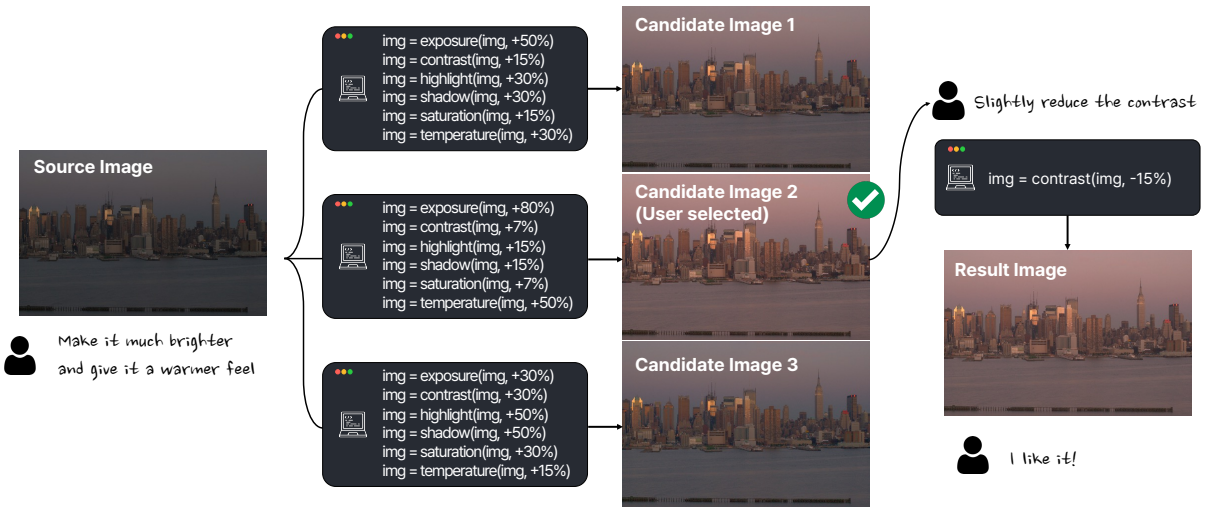

Figure 15: **Qualitative process of user-interactive retouching, corresponding to Fig. 5 in the main paper.** The user provides natural language instructions to guide the retouching process and selects a preferred result among the adjusted candidates. At each iteration, RetouchLLM generates three candidates, and the user selects one to proceed.

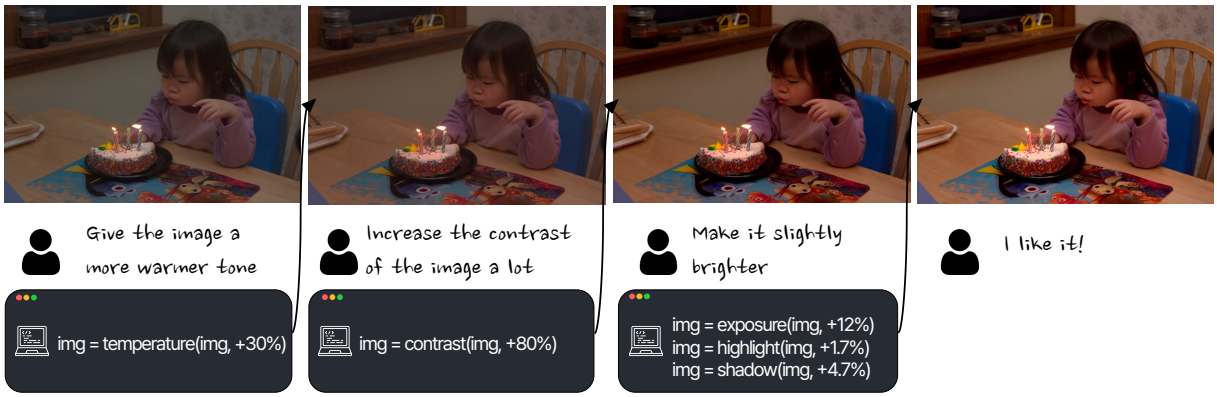

Figure 16: **Additional examples of user interactive retouching, corresponding to Fig. 5 in the main paper.** The user provides natural language instructions to guide the retouching process, and RetouchLLM interprets each instruction in the context of previous interactions to produce context-aware adjustments.

