# UNISLIDER: PERCEPTUALLY UNIFORM SLIDERS FOR CONTINUOUS IMAGE EDITING

**Anonymous authors**
Paper under double-blind review

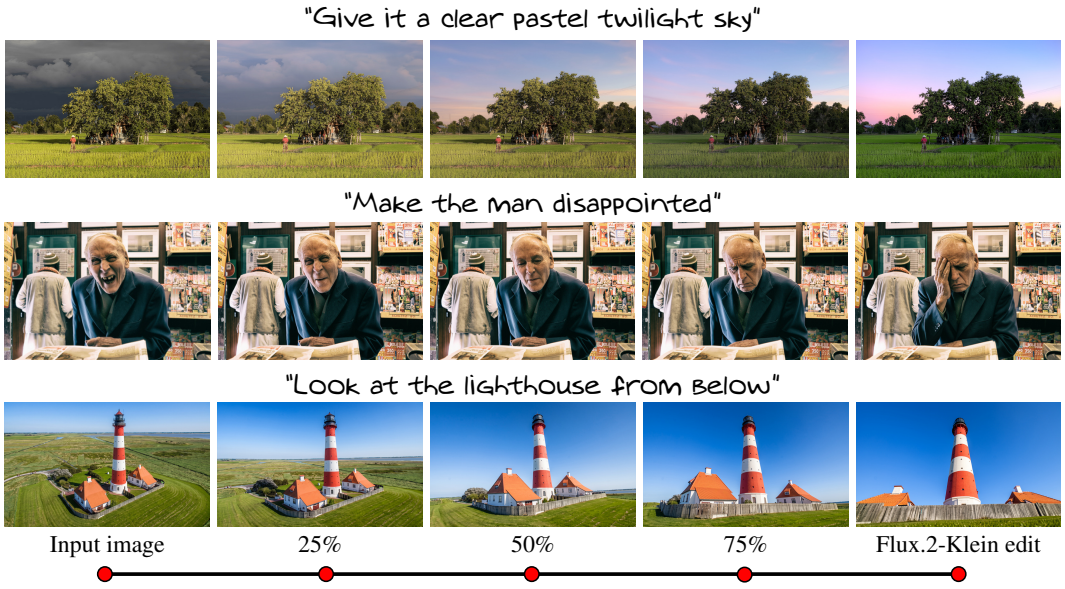


Figure 1: **UniSlider produces perceptually uniform edit strengths.** Each row shows one image edited at increasing strength under the instruction above it. Equidistant steps along the slider produce equal amounts of perceptual change; the full interval is thus utilized smoothly and predictably.

## ABSTRACT

Sliders provide an intuitive interface for continuous image editing. In current generative approaches, however, the slider is simply a rescaling of the method's strength parameter, such as an adapter coefficient, a prompt weight, or an interpolation factor. This strength relates poorly to perceptual change. The image can partially revert as the slider moves, long stretches of the range produce no visible difference, and short intervals transform the image abruptly. Remapping the strength could fix this uneven pace, but only if the trajectory is monotone, which current methods do not enforce. We therefore distinguish the slider from the strength, and require perceptual distance from the input to grow linearly with the slider value. We introduce UniSlider, a lightweight LoRA trained on a few-step editing backbone so that its strength approximates this ideal slider. Few-step sampling lets us impose this objective in pixel space without intermediate ground truth, and the backbone's output is preserved at full strength. However, a low-rank adapter cannot make the strength fully uniform. Our slider is thus an inference-time remapping of the strength, obtained by adaptive sampling. Since training optmizes to make the trajectory monotone, this remapping closes the remaining gap without extra training or parameters. On a new benchmark of 300 continuous edits evaluating uniformity, monotonicity, edit fidelity, and identity preservation, UniSlider outperforms all prior methods and is preferred in a user study.

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

# 1 INTRODUCTION

Sliders are the near-universal interface for continuous image editing. In traditional editing software, a slider behaves like a physical dial, changing the image proportionally to how far the handle moves, so the same displacement produces a comparable change anywhere in the slider range. Generative sliders inherit this interface, but most fail to match its behavior. Because the mapping from slider value to perceptual change is highly non-uniform, users may find them confusing to control.

Existing generative sliders depart from the expected behavior of a physical dial in two ways. The first is non-uniformity, which has two complementary symptoms: **dead zones**, intervals across which the image barely changes, and **jumps**, intervals across which the image changes far more than a small drag should imply. The second is non-monotonicity, which appears as **reversals**: moving the slider forward partially undoes the edit. Much of the range thus becomes unusable, leaving the user with only a few effective settings that depend on the edit.

Perceptual uniformity has not, to our knowledge, been posed as an objective for generative image-editing sliders. Prior work has focused on what the edit does, not on how changes are distributed along the range. The slider is simply a linear rescaling of the method's internal strength parameter, inherited from whichever quantity happens to be continuous: an adapter coefficient (Gandikota et al., 2024; Zarei et al., 2026), a prompt weight (Parihar et al., 2026; Guerrero-Viu et al., 2024), or an interpolation factor (Li et al., 2026; Endo et al., 2026). Remapping this strength could fix non-uniformity, but only if the editing trajectory is monotone. Since the strength is typically supervised only at its endpoints, if at all, nothing enforces this. We therefore distinguish between the internal strength, which drives the edit, and the user-facing slider, whose value should grow linearly with perceptual distance from the input. Figure 2 (top row) illustrates this for SliderEdit (Zarei et al., 2026), the current state of the art. We sample images at equal steps of its strength and plot the relative perceptual change of each. An ideal slider traces the diagonal (equal steps, equal change), but SliderEdit's curve shows a dead zone and a jump.

