InstantRetouch: Efficient and High-Fidelity Instruction-Guided Image Retouching with Bilateral Space

Supplementary Material

Our supplementary material is organized as follows:

- **Sec. A: Datasets.** We provide more details of our proposed *iRetouch benchmark*, and explain the construction of our *training dataset*.
- **Sec. B: Fine-grained Regional Control.** We introduce an extension to our framework that enables *fine-grained regional control* over edits by leveraging bilateral grid blending.
- **Sec. C: Additional Experimental Results.** We present additional results, including evaluation on the *MIT-Adobe FiveK* dataset, expanded comparisons with more baselines and metrics on iRetouch, and additional ablation studies.
- **Sec. D: Additional Method Details.** We offer further details on our method.
- **Sec. E: Additional Experimental Details.** We outline key experimental specifics, including details of our *evaluation metrics*, a full breakdown of our *implementation*, and our *user study*.

A. Datasets

In this section, we provide a more detailed account of the construction process for both our **iRetouch benchmark** used for evaluation and the large-scale **training dataset** used to train our model.

A.1. iRetouch Benchmark

The iRetouch benchmark is designed to provide a comprehensive and realistic testbed for instruction-guided image retouching. It consists of 500 high-resolution before–and–after image pairs, accompanied by manually refined natural language instructions. The image pairs were curated from the Adobe Lightroom community, where professional photographers share their original and edited photos. We selected this source because it represents real-world editing scenarios, high-quality artistic intent, and a wide variety of photographic genres. The selected pairs cover a broad range of subjects, including portraits, landscapes, animals, still-life, macro/close-up, and street scenes, ensuring our benchmark is not biased towards a specific domain.

For each before–and–after pair, we first generated a candidate instruction using the same procedure as our training data generation (detailed in Sec. A.2), prompting a powerful Multimodal Large Language Model (MLLM) to describe the transformation. This was followed by a manual refinement process to ensure clarity, accuracy, and naturalness.

As shown in Fig. 8, our retouching instruction covers global adjustments (brightness up/down, contrast soft/hard, temperature warm/cool/neutral, saturation up/down/local), styles (cinematic, dreamy, vintage, black-and-white, high-/low-key), moods (cheerful, cozy, serene, moody, dramatic, nostalgic, ethereal), time and season (sunset/golden hour, night, astrophotography, autumn/spring/winter), locality (global edits, subject boosting, background darkening/softening, sky/water), subjects (landscape, city, animal, bird, macro, portrait, food, astro), targeted HSL channels (green, blue, yellow, orange, magenta), and effects/intensity (glow, fog, vignette, grain; subtle/strong). Crucially, it also includes instructions for local edits (e.g., subject boosting, background darkening/softening, sky/water enhancements). This diversity allows for a more accurate and comprehensive evaluation of instruction-guided image retouching methods.

A.2. Construction of Training Dataset

Our training dataset, comprising approximately 200K triplets of $(x, x^\star, c_T)$, was constructed via a three-stage pipeline. We first curated high-quality targets $x^\star$, then synthesized degraded inputs x, and finally generated corresponding instructions c_T. While the main paper provides an overview, here we detail the latter two stages.

Synthesis of degraded inputs (x). For each target $x^\star$, we generated a degraded input x by applying a randomized photo-finishing pipeline. This "inverse editing" process includes the following stochastic photometric adjustments:

- **Exposure:** Adjusts the overall image brightness by simulating more or less light capture during exposure.
- **Gamma:** Modifies mid-tone brightness without affecting the black and white points.
- **White Balance:** Corrects color casts by adjusting temperature (cool/blue to warm/yellow) and tint (green to magenta).
- **Contrast:** Alters the tonal range by making darks darker and lights lighter.
- **Tone Curves:** Provides fine-grained control over brightness across different tonal ranges via S-shaped curves.
- **Saturation:** Controls the intensity and vividness of all colors in the image.
- **Shadows/Highlights:** Recovers detail by specifically brightening dark areas (shadows) or darkening bright areas (highlights).
- **HSL:** Selectively adjusts the hue, saturation, or brightness of each primary color range.

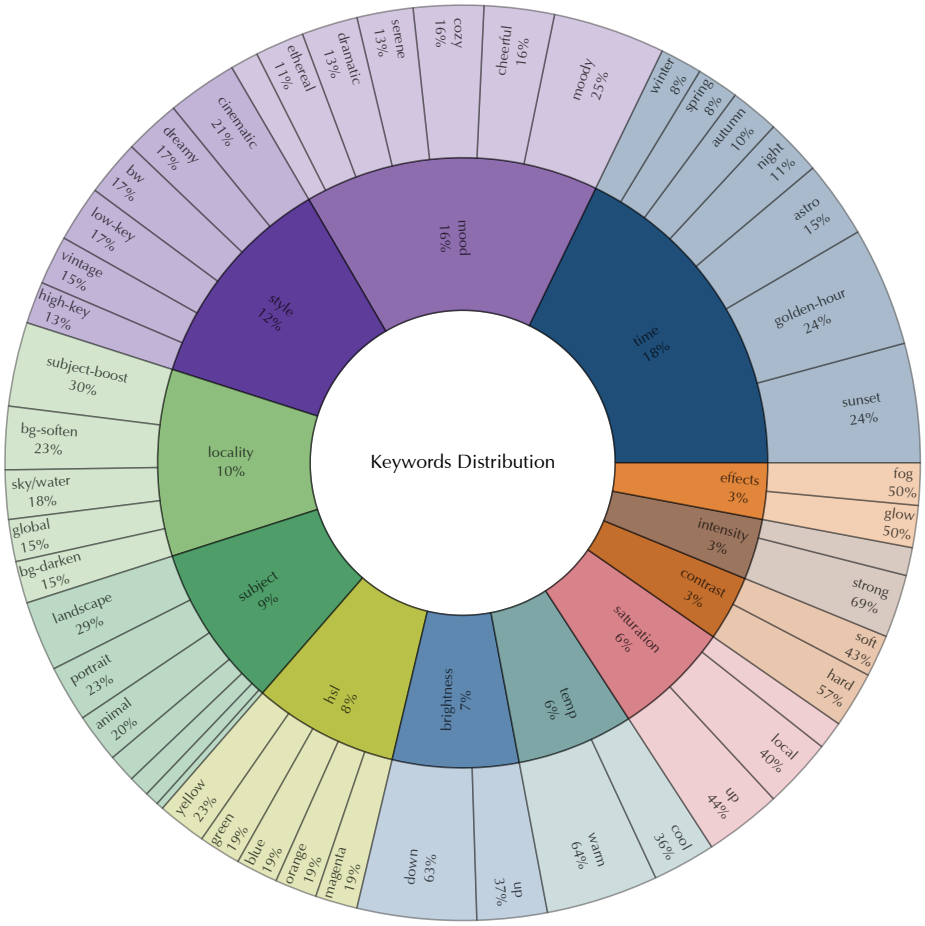

Figure 8. **Hierarchical keyword distribution of our iRetouch benchmark.** The sunburst chart illustrates the diversity and composition of the 500 instructions in our test set, spanning multiple semantic levels from high-level mood and style to specific adjustments like HSL channels and locality. This diversity ensures a comprehensive evaluation of instruction-guided retouching capabilities.

To prevent a bias towards extreme adjustments and to better model subtle, realistic edits, parameters for key operations like exposure were sampled from a Gaussian distribution centered on the no-op value (i.e., a mean of zero for additive adjustments). This ensures that a majority of our training examples represent moderate and fine-grained retouching, improving the model's generalization to real-world user requests.

For local adjustments, we generated masks from multiple sources. We used a Grounding-SAM procedure [27, 34] for semantic masks (e.g., "face", "sky"). To complement these, we also generated non-semantic, geometric masks from superpixel segmentations and simple priors like vertical/horizontal gradients. Applying different degradation parameters to this diverse set of masks allowed us to synthesize a rich variety of spatially varying edits.