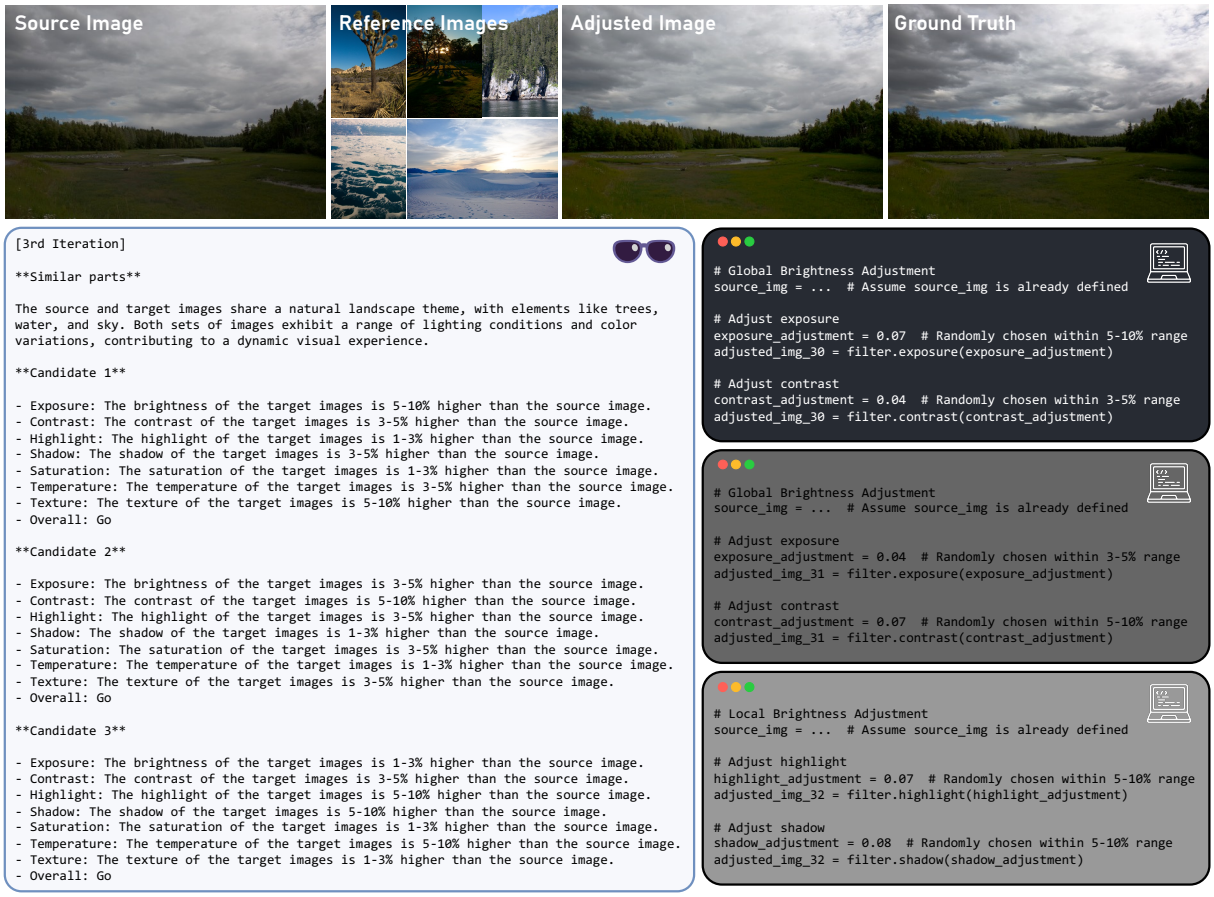

Figure 17: **Generated descriptions and corresponding codes samples 1**

```
[2nd Iteration]

**Similar parts**

The source and target images share a similar range in saturation, with both having a saturation max of
255.00 and a saturation min of 0.00. This indicates that both sets of images have areas of full color
intensity and areas with no color, maintaining a similar color range.

**Candidate 1**

- Exposure: The brightness of the target images is 5-10% higher than the source image.
- Contrast: The contrast of the target images is 5-10% higher than the source image.
- Highlight: The highlight of the target images is 5-10% higher than the source image.
- Shadow: The shadow of the target images is 5-10% higher than the source image.
- Saturation: The saturation of the target images is 1-3% higher than the source image.
- Temperature: The temperature of the target images is 5-10% lower than the source image.
- Texture: The texture of the target images is 1-3% lower than the source image.

**Candidate 2**

- Exposure: The brightness of the target images is 3-5% higher than the source image.
- Contrast: The contrast of the target images is 3-5% higher than the source image.
- Highlight: The highlight of the target images is 3-5% higher than the source image.
- Shadow: The shadow of the target images is 3-5% higher than the source image.
- Saturation: The saturation of the target images is 3-5% higher than the source image.
- Temperature: The temperature of the target images is 3-5% lower than the source image.
- Texture: The texture of the target images is 3-5% lower than the source image.

**Candidate 3**

- Exposure: The brightness of the target images is 1-3% higher than the source image.
- Contrast: The contrast of the target images is 1-3% higher than the source image.
- Highlight: The highlight of the target images is 1-3% higher than the source image.
- Shadow: The shadow of the target images is 1-3% higher than the source image.
- Saturation: The saturation of the target images is 5-10% higher than the source image.
- Temperature: The temperature of the target images is 1-3% lower than the source image.
- Texture: The texture of the target images is 5-10% lower than the source image.
```

```
# Global Brightness Adjustment
adjusted_img_20 = filter.exposure(0.07)
adjusted_img_20 = filter.contrast(0.06)
```

```
# Global Brightness Adjustment
adjusted_img_21 = filter.exposure(0.04)
adjusted_img_21 = filter.contrast(0.03)
```

```
# Color Tone and Texture Adjustment
adjusted_img_22 = filter.saturation(0.07)
adjusted_img_22 = filter.temperature(-0.02)
adjusted_img_22 = filter.texture(-0.08)
```