We introduce UniSlider, a text-based editing method with a perceptually uniform slider. We repurpose a few-step editing backbone by training a lightweight LoRA to edit an image such that the perceptual distance from the input is proportional to the given strength. Using a few-step editing backbone allows decoding to pixel space during training and imposing perceptual uniformity directly, with no ground-truth intermediate images. This alone already removes most of the dead zones and jumps, making our vanilla LoRA track the ideal perceptual diagonal far more closely (Fig. 2, middle row). However, the low-rank space cannot reach full uniformity on every image and editing concept. We close the remaining gap with an inference-time adaptive sampling step that remaps the strength values onto the slider shown to the user, so that equal slider steps produce near-equal perceptual change without any additional training (Fig. 2, bottom row). Figure 1 shows three UniSlider examples.

Evaluating a slider quantitatively is itself challenging. Infinitely many trajectories are valid for the same edit instruction, so metrics that compare against a reference image do not apply. There is also no shared benchmark. Existing slider methods report results on small, method-specific test sets that are not publicly available, while general editing benchmarks such as PIE-Bench (Ju et al., 2024) include edits with no meaningful intermediate strength, such as adding or removing an object. We address this evaluation gap by introducing a benchmark specifically designed to assess continuous image editing.

We summarize our contributions as threefold:

- We identify and formalize perceptual uniformity as a failure mode of text-based image sliders.
- We propose UniSlider, a perceptually uniform slider for continuous image editing. A lightweight LoRA is trained on top of a few-step editing backbone, optimizing a pixel-space perceptual metric so that perceptual change grows linearly with the strength value. Inference-time adaptive sampling then maps the strength to a perceptually uniform slider.
- We introduce a benchmark for continuous text-based image editing, covering 300 cases in three editing categories, which evaluates uniformity jointly with edit fidelity and identity preservation. UniSlider performs consistently best across the metrics.

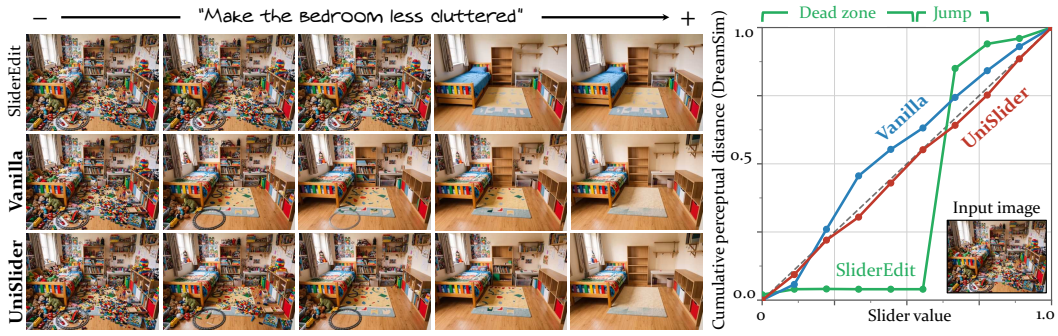


Figure 2: **Training closes most of the gap to a uniform slider; adaptive sampling closes the rest.** All three rows edit the same image with the same instruction. **Left:** frames at equal slider values for SliderEdit, our loss alone (**Vanilla**), and our full **UniSlider** method with strength reparametrization. **Right:** perceptual distance from the input to every slider frame, normalized by the total input–edit distance. The diagonal marks a perfectly uniform slider. We use DreamSim (Fu et al., 2023) as a perceptual metric.

## 2 Related Work

Sliders are attractive for editing as they provide means to dial edits up or down rather than accepting or rejecting them outright. Unfortunately, building generative sliders is challenging, due to the lack of ground-truth supervision. Existing generative sliders fall into two categories. Training-free methods expose a continuous coefficient in the inference process: classifier-free-guidance scale, semantic space interpolation, or velocity-field decomposition without any additional supervision (Endo et al., 2026; Li et al., 2026; Ekin & Gandelsman, 2026). This approach requires no training data, but the resulting axis is a byproduct of the diffusion sampler rather than a quantity shaped to behave predictably, and we find it the less reliable of the two categories (Section 6).

Training-based methods instead learn the strength axis, typically as a LoRA adapter scaled by a coefficient. Concept Sliders (Gandikota et al., 2024) trains one adapter per concept from a small set of contrastive image pairs, targeting generation rather than editing a given input. Alchemist (Sharma et al., 2024) fine-tunes a diffusion model on a synthetic dataset of rendered material variations to control object-intrinsic attributes. Kontinuous Kontext (Parihar et al., 2026) takes a different route to supervision as it builds pseudo-ground-truth intermediate edits by morphing between an input and its edit with a pre-trained morphing method. Consequently, its output quality is tied to the morphing procedure, and it inherits any artifacts that procedure introduces. SliderEdit (Zarei et al., 2026) trains a LoRA to cancel the edit of the base editing model, using the frozen model under a neutral instruction as a self-supervised target. In all these methods, the slider exposed to the user is simply the learned strength, linearly rescaled to $[0, 1]$, so it inherits any non-uniformity of the strength axis.