Generation of instructions (c_T). With an image pair $(x, x^\star)$ prepared, we prompted the Qwen2.5-VL-72B MLLM [1] to generate the corresponding instruction c_T. The prompt, shown in Fig. 17, was carefully engineered to elicit high-quality, user-centric instructions. Specifically, it instructs the model to adopt the persona of an "intuitive photo editing assistant" and to focus on the editing *intention* rather than the technical *action*. By providing rules and examples, the prompt steers the MLLM to produce concise, casual, and actionable instructions (e.g., "give it a warm sunset vibe") while strictly avoiding technical jargon (e.g., "increase temperature"). This approach was crucial for creating a training dataset that reflects how users naturally communicate their desired aesthetic goals.

B. Fine-grained Regional Control

Our model architecture naturally supports fine-grained regional control by blending transformations directly in the bilateral grid space. This allows for complex, fine-grained

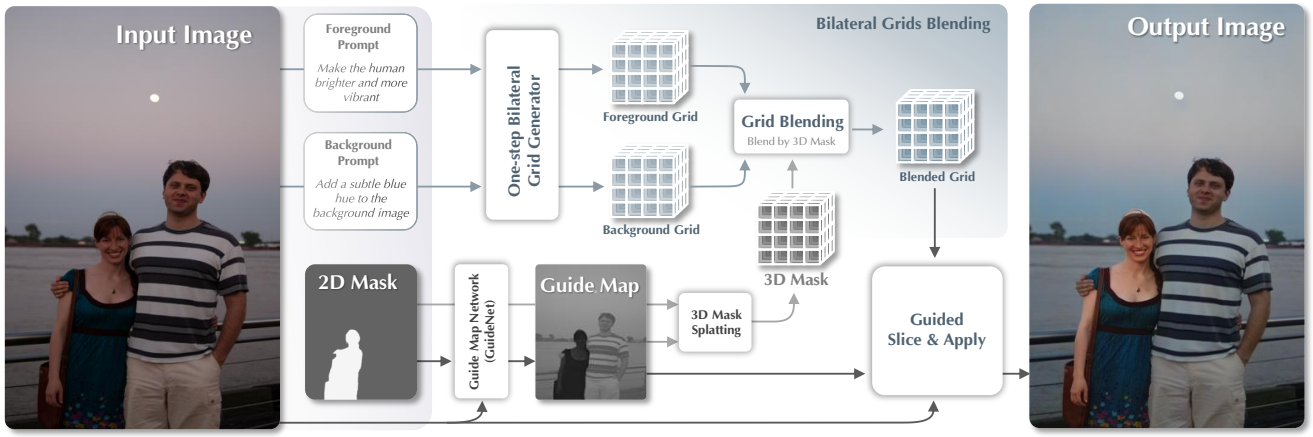

Figure 9. **An overview of our framework for fine-grained regional control.** Given an input image, a 2D mask, and distinct prompts for foreground and background, our model generates two separate bilateral grids ($\Gamma^{\text{fg}}, \Gamma^{\text{bg}}$). In parallel, a guide network produces a mask-aware guide map (z). The crucial step is lifting the 2D pixel mask (M_{pxl}) to a 3D grid mask (M_{grid}) via our proposed **3D Mask Splatting** procedure, using z for depth guidance. This enables a principled blend of the grids in coefficient space, producing a single fused grid (Γ^{fused}) that is rendered in one pass to create the final, artifact-free output.

Figure 10. **Qualitative results of fine-grained regional control through bilateral grid blending.** For each example, a user provides an input image, a 2D mask, and separate text prompts for the foreground and background. Our method successfully applies distinct and often complex edits to the specified regions. Note the seamless transitions at the mask boundaries and the high fidelity of the results, demonstrating the effectiveness of blending in the grid space.

localized edits, such as making one person brighter and keeping the other person in the dark (Fig. 10), while maintaining high visual quality and content fidelity. The core of this capability lies in our grid blending framework, which we detail below.

B.1. Framework for bilateral grids blending

Given a user-provided 2D mask $M_{\text{pxl}} \in \mathbb{R}^{H \times W \times 1}$ (which can be obtained from off-the-shelf models like SAM [34]) and distinct text prompts for the foreground and background regions, our goal is to produce a single, seamlessly edited image.

A naive solution would be to generate two separate images and alpha-blend them. However, this is inefficient as it requires two full-resolution rendering passes and often produces undesirable halo artifacts along the mask boundaries, especially when the two editing styles are significantly different.

We devise a more principled solution, outlined in Fig. 9, that interchanges the blending and rendering steps. First,

Algorithm 1 3D Mask Splatting

Require: Pixel mask M_{pxl}, guide map z, grid size (W_g, H_g, D), prior strength $\alpha \geq 0$, stabilizer $\varepsilon > 0$
Ensure: Grid mask M_{grid}

$N, C \leftarrow \text{zeros}(D, H_g, W_g)$ ▷ Value and weight accumulators
for each pixel (x, y) **do**
  $u \leftarrow x \cdot (W_g - 1)/(W - 1)$
  $v \leftarrow y \cdot (H_g - 1)/(H - 1)$
  $r \leftarrow z[y, x] \cdot (D - 1)$
  ▷ Map to continuous grid coordinate
  $\mathcal{S} \leftarrow \text{GetTriLinNeighborsAndWeights}(u, v, r)$
  ▷ Splat mask value to 8 tri-linear neighbors
  for each neighbor (d, h, w) with weight w_i in $\mathcal{S}$ **do**
    $N[d, h, w] \mathrel{+}= w_i \cdot M_{\text{pxl}}[y, x]$
    $C[d, h, w] \mathrel{+}= w_i$
  end for
end for
$M_{\text{grid}} \leftarrow \text{clip}\left(N/(C + \varepsilon), 0, 1\right)$
return M_{grid}

our model generates two distinct bilateral grids: a foreground grid $\Gamma^{\text{fg}} \in \mathbb{R}^{H_g \times W_g \times D}$ and a background grid $\Gamma^{\text{bg}} \in \mathbb{R}^{H_g \times W_g \times D}$, each corresponding to its respective prompt. In parallel, a lightweight guide network, conditioned on both the input image and the 2D mask, predicts a single, mask-aware guide map $z \in \mathbb{R}^{H \times W \times 1}$. The key challenge is then to blend Γ^{fg} and Γ^{bg} using the 2D mask M_{pxl}. To this end, we introduce a **3D Mask Splatting** procedure that lifts the 2D pixel mask $M_{\text{pxl}} \in \mathbb{R}^{H \times W \times 1}$ into a 3D grid mask $M_{\text{grid}} \in \mathbb{R}^{H_g \times W_g \times D}$, which can then be used to blend the grids directly.

B.2. 3D Mask Splatting and Grid-Level Blending

The splatting operation is the opposite of the standard bilateral grid "slicing." It ensures that the transformation respects the bilateral space structure and perfectly preserves constant-value regions.

The process, detailed in Algorithm 1, begins by mapping each pixel's 2D coordinate (x, y) and its corresponding guide value $z(x, y)$ to a continuous 3D coordinate within the grid. Instead of reading a value (as in slicing), we "splat" or distribute the pixel's mask value, $M_{\text{pxl}}(x, y)$, to the D integer grid cells surrounding the continuous coordinate, weighted by tri-linear interpolation coefficients. These weighted values are accumulated in a numerator grid N, while the weights themselves are summed into a denominator grid C. M_{grid} is then the stabilized ratio of N and C.

With the 3D grid mask constructed, we linearly blend the foreground and background coefficient grids:

$$\Gamma^{\text{fused}} = M_{\text{grid}} \odot \Gamma^{\text{fg}} + (1 - M_{\text{grid}}) \odot \Gamma^{\text{bg}}, \tag{11}$$

where $\odot$ is element-wise multiplication. The resulting Γ^{fused} is then used with the guide map z to render the final output in a single, efficient pass.