Figure 18: **Generated descriptions and corresponding codes samples 2**

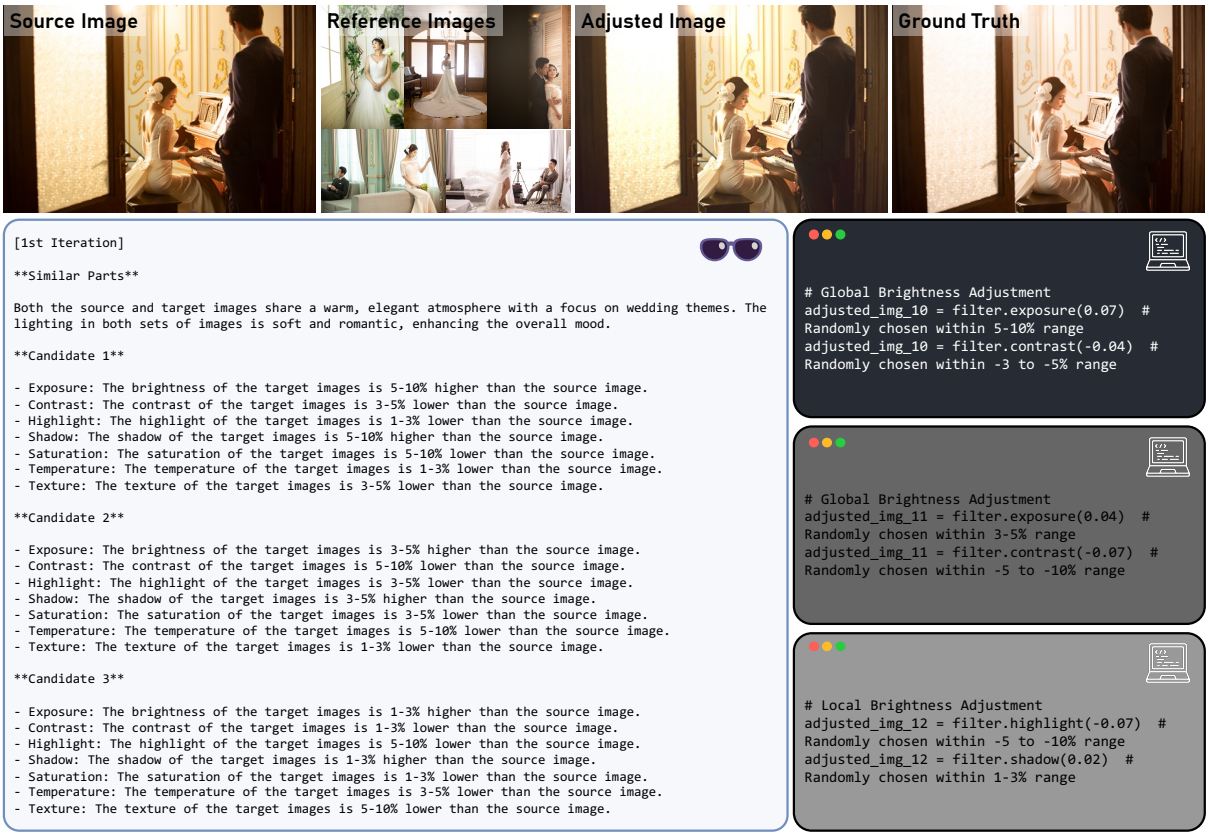

Figure 19: **Generated descriptions and corresponding codes samples 3**

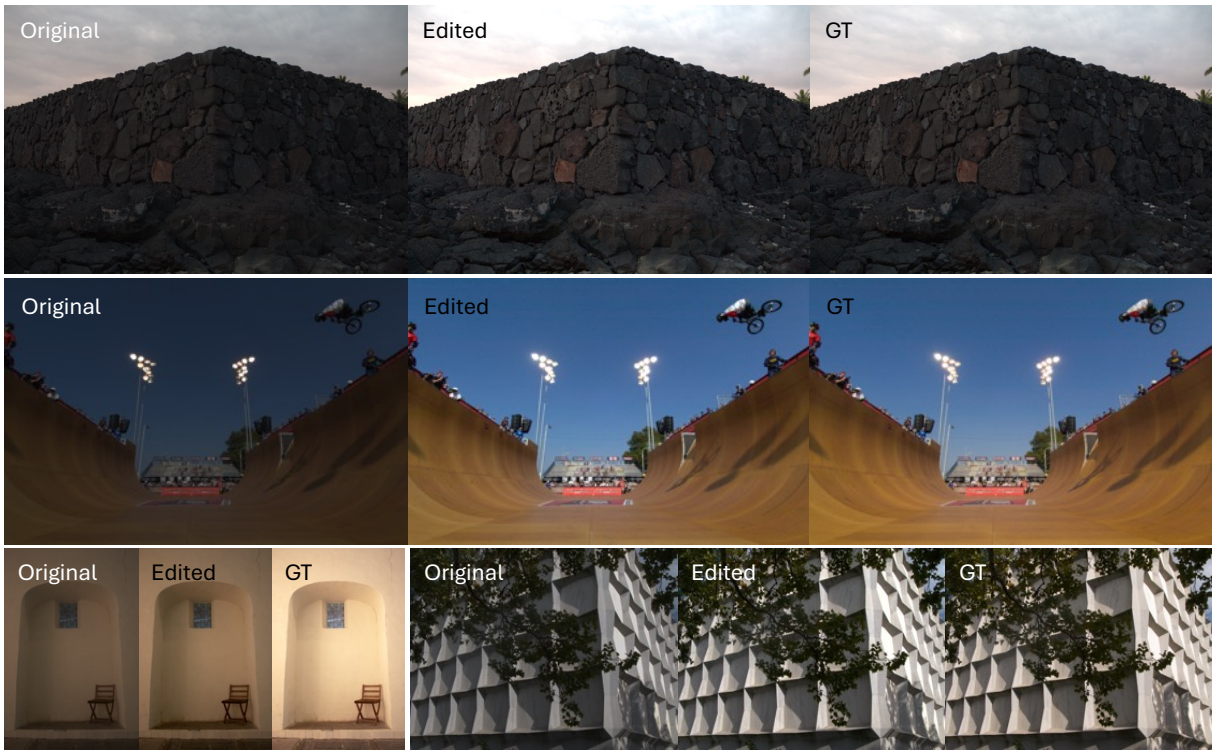

Figure 20: **Failure cases.** (Top) the sky area is overexposed. (Middle) the floor area looks less yellow. (Bottom left) the overall exposure is a bit low. (Bottom right) the colors lean slightly cool.