Taken together, this line of work establishes that a LoRA's scaling coefficient is a good place to expose edit strength as it is simple, requires no architecture changes, and composes with any editing backbone. However, with the partial exception of Kontinuous Kontext's morphed trajectories, none of these methods place any requirement on what happens between the endpoints. The strength coefficient is supervised, if at all, only at its extremes, and the middle of the interpolation is left to whatever the LoRA does on its own. This is precisely the gap UniSlider targets. A broader discussion of related works is provided in Appendix B.

## 3 Method

**Definition.** Let $x_{\text{src}}$ be the input image and $c$ be an editing text instruction. A few-step pretrained image editing model $f$ produces a single output $x_{\text{edit}} = f(x_{\text{src}}, c)$, which we refer to as the *default edit*. We extend this model into a continuous one-parameter family $x_s = f(x_{\text{src}}, c, s)$ where $s \in [0, 1]$ denotes the editing strength. The strength should provide a smooth transition from the input image to the default edit while preserving the behavior of the underlying editor at both endpoints.

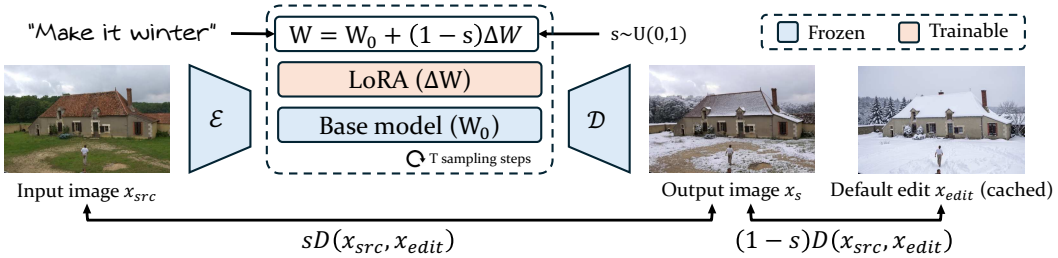


Figure 3: **UniSlider training scheme.** We sample $s$, scale the trainable LoRA against the frozen backbone (Eq. (3)), and run the full $T$-step inference process to get $x_s$; $x_{\text{edit}}$ is cached once at $s$=1. The loss (Eq. (4)) pushes $x_s$ to the $s$-implied perceptual position along the path from $x_{\text{src}}$ to $x_{\text{edit}}$.

In particular, we require $x_{s=0} = x_{\text{src}}$ and $x_{s=1} = x_{\text{edit}}$, so that introducing continuous control does not alter the input or the default edit. Moreover, intermediate outputs $x_{s\in(0,1)}$ should remain on the perceptual path between the two endpoints and not introduce changes unrelated to the instruction. Given a perceptual distance $D(\cdot,\cdot)$, these requirements correspond to

$$D(x_{\text{src}}, x_s) + D(x_s, x_{\text{edit}}) = D(x_{\text{src}}, x_{\text{edit}}), \qquad \forall s \in [0,1]. \tag{1}$$

Most importantly, we want the perceptual change to be distributed uniformly across the slider. Specifically, the perceptual distance from the input should increase linearly with the slider strength,

$$D(x_{\text{src}}, x_s) = sD(x_{\text{src}}, x_{\text{edit}}), \qquad \forall s \in [0,1]. \tag{2}$$

Under this formulation, equal increments of $s$ correspond to equal perceptual changes, eliminating the dead zones and jumps commonly observed in existing editing sliders. For the rest of the paper we use DreamSim (Fu et al., 2023) as perceptual distance $D(\cdot,\cdot)$, as our empirical evaluation demonstrates it correlates best with human judgments of edit magnitude and training stability (Section 6).

**Method.** We build on few-step text-based diffusion transformer (DiT) editing backbones which use $T = 4$ inference steps or less. Strength is exposed through a LoRA (Hu et al., 2022) adapter applied to all DiT projection layers and scaled by the slider value:

$$W(s) = W_0 + (1-s)\Delta W. \tag{3}$$

The adapter $\Delta W$ vanishes at full strength $s = 1$, so the model reduces exactly to the frozen backbone and returns the default edit $x_{s=1} = x_{\text{edit}}$ by construction. The other slider endpoint, the input $x_{s=0} = x_{\text{src}}$, has to be learned. We share the parameterization in Eq. (3) with Zarei et al. (2026) but not the training scheme. Rather than supervising the adapter towards a predefined notion of strength, we train it to produce a trajectory whose perceptual distance from the input follows Eq. (3) while remaining on the path between the input and the default edit. We use Flux.2-Klein-4B as our image editing backbone.

**Training.** At each training step, we draw $s \sim \mathcal{U}(0,1)$, set the adapter scale according to Eq. (3), and run the full $T$-step sampling process to obtain $x_s$. Because $T$ is small, this is affordable inside the training loop and, following NP-Edit (Kumari et al., 2026), allows us to define the objective directly on decoded pixels rather than on velocity estimates.