B.3. Training Objectives for Regional Control

To train the guide network and ensure mask fidelity, we introduce two additional loss terms.

Mask fidelity loss. To ensure the grid mask accurately represents the original 2D mask after slicing, we enforce a reconstruction loss. We slice the predicted 3D grid mask M_{grid} using the learned guide map z and penalize its L2 distance to the input pixel mask M_{pxl}:

$$\mathcal{L}_{\text{mask}} = \|\text{slice}(M_{\text{grid}}, z) - M_{\text{pxl}}\|_2^2. \tag{12}$$

This encourages the entire 3D mask generation pipeline to be self-consistent.

Guide map regularization. To prevent the guide network from producing arbitrary guide maps that might degrade slicing quality, we regularize the learned guide map z to stay close to the input image's luminance I_{gray}:

$$\mathcal{L}_{\text{guide}} = \|z - I_{\text{gray}}\|_2^2. \tag{13}$$

C. Additional Experimental Results

In this section, we provide more extensive quantitative and qualitative results to supplement the main paper. We first present a comprehensive comparison on our iRetouch benchmark, including more baselines, metrics, and additional visual comparison. We then report our performance on the standard MIT-Adobe FiveK dataset. We also perform additional ablation studies.

C.1. Additional Evaluation on iRetouch Benchmark

C.1.1. Additional Baselines and Metrics

To provide a more comprehensive comparison beyond the main paper, we present an expanded quantitative evaluation on our iRetouch benchmark in Tab. 4. This supplementary table introduces the following additions:

- **Additional baselines.** We include four additional baselines: (i) DiffRetouch [9], a recent state-of-the-art non-instructional retouching method; (ii) MonetGPT [59], an MLLM-based retouching agent; (iii) T2ONet [57], a text-guided photometric enhancement method; and (iv) GenColor [58], a text-guided texture-preserving retouching approach. Since their generative model is not open-sourced, we follow their paper and use nano-banana, one of the strongest baselines reported, as the text-guided generative model. Among these, DiffRetouch achieves comparable fidelity to other non-instructional methods but is

Table 4. **Comprehensive comparison on our iRetouch benchmark.** We evaluate methods across four axes: efficiency, content fidelity, editing quality, and no-reference image quality (NR-IQA). Our method achieves the best overall balance, delivering top-tier content fidelity and editing quality at real-time speed while maintaining competitive image quality scores.

Method	Efficiency	Content Fidelity				Editing Quality					Image Quality		
	runtime↓	SSIM↑	CW-SSIM↑	GSMD↓	DISTS↓	L1↓	L2↓	SC↑	PQ↑	O↑	MUSIQ↑	HyperIQA↑	NIMA↑
3DLUT [49]	0.066	0.982	**0.981**	0.013	0.024	0.136	0.034	-	-	-	59.56	0.47	5.60
RSFNet [32]	**0.029**	0.975	0.976	0.012	0.038	0.137	0.034	-	-	-	58.29	0.43	5.58
DiffRetouch [9]	1.671	0.982	0.979	0.014	0.028	0.133	0.034	-	-	-	59.35	0.44	5.55
MonetGPT [59]	62.78	0.982	0.970	0.016	0.028	0.130	0.032	-	-	-	56.97	0.40	5.54
InstructPix2Pix [2]	4.632	0.742	0.768	0.149	0.177	0.164	0.050	7.11	7.58	7.34	48.00	0.33	5.48
T2ONet [57]	0.056	0.967	0.975	0.024	0.048	0.146	0.041	6.27	8.69	7.38	52.41	0.37	5.44
GenColor [58]	22.46	0.942	0.930	0.040	0.054	0.117	0.025	7.76	8.81	8.27	59.07	0.42	5.59
Step1X-Edit [28]	57.932	0.706	0.694	0.174	0.167	0.140	0.036	7.63	8.52	8.06	**67.22**	0.50	5.90
GPT-Image-1 [17]	15.427	0.505	0.397	0.242	0.216	0.215	0.082	8.09	8.56	8.32	66.31	**0.62**	**5.91**
Qwen-Image [44]	7.720	0.689	0.744	0.174	0.147	0.168	0.054	8.12	8.67	8.39	63.57	0.52	5.78
FLUX.1-Kontext-Pro [22]	10.235	0.802	0.857	0.112	0.132	0.161	0.050	7.56	8.72	8.12	61.61	0.51	5.59
Gemini-2.5-Flash [6]	14.440	0.676	0.796	0.175	0.115	0.137	0.036	**8.56**	8.94	**8.74**	61.73	0.50	5.77
Ours	0.065	**0.989**	0.973	**0.012**	**0.022**	**0.099**	**0.018**	8.14	**8.98**	8.54	61.55	0.49	5.65

significantly slower. T2ONet exhibits poor semantic consistency (SC of 6.27), indicating limited ability to interpret diverse retouching instructions. GenColor and MonetGPT are $300-900\times$ slower than our method.

- **Additional metrics.** We report scores from a suite of common No-Reference Image Quality Assessment (NR-IQA) models, namely MUSIQ [19], HyperIQA [53], and NIMA [54]. The results of these NR-IQA metrics reveal an important discrepancy: while our method outperforms all non-instructional retouching baselines [9, 32, 49] on these scores, it is surpassed by several large generative models, such as Step1X-Edit [28], GPT-Image-1 [17], and Qwen-Image [44]. This observation motivates a deeper investigation into the validity of NR-IQA metrics for the retouching task, as they appear to favor outputs that compromise content fidelity.

C.1.2. Discussion of NR-IQA Metrics for Retouching

Our analysis reveals a critical pitfall in using standard NR-IQA metrics for evaluating image retouching: generative models can "hack" these scores by sacrificing content fidelity. This phenomenon is illustrated by the visual case study in Fig. 11.

Given the instruction to make the aurora "more vibrant for a stunning Instagram look," generative models like Step-1X and Qwen-Image achieve remarkably high MUSIQ scores (62.81 and an astonishing 70.91, respectively). However, this comes at a devastating cost to fidelity, as reflected in their extremely poor DISTS scores (0.144 and 0.265). The visual results expose the reason for this discrepancy. Step-1X invents unrealistic, oversaturated magenta hues not present in the original scene. Moreover, Qwen-Image completely misinterprets the context and hallucinates the image into a magazine layout, yielding a numerically impressive but contextually meaningless MUSIQ score. Other models like GPT-Image-1 and Gemini-2.5-Flash significantly distort the natural shape and flow of the aurora.

Figure 11. **Visual Case Study: NR-IQA (MUSIQ) vs. Content Fidelity (DISTS).** For the instruction to enhance the aurora, generative models achieve high MUSIQ scores by fabricating content (*e.g.*, Step-1X adds unnatural colors, Qwen-Image hallucinates a magazine). Our method faithfully enhances the scene, resulting in a slightly lower MUSIQ score but drastically better fidelity (low DISTS), which is the desired behavior for retouching.

In contrast, our method achieves a reasonable MUSIQ score (49.59) while maintaining exceptional fidelity (DISTS of just 0.029). Visually, our result successfully enhances the vibrancy and color of the aurora as it existed in the input, fulfilling the user's intent without fabricating content. This case study provides evidence that a high NR-IQA score in the context of generative editing often signals content alteration rather than faithful retouching. It underscores that for image retouching, fidelity-aware metrics and human judgment remain more reliable indicators of true performance.

C.1.3. Additional Visual Comparison

Fig. 15 provides further full-page visual comparisons across a diverse set of examples from our iRetouch benchmark.

Table 5. **Quantitative comparison on the MIT-Adobe FiveK dataset.** Our method significantly outperforms all baselines in content fidelity and achieves state-of-the-art editing quality, demonstrating its effectiveness on this standard benchmark.