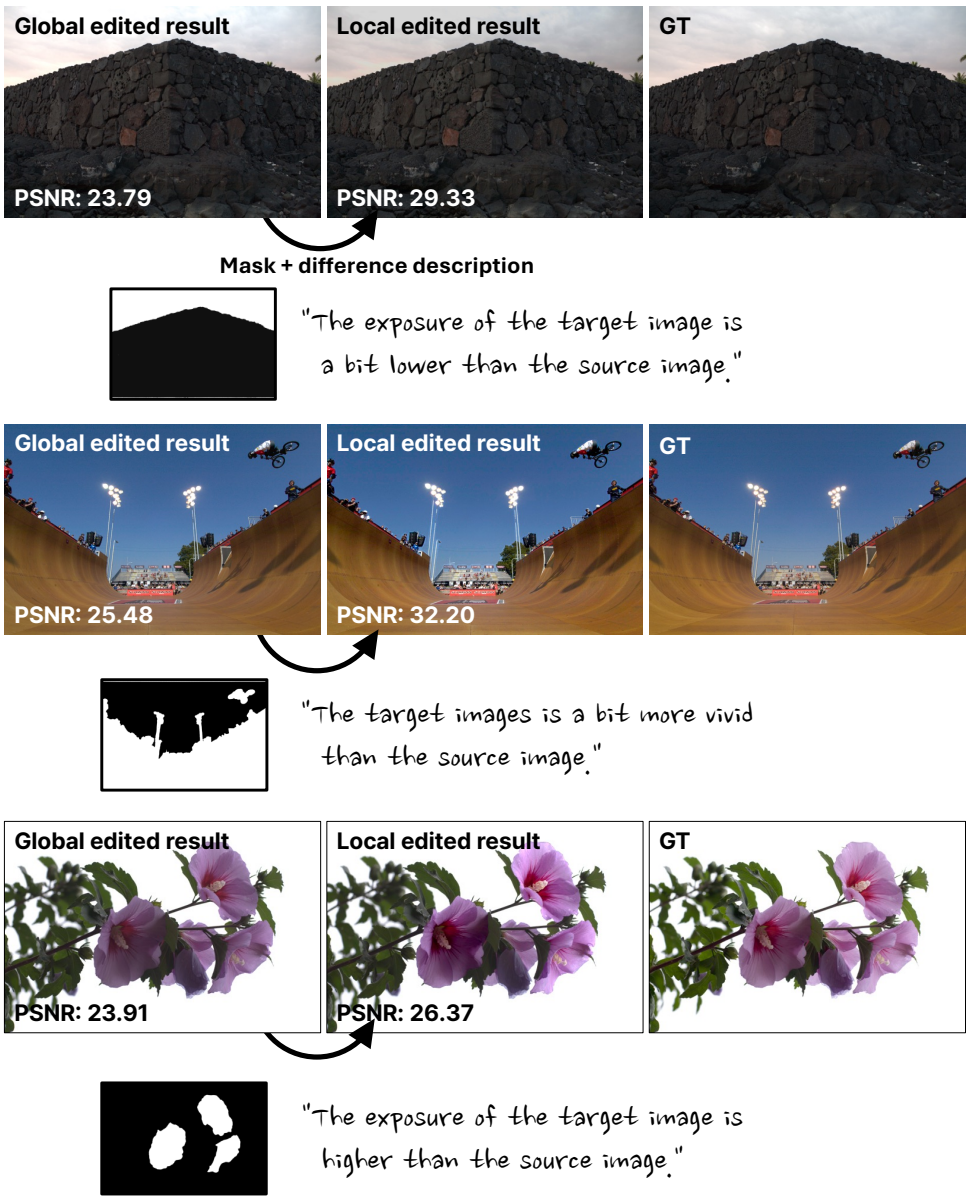

Figure 21: **Local image editing.** Local editing with a segmentation mask further improves similarity to the ground truth compared to global operations alone, demonstrating the possibility of extending our framework toward localized retouching. Note that the local region represented with white on the mask is edited, and the mask is predicted by SAM (Kirillov et al., 2023).