We measure the perceptual distances between the strength-$s$ edit and the two endpoints, penalizing deviations from their desired values:

$$\mathcal{L} = \underbrace{\left(D(x_{\text{src}}, x_s) - sD(x_{\text{src}}, x_{\text{edit}})\right)^2}_{\text{distance to input}} + \underbrace{\left(D(x_s, x_{\text{edit}}) - (1-s)D(x_{\text{src}}, x_{\text{edit}})\right)^2}_{\text{distance to default edit}}. \tag{4}$$

The two symmetric terms enforce uniform progress along the perceptual path between the input and the default edit. Because the target distance from the input grows strictly with $s$ and the target distance to the default edit shrinks, the loss also penalizes reversals. A trajectory whose distance from the input decreases anywhere, or whose distance to the default edit increases, cannot reach zero loss. Our loss thus promotes perceptual uniformity, monotonicity and path consistency without requiring ground-truth intermediate images. $x_{\text{edit}}$ is produced by the frozen backbone from its corresponding $x_{\text{src}}$ and cached. Figure 3 shows an overview of the UniSlider training scheme.

**Training data.** To train the LoRA layers, we curate a small training set of 300 images from the MIT5K (Bychkovsky et al., 2011) dataset. We then use Qwen3-VL-30B (Bai et al., 2025) to generate image-specific text instructions describing meaningful and continuous edits for each image. Examples from the training set are provided in the supplementary material.

**Spatial masking.** When the backbone operates in a (latent) space aligned with the image (e.g. Flux.2-Klein), we can spatially localize the editing process at inference time. Inspired by inpainting methods (Lugmayr et al., 2022; Avrahami et al., 2022), we preserve the latent features of the input image $z_0$ in regions that should remain unchanged. Specifically, we modify the latent representation at each flow-matching step as

$$z_t = z_t \odot \mathcal{M} + z_\sigma \odot (1 - \mathcal{M}). \tag{5}$$

$\mathcal{M}$ denotes a spatial mask and

$$z_\sigma = (1 - \sigma)z_0 + \sigma\epsilon, \tag{6}$$

where $\sigma$ is the noise sampler. Unlike conventional inpainting methods, which typically use a binary mask, we allow $\mathcal{M} \in [0, 1]$ to enable smooth spatial transitions between edited regions (mask values 1) and preserved regions (mask values 0).

## 4 Improving Perceptual Uniformity with Adaptive Sampling

The training objective in Eq. (4) produces slider trajectories that satisfy endpoint consistency, remain on the editing path, and vary smoothly and monotonically with the slider strength. While these properties provide a strong basis for perceptually uniform editing, the limited capacity of a low-rank model adapter prevents achieving perfect uniformity across arbitrary image–instruction pairs.

To characterize the capacity limitation, we overfit the adapter on $N \in 1, 2, 4, 8, 16, 32$ examples and evaluate the achieved uniformity on them, sweeping the adapter rank $r \in 4, 8, 16, 32$. In Fig. 4, each panel plots the perceptual (DreamSim) distance $D(\cdot, \cdot)$ between the input to the slider output at strength $s$, normalized by the total distance between input and default edit. An ideal, perceptually uniform slider traces the diagonal, whereas flat and steep regions represent dead zones and jumps.

Figure 4 reveals a consistent trend. For small $N$, the adapter achieves near-perfect uniformity, but increasing the number of examples eventually leads to less uniform trajectories. This transition occurs at larger $N$ when the adapter rank is higher, indicating that the adapter capacity determines how many image–instruction pairs can be represented with highly uniform trajectories. Increasing the rank can extend this capacity, but at the cost of increased parameter count and computation, contrary to our goal of lightweight adaptation. Instead of increasing model capacity, we therefore ask whether the existing trajectory can be traversed more effectively at inference time.

**Adaptive sampling.** We introduce adaptive sampling as a lightweight extension to our training scheme that improves perceptual uniformity without modifying or retraining the learned adapter. Rather than exposing the strength directly as the slider (i.e. sampling the learned trajectory at equispaced values of $s$), we place the samples adaptively by recursively splitting the interval with the largest perceptual distance under $D$ (Appendix C). We then use these samples to remap the strength axis. For each edit, we compute a mapping $u = g(s)$ from strength to the user-facing slider value $u \in [0, 1]$, such that the perceptual distance from the input grows approximately linearly with $u$, as in Eq. (2). This remapping relies on the monotonicity obtained through training. Prior methods, whose strength is supervised at most at its endpoints, offer no such guarantee, and remapping cannot repair their reversals (Table 3).

The last column of Fig. 4 demonstrates the benefit of this inference-time strategy. We train a single adapter on our full 300-example training set at different ranks $r$ and apply adaptive sampling at inference time, evaluating on the $N = 32$ (now unseen) examples corresponding to the final column. Despite using the same adapter and sampling budget, adaptive sampling substantially improves the uniformity of the resulting slider trajectories by being an average of $\times 3.7$ times closer to the diagonal in terms of RMSE againts the $N = 32$ experiment. Crucially, this improvement requires no additional training or model parameters with a negligible overhead. Thus, adaptive sampling provides a simple way to exploit the smooth trajectories learned by the LoRA adapter while compensating for variations in how perceptual change is distributed along the strength axis.