Method	Content Fidelity				Editing Quality				
	SSIM↑	CW-SSIM↑	GSMD↓	DISTS↓	L1↓	L2↓	SC↑	PQ↑	O↑
InstructPix2Pix [2]	0.749	0.734	0.156	0.184	0.149	0.042	7.20	7.85	7.52
Step1X-Edit [28]	0.725	0.694	0.174	0.158	0.143	0.039	8.16	8.30	8.23
GPT-Image-1 [17]	0.543	0.452	0.229	0.219	0.212	0.083	**8.17**	8.60	8.38
Qwen-Image [44]	0.682	0.795	0.174	0.112	0.156	0.050	7.49	8.85	8.14
FLUX.1-Kontext-Pro [22]	0.791	0.733	0.123	0.111	0.159	0.051	7.28	8.91	8.05
Gemini-2.5-Flash [6]	0.641	0.762	0.183	0.106	0.154	0.047	7.72	8.98	8.33
Ours	**0.987**	**0.971**	**0.015**	**0.020**	**0.077**	**0.010**	7.87	**9.01**	**8.42**

These results demonstrate our model's versatility in handling a wide range of scenarios. Across all examples, our method consistently produces high-quality, faithful edits that respect the original image content, unlike generative baselines that often introduce unintended content drifts.

C.1.4. Additional Results on Prompt Controllability

Figure 12. Additional results on prompt controllability and diversity.

Fig. 12 shows that applying distinct prompts to the *same* input yields diverse, semantically consistent styles (seasonal, tonal, lighting).

C.2. Evaluation on MIT-Adobe FiveK Dataset

We further validate our method on the widely-used MIT-Adobe FiveK dataset [59], which contains 5,000 images, each retouched by five different experts (Experts A-E). For our evaluation, we use the 500-image test split and treat the edit by Expert C as the ground truth. Since this dataset lacks textual instructions, we use our instruction generation pipeline (detailed in Sec. A.2) to automatically create an instruction for each input-target pair.

C.2.1. Quantitative Comparison

As shown in Tab. 5, our method again demonstrates dominant performance. It achieves the best scores across all four content fidelity metrics by a significant margin, confirming its exceptional ability to preserve image structure and details. Furthermore, it obtains the highest scores in editing quality metrics PQ and O, and a highly competitive SC score, indicating that the edits are both aesthetically pleasing and closely aligned with the expert's intent. The low L1 and L2 scores further underscore the precision of our edits.

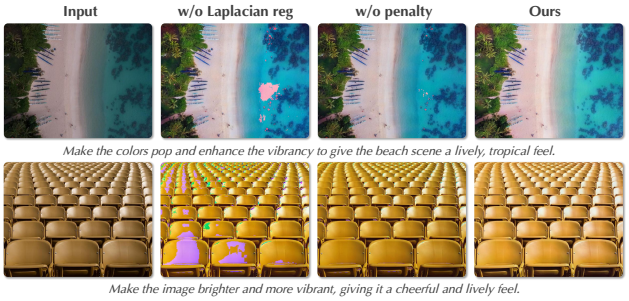

Figure 13. **Ablation study on the bilateral loss regularizers.** Removing the Laplacian regularizer introduces splotchy, spatially inconsistent artifacts. Removing the overflow penalty leads to severe color clipping and out-of-gamut colors. Our full method with both terms produces clean and high-quality results.

C.2.2. Visual Comparison

The qualitative results on MIT-FiveK, shown in Fig. 16, further highlight the superiority of our approach in delivering high-fidelity, high-quality, and accurate retouching results.

C.3. Additional Ablation Studies

To validate the design choices of our bilateral loss function, $\mathcal{L}_{\text{bila}}$ (defined in Section 3.4.4 of the main paper), we conduct an ablation study on its key regularization components. Specifically, we analyze the effects of the 3D Laplacian regularizer on the bilateral grid ($\|\Delta^3\Gamma\|_2^2$) and the RGB overflow penalty ($\Psi(\hat{x}_B)$).

As visualized in Fig. 13, both components are crucial for generating clean, artifact-free results.

- **Without the Laplacian regularizer**, the model produces severe spatial artifacts. This demonstrates that the reg-

ularizer is essential for ensuring smoothness and spatial consistency in the predicted affine transformations.

- **Without the overflow penalty**, the edits suffer from harsh color clipping. Both examples exhibit large areas of blown-out highlights and unnatural color patches. This shows the penalty's effectiveness in preventing out-of-gamut colors and maintaining a natural, photographic appearance.

Our full method, which includes both terms, produces visually pleasing results that are smooth, vibrant, and free from these distracting artifacts, confirming the necessity of our proposed bilateral loss formulation.

D. Additional Method Details

D.1. Bilateral Space

Our full-resolution bilateral processing branch is central to achieving high-fidelity, efficient editing, particularly at high resolutions. Its design is rooted in the concept of the bilateral space, which we detail here to provide further context for our methodological choices.

The concept of bilateral space was first introduced by Paris and Durand [55] for fast, edge-aware image filtering. The core idea is to lift a 2D image $I(x, y)$ into a higher-dimensional space that jointly considers spatial and range (intensity) coordinates. In this augmented space, Euclidean distances naturally correspond to perceptual similarity, making operations inherently edge-aware. This process was made highly efficient via the *bilateral grid*, which reformulates the operation into three steps: **splatting** the input image's pixels onto a coarse, regular grid; **convolving** this low-resolution grid; and **slicing** the output by trilinearly interpolating from the grid at the original pixel coordinates. Since the computationally expensive convolution is performed on a small, fixed-size grid, the algorithm's runtime is effectively decoupled from the image resolution.

This framework was later extended from simple filtering to arbitrary image-to-image transformations by storing a locally affine model in each grid cell instead of a scalar value [5, 12]. Crucially, Gharbi et al. [12] demonstrated that the slicing operation is differentiable, which allows the bilateral grid to be integrated as a trainable layer in a deep neural network. In such a framework, a neural network learns to predict the parameters of the affine grid, which then transforms the input image to produce the final output. This formulation enforces strong priors on the output; as a locally affine transformation of the input, it is constrained from hallucinating novel textures or structures, thus preserving content fidelity and preventing common artifacts like amplified noise or false edges.

Our method builds upon this foundation by replacing the simple convolutional network used in prior work [12] with an instruction-guided one-step diffusion model. The role of our diffusion U-Net is not to render the final pixels, but to leverage its powerful semantic understanding to interpret the text prompt and input image, predicting the parameters of a single, low-dimensional affine bilateral grid. This predicted grid is then applied to the original, full-resolution input via the differentiable slicing operation. This design synergistically combines the strengths of both worlds: we harness the expressive power of generative models for complex, instruction-driven edits while inheriting the fundamental properties of the bilateral grid. This ensures our edits are high-fidelity, preserve subject identity, and remain exceptionally efficient across resolutions up to 4K, overcoming the key limitations of purely generative approaches.

D.2. Lightweight Bilateral Adapter

The Lightweight Bilateral Adapter is a convolutional network tasked with predicting the affine bilateral grid from the feature maps produced by our one-step diffusion branch. The architecture is designed to be efficient while effectively processing both local and global information.

The process begins by aligning the output features from the diffusion U-Net using a sequence of four convolutional layers. The resulting features are then passed to a stride-2 convolutional encoder to reduce spatial resolution. The network then splits into two parallel paths: (i) *Local Path:* This path consists of two stride-1 convolutional layers. By preserving the spatial resolution of its input features, this path focuses on capturing fine-grained local details necessary for spatially varying adjustments. (ii) *Global Path:* This path uses another stride-2 convolutional layer followed by three fully-connected layers to distill the features into a single global scene summary vector. This vector ensures the final transformation is spatially smooth and stable. The features from both paths are then fused. The global feature vector is broadcast and added to the local feature map at every spatial location. The fused feature map is then passed through a final linear layer to generate the affine bilateral grid, which has a shape of $B \times H_g \times W_g \times D \times 12$, where (H_g, W_g, D) are the dimensions of the coarse grid and 12 represents the parameters of a 3×4 affine matrix. In our implementation, we set $H_g = W_g = 32, D = 8$.