Figure 22: **Qualitative results of generated code.** The code generator plans the editing order by starting with global adjustments before proceeding to finer details.

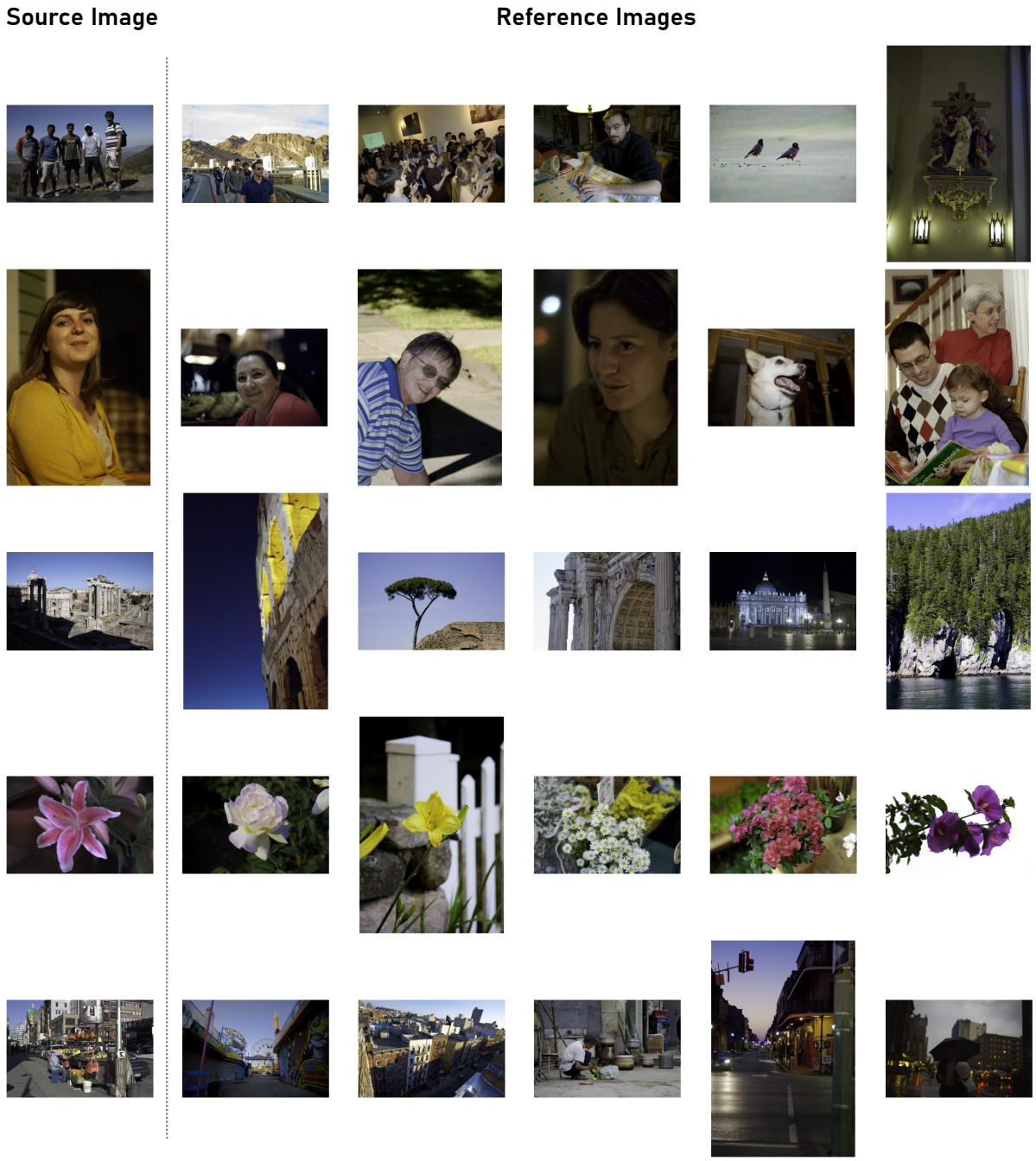

Figure 23: **Constructed reference image set examples from MIT-Adobe FiveK.**