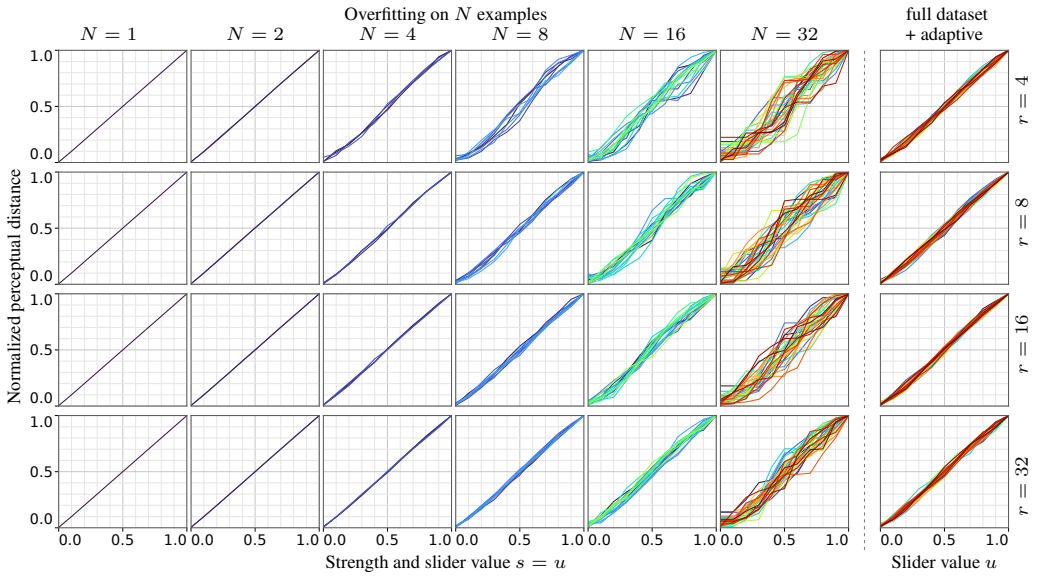


Figure 4: **Uniformity degrades with the number of training examples, and is recovered by adaptive sampling.** Normalized DreamSim distance from the input to the output, for adapter rank $r$ and $N$ overfitting examples. The diagonal is the ideal uniform slider. First six columns: the strength is the slider ($s = u$). Rightmost column (full method): one adapter trained on the full dataset, evaluated on $N = 32$ unseen examples, with adaptive sampling mapping $u$ to $u = g(s)$.

## 5 BENCHMARK

Existing image editing benchmarks, such as Pie-Bench (Ju et al., 2024), primarily evaluate whether a model can perform a given edit, but are not designed to assess continuous editing across slider strengths. Other works therefore rely on small, concept-specific evaluation sets, limiting the diversity of edits and scenes considered. To address this gap, we introduce a benchmark specifically designed to evaluate continuous image editing, comprising 300 carefully curated image–instruction pairs spanning diverse scenes and editing concepts. We categorize the edits into three types: *Global or Stylization*, which apply a consistent change across the image; *Attribute or Local*, which modify a specific object or region; and *Structural*, which alter the image content with changes that depend on its semantics. Detailed examples of the image and instruction generation, full metric definitions and evaluation protocols are provided in the Supplementary Material.

**Images and text instructions.** We select diverse images from the publicly available Adobe-MIT-FiveK (Bychkovsky et al., 2011) dataset and curate text instructions describing edits that admit a meaningful continuous progression from the original image to the desired result. Instructions were designed to avoid inherently binary operations, such as object removal, and instead cover edits for which intermediate strengths are well-defined. We further manually curate the resulting image–instruction pairs to ensure that the requested edits are meaningful and sufficiently diverse across the three categories.

**Metrics.** We evaluate six properties that characterize a high-quality image editing slider: *edit fidelity*, *identity preservation*, *endpoint consistency*, *image quality*, *strength monotonicity*, and *perceptual uniformity*. Edit fidelity measures whether the requested edit is successfully performed at full strength, while identity preservation evaluates whether the original scene and unrelated content are maintained. At its lower endpoint, the slider should return the input image unchanged. We assess image quality on all generated images using a no-reference quality metric. Strength monotonicity measures violations in the progression of edit fidelity, while perceptual uniformity is measured from the variation of perceptual distances between consecutive slider outputs. We use complementary image- and VLM-based metrics for these properties. Full metric definitions, validations, examples of the metrics, and evaluation protocols are provided in Appendix E.

| Method | Edit Fidelity | | Identity | Endpoint | IQA | Monotonicity | | Uniformity | User Study | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $\text{CLIP}_{\text{dir}}\uparrow$ | $\text{VQA}_{\text{edit}}\downarrow$ | $\text{VQA}_{\text{id}}\downarrow$ | RMSE$\downarrow$ | MUSIQ$\uparrow$ | $\mathcal{V}_{\text{VQA}}\downarrow$ | $\mathcal{V}_{\text{LPIPS}}\downarrow$ | CV-DSim$\downarrow$ | ★ $\uparrow$ | $\equiv\uparrow$ |
| K. Kontext | 0.078 | 0.430 | 0.161 | 0.021 | 65.34 | 0.047 | 0.116 | 1.153 | 40.3% | 46.8% |
| Diff. Steering | 0.107 | 0.710 | 0.167 | 0.151 | 64.59 | 0.094 | 0.390 | 1.045 | 61.4% | 43.9% |
| FlowSlider | 0.090 | 0.322 | 0.378 | 0.047 | 65.84 | 0.067 | 0.165 | 0.675 | 14.1% | 27.0% |
| VeloEdit | 0.079 | 0.470 | 0.151 | 0.023 | 53.18 | 0.072 | 0.078 | 0.996 | 57.7% | 61.7% |
| SliderEdit | 0.087 | 0.440 | **0.103** | 0.048 | 63.70 | 0.047 | 0.251 | 1.199 | 42.2% | 35.6% |
| Firefly Tune | 0.093 | 0.669 | 0.136 | 0.096 | **68.07** | **0.030** | 0.488 | 1.377 | 49.2% | 52.1% |
| UniSlider | **0.112** | **0.799** | 0.125 | **0.018** | 67.09 | 0.046 | **0.016** | **0.652** | **82.7%** | **82.6%** |

Table 1: **Comparison against baselines.** All metrics are averaged over the 300 benchmark examples. Shading marks the **best** and second-best method per column. ★ marks the method preferred overall in the user study and $\equiv$ the one preferred for uniformity.