D.3. Guided Slice and Apply

As illustrated in Fig. 14, the Guided Slice & Apply module is a fully differentiable operation that applies the predicted edit to the full-resolution image. It takes the affine bilateral grid and the original input image, producing the final retouched output. The process can be broken down into three main steps:

Guide map generation. The full-resolution input image (size $H \times W \times 3$) is first passed through a lightweight Guide Net. This network consists of three convolutional layers and outputs a full-resolution, single-channel guide

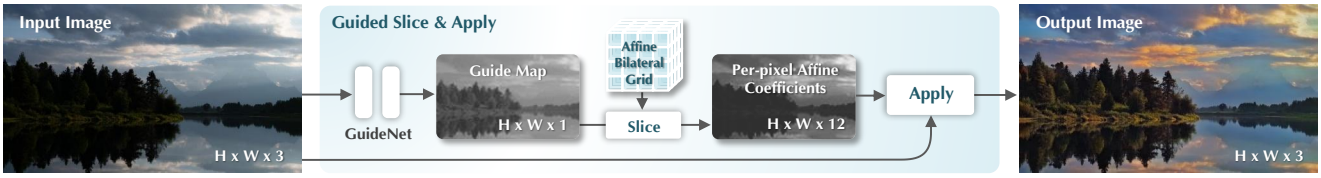

Figure 14. **An overview of our Guided Slice & Apply module.** This fully differentiable module operates at full resolution. First, a lightweight Guide Net processes the input image to produce a single-channel guide map. This map, along with the affine bilateral grid predicted by our adapter, is fed into the *Slice* operation. This operation performs a lookup to generate per-pixel affine coefficients. Finally, the *Apply* operation uses these coefficients to transform the original input pixels into the final retouched output, ensuring high-fidelity, edge-aware editing.

map (size $H \times W \times 1$). This map serves as the range (intensity) coordinate for the subsequent slicing operation.

A design choice here is the use of a *learned* Guide Net instead of a fixed function like a simple grayscale conversion. This is because different colors (RGB values) can map to the same intensity value. If such colors appear adjacent to each other at an important edge, a simple grayscale guide would map them to similar coordinates in the bilateral space. This would cause them to sample from the same or nearby cells in the bilateral grid, leading to a blurring of the edit across that edge. By learning the guide map, the network can create a representation that better separates distinct colors, thereby preserving sharp edges and enhancing the expressive power of the bilateral solution.

Slicing for per-pixel coefficients. The *Slice* operation is the core of the bilateral framework. For each pixel at spatial coordinate (x, y) in the input image, we use its corresponding value from the guide map, $g(x, y)$, as the third coordinate. This forms a 3D coordinate $(x, y, g(x, y))$ which is used to look up a value from the low-resolution affine bilateral grid. This lookup is performed via fast tri-linear interpolation from the 8 neighboring cells in the grid. The result of this operation is a full-resolution tensor of per-pixel affine coefficients with shape $H \times W \times 12$. Each 12-dimensional vector corresponds to the interpolated affine transformation for that specific pixel.

Applying the transformation. The final *Apply* step performs the color transformation. For each pixel, its 12 affine coefficients are reshaped into a 3×4 affine matrix. This matrix is then multiplied with the homogeneous coordinate vector of the corresponding pixel from the original input image, $[R, G, B, 1]^T$. This computes the new RGB values for the output pixel, effectively applying the locally varying, instruction-guided edit to produce the final, high-fidelity output image.

D.4. Details of Prompt Alignment Loss

In the main paper, we introduced the prompt alignment loss, $\mathcal{L}_{\text{align}}$, as a crucial component to ensure our one-step model accurately follows user instructions. Distilling a multi-step diffusion editor can weaken the coupling between the text prompt and the generated output, especially for compositional, directional edits common in photo retouching (e.g., "make it a bit warmer and cozier"). The $\mathcal{L}_{\text{align}}$ loss counteracts this by providing strong, explicit directional supervision. This section provides a detailed breakdown of the attribute matching pipeline and the complete attribute bank used to compute this loss.

The core idea is to convert a free-form user instruction c_T into a small, stable set of atomic retouching attributes $\mathcal{A}(c_T)$. Each attribute corresponds to a predefined positive/negative text prompt pair from our attribute bank (see Table 6). To robustly perform this mapping, we employ a multi-stage pipeline:

1. **Rule-based matching:** We first apply a fast, rule-based matcher that uses keyword and pattern matching to identify attributes. For example, an instruction containing "increase brightness" or "make it brighter" will be mapped to the `brightness:up` attribute. Similarly, "cinematic mood" maps to `style:cinematic`.
2. **Attribute capping:** To ensure training stability and prevent conflicting signals from overly complex prompts, we cap the number of matched attributes at a maximum of three. The three attributes with the highest confidence scores from the rule-based matcher are selected.
3. **LLM fallback:** If the rule-based matcher fails to identify any attributes (i.e., $|\mathcal{A}(c_T)| = 0$), we employ a large language model (LLM) as a fallback mechanism. The LLM is prompted to perform the same task: analyze the instruction c_T and map it to one or more attributes from our predefined bank. This significantly increases the robustness of our matching process for unconventional or creative instructions. The prompt template is shown in Fig. 18.
4. **Loss omission:** In the rare case that both the rule-based matcher and the LLM fallback fail to find any corresponding attributes, the prompt alignment loss term is simply omitted for that specific training instance.

Once the set of attributes $\mathcal{A}(c_T)$ is determined, the fi-

Table 6. The complete Attribute Bank for the $\mathcal{L}_{\text{align}}$ loss. The bank is organized into a two-part layout for readability. Each attribute provides a positive/negative prompt pair for stable, directional contrastive learning.

Part 1: Core Adjustments, Effects, & Styles

Category	Attribute	Positive Prompt	Negative Prompt
Brightness	up	Bright Image	Dark Image
	down	Dark Image	Bright Image
Contrast	up	High Contrast Image	Low Contrast Image
	down	Low Contrast Image	High Contrast Image
	hard	High Contrast Image	Low Contrast Image
	soft	Low Contrast Image	High Contrast Image
Temperature	warm	Warm Color Temperature	Cool Color Temperature
	cool	Cool Color Temperature	Warm Color Temperature
	neutral	Neutral Color Temperature	Tinted Color Temperature
Saturation	up	High Saturation Image	Low Saturation Image
	down	Low Saturation Image	High Saturation Image
Vibrance	up	High Vibrance Colors	Muted Colors
	down	Muted Colors	High Vibrance Colors
Clarity	up	Sharp Detailed Image	Soft Blurry Image
	down	Soft Blurry Image	Sharp Detailed Image
Haze	up	Hazy Foggy Scene	Clear Dehazed Scene
	down	Clear Dehazed Scene	Hazy Foggy Scene
Effects	glow_on	Soft Glow Effect	No Glow Effect
	fog_on	Foggy Scene	Clear Air Scene
	vignette_on	Vignette Effect	No Vignette
	grain_on	Film Grain Effect	Clean Image
Style	cinematic	Cinematic Color Grading	Neutral Documentary Look
	dreamy	Dreamy Soft Look	Crisp Clinical Look
	vintage	Vintage Film Look	Modern Clean Look
	bw	Black and White Photo	Color Photo
	high-key	High-Key Lighting Photo	Low-Key Lighting Photo
	low-key	Low-Key Lighting Photo	High-Key Lighting Photo

Part 2: Moods, Time, HSL, & Intensity

Category	Attribute	Positive Prompt	Negative Prompt
Mood	cheerful	Cheerful Color Grading	Gloomy Color Grading
	cozy	Cozy Warm Look	Cold Sterile Look
	serene	Serene Calm Look	Tense Dramatic Look
	moody	Moody Dark Look	Bright Cheerful Look
	dramatic	Dramatic Look	Flat Calm Look
	nostalgic	Nostalgic Look	Contemporary Clean Look
	ethereal	Ethereal Soft Look	Grounded Realistic Look
Time/Season	sunset	Sunset Colors	Neutral Daylight Colors
	golden-hour	Golden Hour Warm Light	Midday Neutral Light
	night	Nighttime Photo	Daytime Photo
	astro	Astrophotography Night Sky	Daylight Sky Photo
	autumn	Autumn Warm Foliage Colors	Neutral Foliage Colors
	spring	Spring Fresh Colors	Neutral Colors
	winter	Winter Cool Tones	Neutral Tones
HSL Adjust	green:up	More Green Tones	Less Green Tones
	green:down	Less Green Tones	More Green Tones
	blue:up	More Blue Tones	Less Blue Tones
	blue:down	Less Blue Tones	More Blue Tones
	yellow:up	More Yellow Tones	Less Yellow Tones
	yellow:down	Less Yellow Tones	More Yellow Tones
	orange:up	More Orange Tones	Less Orange Tones
	orange:down	Less Orange Tones	More Orange Tones
	magenta:up	More Magenta Tones	Less Magenta Tones
	magenta:down	Less Magenta Tones	More Magenta Tones
Intensity	subtle	Subtle Edit	Strong Edit
	strong	Strong Edit	Subtle Edit

nal loss is calculated as the average of the per-attribute InfoNCE losses, as defined in Eq. 6 of the main paper. This structured approach turns potentially weak or ambiguous instructions into strong, reliable supervision signals that guide the model toward the desired semantic direction.

E. Additional Experimental Details

E.1. Metrics Details and Justification

Content fidelity metrics. The core principle of image retouching is to enhance, not replace. An edit must preserve the original scene's structure and textures. To evaluate this rigorously while factoring out intentional, instruction-guided changes in color and tone, we perform a preprocessing step before calculating fidelity metrics. We first convert both the model's output and the ground truth to grayscale. Then, we apply histogram matching to the model's output, aligning its tonal distribution with that of the input image's grayscale version. This ensures our metrics focus purely on structural and textural integrity. On these processed images, we compute:

- **SSIM** [42]: Measures perceptual similarity based on structure, luminance, and contrast.
- **CW-SSIM** [36]: A complex wavelet variant of SSIM, more sensitive to geometric and textural distortions.
- **DISTS** [8]: A learned metric that explicitly models textural similarity and is highly correlated with human perception of distortion.
- **GMSD** [47]: Measures the deviation of gradient magnitudes, effectively capturing local structural changes.

Editing quality metrics. To assess how well the model follows instructions and the aesthetic quality of the result, we use several metrics. L1 and L2 distances are computed against the ground truth to measure pixel-level accuracy. Following recent work [28, 51], we also employ GPT-4o to provide automated ratings for:

- **SC (Score-Correctness, 0-10):** Instruction-image alignment.
- **PQ (Perceptual Quality, 0-10):** Overall perceptual quality.
- **O (Overall Score):** The geometric mean $\sqrt{\text{SC} \times \text{PQ}}$ to balance both aspects.

The prompt template used to elicit these scores from GPT-4o is detailed in Fig. 19.

Justification on no-reference IQA (NR-IQA). In our experiments (Sec. C.1.2), we observed that generative models often achieve high NR-IQA scores (*e.g.*, MUSIQ, Hyper-IQA) at the cost of severely degraded content fidelity. These NR-IQA models, trained on diverse web data, tend to reward clear details, which generative models can "hack" by

hallucinating new textures or altering content. For the task of retouching, where preserving the original subject matter is paramount, this behavior is undesirable. Therefore, while we report NR-IQA scores for completeness, we argue that fidelity-aware metrics (DISTS, SSIM) and reference-based quality metrics (L1/L2, SC/PQ) are more reliable indicators of true performance in the context of image retouching.

E.2. Additional Implementation Details

Our model is built upon a pre-trained InstrucPix2Pix model [2]. We initialize our U-Net weights from this pre-trained model and keep the VAE frozen throughout training. Our training process follows a two-stage progressive distillation strategy.

Stage 1: Low-Resolution Branch Distillation. In this stage, we exclusively train the low-resolution one-step diffusion branch (ϵ_θ) and its VSD regularizer ($\epsilon_{\phi'}$). Training in this stage follows a three-phase curriculum that anneals the VSD timestep t and adjusts loss weights, as detailed in Tab. 7. This curriculum strategy guides the model to first learn coarse structure (high t), then focus on instruction alignment (mid t), and finally refine subtle details (low t). The total loss for this stage is $\mathcal{L}_{\text{stage1}} = \mathcal{L}_{\text{data}} + \lambda_{\text{VSD}}\mathcal{L}_{\text{VSD}} + \lambda_{\text{align}}\mathcal{L}_{\text{align}}$.

Table 7. Three-phase curriculum for Stage 1 training.

VSD Timestep Range	Loss Function Composition	Training Steps
$[400, 800]$	$\mathcal{L}_{\text{vsd}} + 10 \cdot \mathcal{L}_{\text{data}}$	40k
$[200, 400]$	$\mathcal{L}_{\text{vsd}} + 2 \cdot \mathcal{L}_{\text{data}} + \mathcal{L}_{\text{align}}$	40k
$[50, 200]$	$\mathcal{L}_{\text{vsd}} + 0.5 \cdot \mathcal{L}_{\text{data}} + \mathcal{L}_{\text{align}}$	40k

Stage 2: Joint Bilateral Distillation. After Stage 1 converges (after 120k steps), we unfreeze the lightweight bilateral adapter and train the entire network end-to-end for an additional 80k steps. The loss function is augmented with our bilateral loss: $\mathcal{L}_{\text{stage2}} = \mathcal{L}_{\text{stage1}} + \lambda_{\text{bila}}\mathcal{L}_{\text{bila}}$. This stage fine-tunes the full pipeline, teaching the bilateral branch to generate high-fidelity, full-resolution edits.

Hyperparameters. We train our model at a 512x512 resolution with a total batch size of 64. We use the AdamW [29] optimizer with a learning rate of 1×10^{-4}. We use automatic mixed-precision training for efficiency and apply gradient clipping with a max norm of 1.0. An exponential moving average (EMA) of the model weights is maintained with a decay rate of 0.999. We use a $32 \times 32 \times 8$ bilateral grid in our implementation.

E.3. Additional User Study Details

To complement our quantitative analysis, we conducted a subjective user study to assess human preference.

Setup. The study involved 30 participants with diverse backgrounds. We drew 20 examples from our iRetouch benchmark for each user, covering a wide range of instructions. For each example, we presented the original image and the text instruction, followed by five edited results generated by our method and four leading baselines (FLUX.1-Kontext-pro, Gemini-2.5-Flash, Qwen-Image, GPT-Image-1).

Procedure. To ensure impartiality, the five results were displayed in a randomized order and labeled anonymously as 'A', 'B', 'C', 'D', and 'E'. Participants were unaware of which method corresponded to each label. For each example, they were asked to evaluate the five results based on four distinct criteria by answering the following questions:

- **Content fidelity:** "Which result(s) best enhance the image while preserving the original content and structure, without adding or removing objects/textures?"
- **Editing ability:** "Which result(s) most accurately and effectively follow the given text instruction?"
- **Visual quality:** "Ignoring the instruction for a moment, which result(s) have the highest overall visual and aesthetic quality (e.g., pleasing colors, no artifacts)?"
- **Overall preference:** "If you could keep or share one or more edited images, which one(s) would you pick?"

Participants were asked to select all options they considered the best for each question (multiple selections allowed). The win rate for each method is computed as the percentage of trials in which it was selected. The aggregated results, presented in Fig. 4 of the main paper, show a clear and consistent preference for our method across all four dimensions, validating its superior performance in producing high-fidelity, instruction-aligned, and aesthetically pleasing edits.