Figure 5: **UniSlider results.** From left to right input image, all the intermediate strength levels and the default output of Flux-Klein-4B.

## 6 Experiments

**Benchmark analysis.** Table 1 reports all the slider methods of Section 2 on our benchmark. These methods use different editing backbones, so we normalize the comparison when possible. The training-free methods (FlowSlider, VeloEdit, DiffusionSteering) carry no dataset dependency, and we reimplement them on Flux.2-Klein-4B, the same backbone UniSlider uses. Kontinuous Kontext and SliderEdit are tied to their own training data, which is not publicly available, so we report them on Flux.1-Kontext as published. For SliderEdit, we additionally retrain it on our training set with the Flux.2-Klein-4B backbone (see Appendix E). We also include the closed-source Firefly Tune from Adobe Firefly. UniSlider achieves the best overall performance, consistently ranking either first or second across all metrics, with the largest gains observed in monotonicity and uniformity, the two properties we identified as critical for effective sliders. Note that SliderEdit and UniSlider preserve the editing capability of the base model at full strength, while the other methods degrade it because of their formulation.

**User study.** We also conducted a two-alternative forced-choice (2AFC) user study with 25 observers, with a total of 881 pairs of sliders compared from the 7 methods shown in Table 1 where we ask: "Which method is better overall?" and "Which one spreads the change more evenly?", ★ and $\equiv$, respectively in Table 1. UniSlider wins $82\%$ of the comparisons for both questions as computed using the Bradley-Terry method (Bradley & Terry, 1952), outperforming the second-best method by a large margin in both overall quality and uniformity. We provide further metrics in the supplementary section Appendix E.

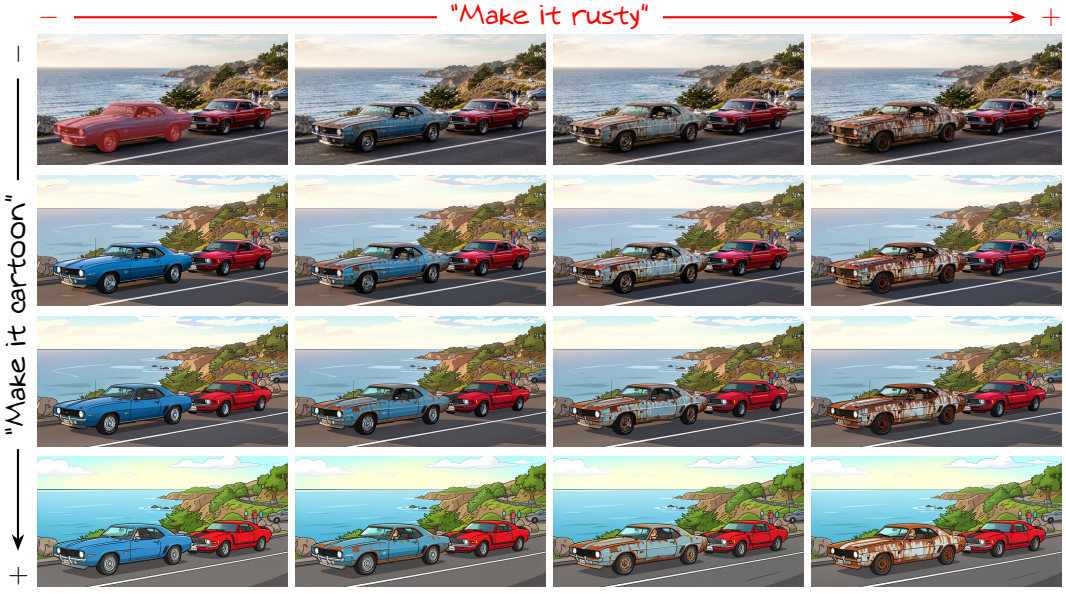


Figure 6: **Composing a masked edit with a global one.** A $4\times4$ grid over two instructions applied to the same input. Strength increases left to right and top to bottom. The second axis is obtained by re-running UniSlider on the output of the first, no composed or masked examples appear in training.

**Qualitative results.** Figure 5 shows UniSlider on three edits: a global one (turning the scene into modelling clay), a local one (making the man more excited), and a structural one (turning on the burner). In all three cases, the intermediate steps are perceptually evenly spaced, with no dead zones or jumps. The clay example reaches a fully sculpted scene while preserving the pose and layout of the crocodile. The facial example alters only the expression, keeping identity, make-up and hair stable at every slider value.The burner example increases the intensity of the flame in perceptually equal steps.

**Simultaneous and localized edits.** Figure 6 combines a masked edit with a global one in a 2D grid. The horizontal axis applies a masked edit restricted to the blue car (the masked region is shown in red in the first cell), while the vertical axis applies a global stylization to the whole scene. We obtain the second axis by composition. We run a forward pass with the first instruction, then feed each output back as the input for the second. UniSlider is trained on a single instruction at a time and never sees composed edits or spatial masks during training, yet the grid remains approximately uniform along both axes and the two edits remain largely independent, moving along one axis neither undoes nor amplifies the other. See Appendix F for additional qualitative results, including masked and 2D edits.