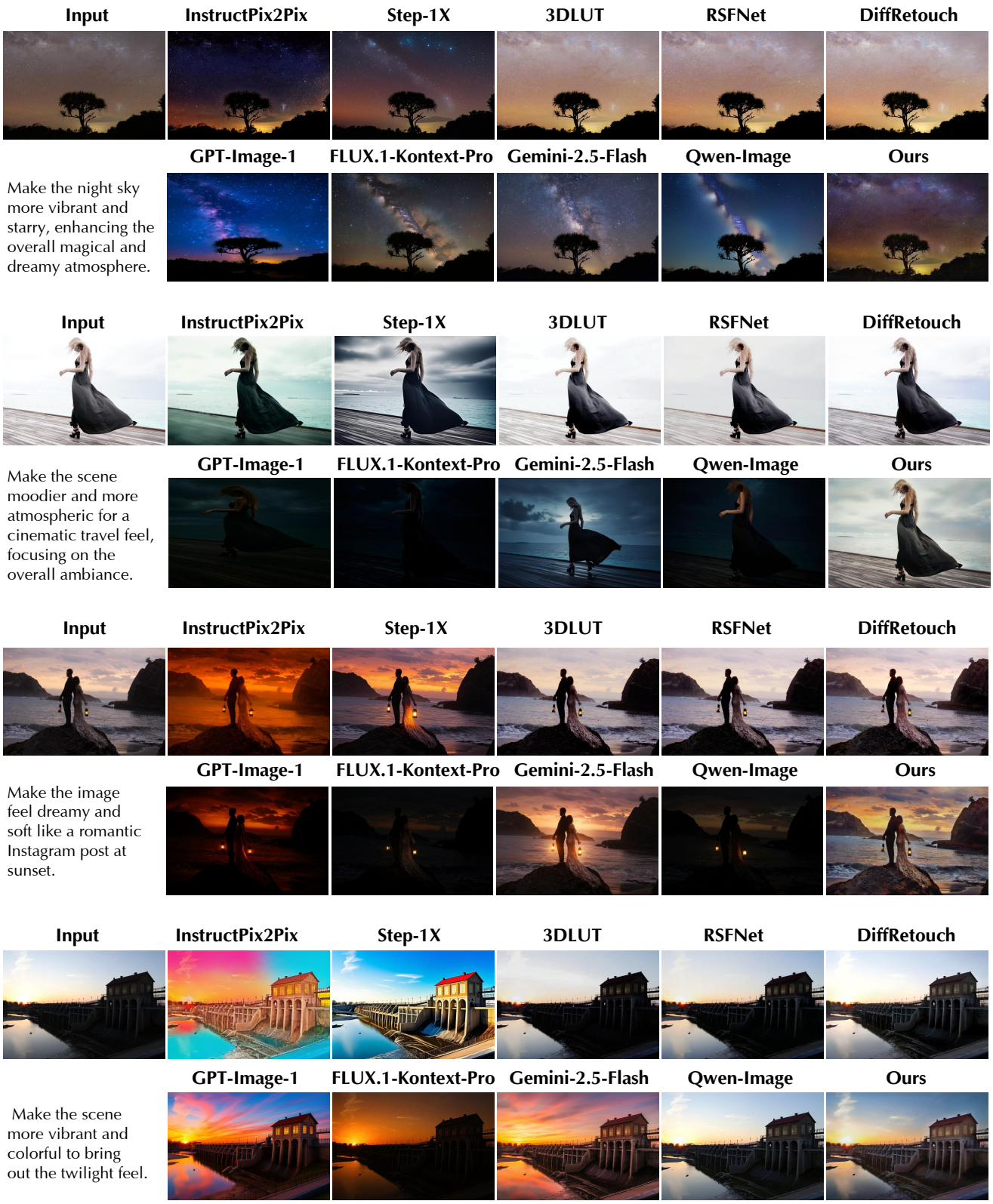

Figure 15. **Additional visual comparisons on our iRetouch benchmark.** Our method produces high-fidelity results that accurately follow the instructions, while baselines often alter content or fail to produce the desired effect.

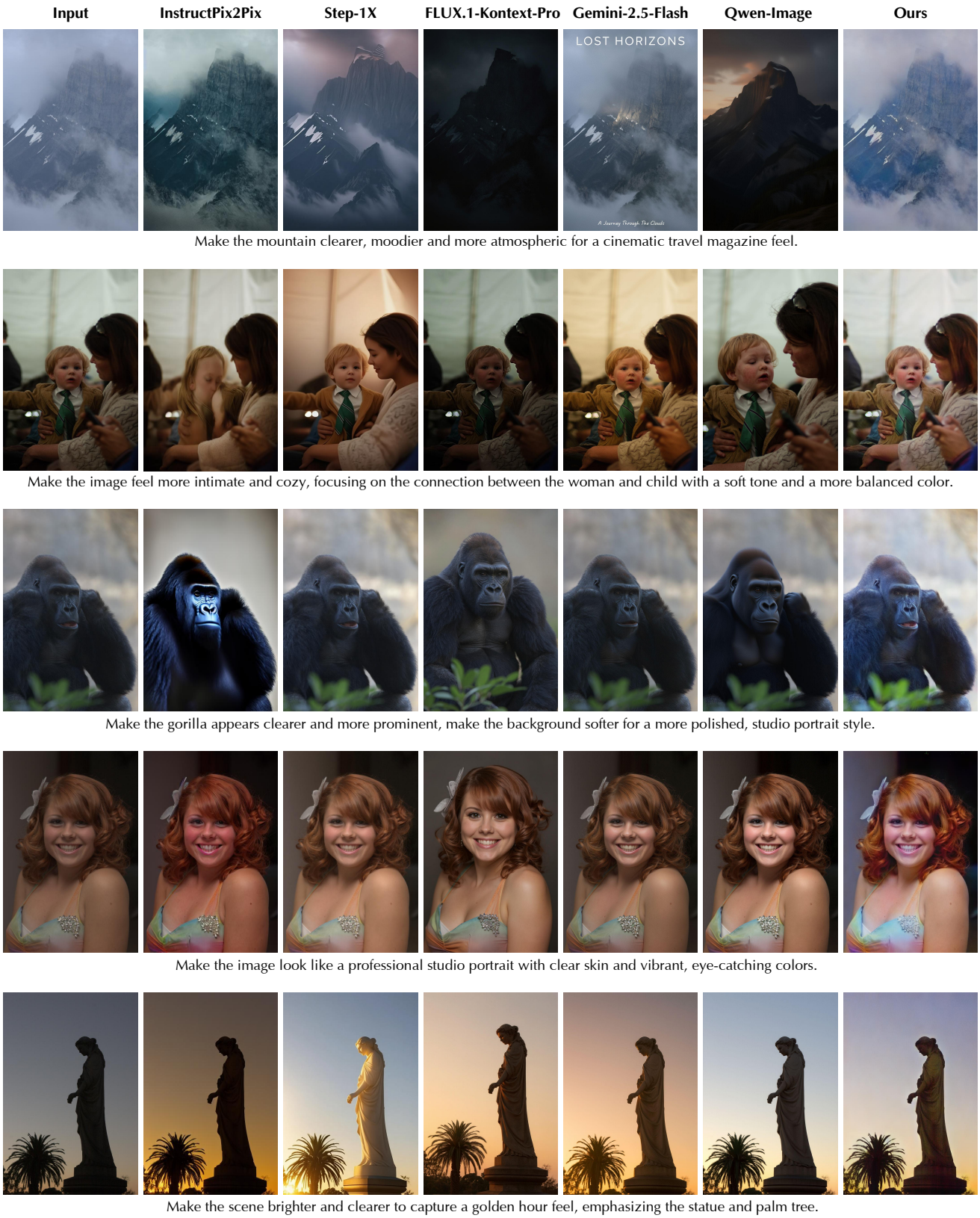

Figure 16. **Qualitative comparison on the MIT-Adobe FiveK dataset.** Our method demonstrates a superior ability to interpret complex, nuanced instructions and produce faithful, high-quality results compared to leading generative models, which often introduce unwanted repaints or hallucinate content.

Prompt Template for MLLM-based Instruction Generation

System Prompt

You are a helpful and intuitive photo editing assistant that interprets image edits from the perspective of an everyday user.

Your task is to analyze two images:
- The first is the original image before editing.
- The second is the edited version.

From these, infer the editing intention behind the changes, and express this intention as a natural, casual instruction that a general user might give to a photo editor.

User Prompt

Special Instructions:

1. Simulate a real person describing how they want the image to look — in casual, business-friendly English, not technical terms.
2. Describe the editing **intention** and **overall style** ("make it moodier", "give it a cinematic look", "add a warm sunset vibe", etc.), not the editing **action** ("increase contrast").
3. Do **not** describe what changed in the image. Instead, say what the user wanted to achieve with the edit.
4. Use **relatable terms** and style descriptors — e.g., "Instagram look", "studio portrait style", "vintage tone", "travel magazine feel", and more.
5. Avoid any numeric values or technical jargon (e.g., "exposure +1", "saturation 15").
6. Keep the instruction short — ideally one sentence under 20 words. **NO** quotation marks or special formatting.
7. If there were localized changes (e.g., face brightening, making certain area more prominent, etc.), include those intentions smoothly into the sentence. Look for enhancing specific features like making certain areas brighter, darker, more vibrant, or more prominent.

Examples (maybe some are too long, refer to the sentence structure only, ensure diversity of sentence structures):

1. Enhance the cat's eyes to pop more for a striking Instagram look while keeping the background soft and dreamy.
2. Make the sky brighter and warmer to enhance the sunset vibe and create a more vibrant, uplifting atmosphere.
3. I want this to look like it's straight out of a movie with depth, atmosphere, and film texture. Make the person clearer, smooth the skin, and enhance the jacket's colors and details.
4. Make my face look brighter and my eyes to stand out more. Can you make the skin smoother and the hair darker? Enhance the whole image without over-saturating colors.

Input:

Original Image: <img_1>
Edited Image: <img_2>

Output (Editing Intention):

Figure 17. **The complete prompt template used to guide the MLLM for instruction generation.** This structured prompt is designed to elicit user-centric, intention-focused descriptions of photo edits. By adopting a specific persona and following a set of detailed rules and examples, the model learns to generate concise and natural language instructions that form our training data.

Prompt Template for Instruction Parsing (LLM Fallback)

You are given multiple image editing instructions and an attribute OPTIONS dictionary.

For each instruction, pick ZERO, ONE, or MULTIPLE matching attributes (category+key) from OPTIONS. Also set `local_hint=true` if the instruction targets a region/object (e.g., sky, skin, grass, rose, background, subject); else `false`.

Return STRICT JSON with this schema:

```
{
  "results": [
    {"i": <int>, "matches": [{"category": <str>, "key": <str>, ...]},
    ...
  ]
}
```

Rules:

- Only use keys exactly from OPTIONS.
- If nothing fits, return `matches: []` for that item.
- Do NOT output extra text.

DATA:

```
{
  "instructions": [
    {
      "i": 0,
      "text": "make the sunset clouds more dramatic"
    },
    {
      "i": 1,
      "text": "give it a soft, dreamy feel"
    }
  ],
  "options": {
    "brightness": ["up", "down"],
    "contrast": ["up", "down", "hard", "soft"],
    "temp": ["warm", "cool", "neutral"],
    "saturation": ["up", "down"],
    "vibrance": ["up", "down"],
    "clarity": ["up", "down"],
    "haze": ["up", "down"],
    "effects": ["glow_on", "glow_off", "vignette_on", ...],
    "style": ["cinematic", "dreamy", "vintage", "bw", ...],
    "mood": ["cheerful", "cozy", "serene", "moody", ...],
    "time": ["sunset", "golden-hour", "night", "astro", ...],
    "hsl": ["green.up", "green.down", "blue.up", "blue.down", ...]
  }
}
```

Figure 18. **The prompt template for our LLM-based instruction parser.** This prompt provides the LLM with a list of user instructions and a dictionary of valid, predefined attributes. The model is tasked with returning a structured JSON object that maps each instruction to one or more corresponding edits from the dictionary, including a hint for whether the edit is local. This mechanism acts as a robust fallback, translating complex or ambiguous requests into pre-defined attributes.

Prompt Template for MLLM-based Evaluation

System Prompt

You are a post-production specialist with expertise in enhancing photographic imagery through advanced digital editing techniques. We now need your help to evaluate the performance of an AI-powered image post-editing tool for photography.

User Prompt

INPUTS:

1. Two images will be provided: The first being the original photographic image and the second being an edited version of the first.
2. The editing instruction will be provided: The post-editing needs of photographic images expressed by users with no image processing knowledge.

METRICS (From scale 0 to 10):

User Instruction Satisfaction Score: A score from 0 to 10 will be given based on how well the edits follow the user's instructions.

- Users typically have both global and local editing requirements when working with photographic images. Therefore, this score should be evaluated holistically, taking into account the user's needs for both local and global adjustments, with equal importance given to each.
- 0 indicates that the edited image does not follow the editing instruction at all.
- 10 indicates that the edited image follows the editing instruction text perfectly.

Perceptual Quality Score: A second score from 0 to 10 will rate the visual quality of the image after editing.

- Need to evaluate visual quality regarding image naturalness, artifacts and aesthetic quality.
- The edited image should look natural. It should have a natural color, lighting and photorealistic appearance.
- Need to compare before and after images to assess image artifacts. The edited image should prevent visual artifacts, such as content drifts, texture distortion, subjects not harmonized, etc.
- 0 indicates that the edited image does not look natural at all or contains a large portion of artifacts such as distortion, blurred faces, or subjects not harmonized.
- 10 indicates that the edited image looks natural and has no artifacts.

You will have to give your output in this way (Keep your reasoning concise and short.):

```
{
"score" : [score1, score2],
"reasoning" : "..."
}
```

Put the score in a list such that output score = [score1, score2], where 'score1' evaluates the User Instruction Satisfaction Score and 'score2' evaluates the Perceptual Quality Score.

Editing instruction:

Figure 19. **The complete prompt template for MLLM-based evaluation.** The MLLM, acting as a post-production specialist, assesses an AI-edited image based on two criteria: fidelity to the user's instruction and overall perceptual quality. It outputs a structured JSON with scores and a brief rationale.

References

[54] Shaolin Su, Qingsen Yan, Yu Zhu, Cheng Zhang, Xin Ge, Jinqiu Sun, and Yanning Zhang. Blindly Assess Image Quality in the Wild Guided by a Self-Adaptive Hyper Network. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[55] Hossein Talebi and Peyman Milanfar. NIMA: Neural Image Assessment. *IEEE Transactions on Image Processing*, 27(8):3998–4011, 2018.

[56] Sylvain Paris and Frédo Durand. A fast approximation of the bilateral filter using a signal processing approach. *International Journal of Computer Vision*, 81(1):24–52, 2009.

[57] Jing Shi, Ning Xu, Yihang Xu, Trung Bui, Franck Dernoncourt, and Chenliang Xu. Learning by planning: Language-guided global image editing. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13590–13599, 2021. 4, 5

[58] Yi Dong, Yuxi Wang, Xianhui Lin, Wenqi Ouyang, Zhiqi Shen, Peiran Ren, Ruoxi Fan, and Rynson W.H. Lau. GenColor: Generative and expressive color enhancement with pixel-perfect texture preservation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2025. 4, 5

[59] Niladri Shekhar Dutt, Duygu Ceylan, and Niloy J. Mitra. MonetGPT: Solving puzzles enhances MLLMs' image retouching skills. *ACM Transactions on Graphics (TOG)*, 44(4):1–12, 2025. 4, 5

[60] Vladimir Bychkovsky, Sylvain Paris, Eric Chan, and Frédo Durand. Learning photographic global tonal adjustment with a database of input/output image pairs. In *CVPR*, pages 97–104, 2011.