**Training analysis.** UniSlider relies on a perceptual distance $D(\cdot,\cdot)$ to define perceptual uniformity and to train for it. We therefore examine three questions: (i) whether perceptual distances track how humans judge the magnitude of an edit, (ii) how much the choice of $D$ matters for training, and (iii) whether supervising intermediate strengths is necessary or constraining the endpoints alone is enough.

**(i)** Learned perceptual metrics are trained to judge which of two concepts is closer to a reference (Fu et al., 2023; Wang et al., 2026), but our method needs $D$ to measure the magnitude (e.g. be calibrated) of an edit, comparably across edit types. To validate our intended use, we designed a 2AFC study. 15 annotators saw an input image and two edits of it, made with different instructions that represented different strengths, and chose the one that changed more. We correlate the aggregated choices by Bradley–Terry (Bradley & Terry, 1952) with each metric's distance to the input (Table 2, first two columns). PLCC is the key measure, since uniformity requires $D$ to scale linearly with perceived change. Perceptual metrics correlate best (PLCC $0.81$–$0.86$, highest for DreamSim and TPIPS), while pixel RMSE ($0.74$) and a color histogram ($0.51$) lag. This proves that perceptual metrics are a good proxy for measuring strength, specially when compared to pixel-wise metrics.

| Metric | User Study PLCC | User Study SRCC | $D$ in Eq. (4) RMSE$_{@0}\downarrow$ | $D$ in Eq. (4) $\mathcal{V}_{\text{VQA}}\downarrow$ |
|---|---|---|---|---|
| RMSE | 0.740 | 0.724 | 0.063 | 0.067 |
| Histogram | 0.513 | 0.464 | 0.078 | 0.082 |
| LPIPS | 0.836 | 0.827 | 0.021 | 0.048 |
| CLIP | 0.808 | 0.802 | 0.033 | 0.049 |
| DreamSim | **0.862** | 0.855 | **0.018** | **0.046** |
| TPIPS | 0.859 | **0.856** | 0.029 | 0.048 |

Table 2: **Perceptual metric analysis.** 2AFC user study and perceptual metric choice.

| Method | Sampling | Mono. $\mathcal{V}_{\text{VQA}}\downarrow$ | Uniformity CV-DSim$\downarrow$ | Uniformity CV-TPIPS$\downarrow$ |
|---|---|---|---|---|
| VeloEdit | uniform | 0.072 | 0.996 | 0.427 |
| | adaptive | 0.073 | 0.914 | 0.410 |
| SliderEdit | uniform | 0.047 | 1.199 | 0.433 |
| | adaptive | 0.063 | 0.874 | 0.429 |
| UniSlider | uniform | 0.053 | 0.871 | 0.407 |
| | adaptive | **0.046** | **0.652** | **0.372** |

Table 3: **Adaptive sampling analysis.** Different methods with and without adaptive sampling.

**(ii)** We ablate the choice of $D(\cdot, \cdot)$ by replacing it in both the training objective (Eq. (4)) and the adaptive sampling, leaving everything else fixed. The last two columns of Table 2 show the results. DreamSim performs best in both endpoint consistency and monotonicity, followed closely by the other perceptual metrics. Pixel-space RMSE and color-histogram distance fall clearly behind, with three to four times larger error at $s = 0$. We therefore use DreamSim throughout the paper, as it performs best while being efficient.

**(iii)** We train a variant of our model supervised only at the lower endpoint. Mathematically, this corresponds to minimizing $\mathcal{L} = D(x_{\text{src}}, x_{s=0})$, ensuring that the lower endpoint of the slider represents the unedited image. Consequently, the intermediate edit strengths remain unconstrained. This variant does not produce a uniform slider, even when combined with the adaptive sampling at inference. Quantitatively, the produced slider is worse by a factor of 3 in monotonicity and uniformity. This drop is caused because while the endpoints pin the ends of the trajectory, they say nothing about how it is traversed in between, which isolates the contribution of the uniformity term.

**Adaptive sampling analysis.** Table 3 applies our adaptive sampling to other sliders to isolate the contributions of training and remapping. Under DreamSim, the metric used for sampling, remapping improves the uniformity of every method, bringing SliderEdit to the level of UniSlider without remapping. Under TPIPS, which is used in neither training nor sampling, the gain shrinks for the baselines, whereas UniSlider still improves by 9%. Remapping can only equalize a trajectory that is already smooth and monotone, and SliderEdit's is not. Its $\mathcal{V}_{\text{VQA}}$ increases by 34%, because remapping concentrates samples where the change is fastest, which is precisely where the trajectory doubles back. For UniSlider, whose trajectory is optimized to be monotone and uniform, the same procedure instead reduces $\mathcal{V}_{\text{VQA}}$ by 13%. Training and adaptive sampling are therefore complementary. UniSlider without remapping already matches or outperforms every baseline with remapping, and combining both is best overall. Further analysis and additional ablations are reported in Appendices G and H, respectively.

## 7 Conclusions

We have shown that generative editing sliders are not perceptually uniform, and that this is largely a matter of how the strength is parameterized rather than of what the base model can generate. UniSlider separates the slider exposed to the user from the strength that drives the edit. It trains a lightweight LoRA with a pixel-space perceptual objective so that its strength approximates a perceptually uniform slider, and remaps the strength at inference time to recover the uniformity a low-rank adapter cannot represent on its own. The result is a slider whose value has the same meaning across images and edits, as confirmed by our benchmark, which measures uniformity jointly with edit fidelity and identity preservation.

**Limitations.** UniSlider is bounded by its editing backbone as the slider interpolates towards the backbone's default output, so any misinterpreted instruction or artifact is carried along the whole trajectory. Likewise, UniSlider aims to be uniform under $D$ rather than under human perception. Our user study shows that current metrics are a good proxy, but not a perfect one. Adaptive sampling, limited by a fixed rendering budget, approaches uniformity under $D$ without reaching it exactly. See Appendix C for further details. Training an editing model whose strength is perceptually uniform by design, removing the need for remapping, is a natural next step.

## REFERENCES


Eirikur Agustsson and Radu Timofte. NTIRE 2017 challenge on single image super-resolution: Dataset and study. In *CVPR Workshops*, 2017.

Omri Avrahami, Dani Lischinski, and Ohad Fried. Blended diffusion for text-driven editing of natural images. In *CVPR*, 2022.

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

Canberk Baykal, Valentin Deschaintre, Yannick Hold-Geoffroy, Michael Fischer, Anna Frühstück, Cengiz Öztireli, and Iliyan Georgiev. ResEdit: Residual embeddings for precise generative image editing. In *Computer Graphics Forum*, 2026.

Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

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

Muhammad Atif Butt, Diego Hernandez, Alexandra Gomez-Villa, Kai Wang, Javier Vazquez-Corral, and Joost Van De Weijer. NumColor: Precise numeric color control in text-to-image generation. In *ECCV*, 2026.

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

Xi Chen, Lianghua Huang, Yu Liu, Yujun Shen, Deli Zhao, and Hengshuang Zhao. Anydoor: Zero-shot object-level image customization. In *CVPR*, 2024.

Yigit Ekin and Yossi Gandelsman. The unreasonable effectiveness of text embedding interpolation for continuous image steering. *arXiv preprint arXiv:2603.17998*, 2026.

Taichi Endo, Guoqing Hao, and Kazuhiko Sumi. FlowSlider: Training-free continuous image editing via fidelity-steering decomposition. *arXiv preprint arXiv:2604.02088*, 2026.

Stephanie Fu, Netanel Tamir, Shobhita Sundaram, Lucy Chai, Richard Zhang, Tali Dekel, and Phillip Isola. DreamSim: Learning new dimensions of human visual similarity using synthetic data. In *NeurIPS*, 2023.

Rinon Gal, Or Patashnik, Haggai Maron, Amit H Bermano, Gal Chechik, and Daniel Cohen-Or. StyleGAN-nada: Clip-guided domain adaptation of image generators. *ACM Transactions on Graphics (ToG)*, 41(4):1–13, 2022.

Rohit Gandikota, Joanna Materzyńska, Tingrui Zhou, Antonio Torralba, and David Bau. Concept sliders: LoRA adaptors for precise control in diffusion models. In *ECCV*, 2024.

Julia Guerrero-Viu, Milos Hasan, Arthur Roullier, Midhun Harikumar, Yiwei Hu, Paul Guerrero, Diego Gutierrez, Belen Masia, and Valentin Deschaintre. TexSliders: Diffusion-based texture editing in clip space. In *SIGGRAPH*, 2024.

Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross-attention control. In *ICLR*, 2023.

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

Xuan Ju, Ailing Zeng, Yuxuan Bian, Shaoteng Liu, and Qiang Xu. PnP inversion: Boosting diffusion-based editing with 3 lines of code. In *ICLR*, 2024.

Bahjat Kawar, Shiran Zada, Oran Lang, Omer Tov, Huiwen Chang, Tali Dekel, Inbar Mosseri, and Michal Irani. Imagic: Text-based real image editing with diffusion models. In *CVPR*, 2023.

Junjie Ke, Qifei Wang, Yilin Wang, Peyman Milanfar, and Feng Yang. Musiq: Multi-scale image quality transformer. In *ICCV*, 2021.

Nupur Kumari, Sheng-Yu Wang, Nanxuan Zhao, Yotam Nitzan, Yuheng Li, Krishna Kumar Singh, Richard Zhang, Eli Shechtman, Jun-Yan Zhu, and Xun Huang. Learning an image editing model without image editing pairs. In *ICLR*, 2026.

Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, et al. Flux. 1 kontext: Flow matching for in-context image generation and editing in latent space. *arXiv preprint arXiv:2506.15742*, 2025.

Zongqing Li, Zhihui Liu, Yujie Xie, Shansiyuan Wu, Hongshen Lv, and Songzhi Su. VeloEdit: Training-free consistent and continuous instruction-based image editing via velocity field decomposition. *arXiv preprint arXiv:2603.13388*, 2026.

Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *CVPR*, 2022.

Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. In *ICLR*, 2022.

Anish Mittal, Rajiv Soundararajan, and Alan C Bovik. Making a "completely blind" image quality analyzer. *IEEE Signal Processing Letters*, 20(3):209–212, 2012.

Xingang Pan, Ayush Tewari, Thomas Leimkühler, Lingjie Liu, Abhimitra Meka, and Christian Theobalt. Drag your GAN: Interactive point-based manipulation on the generative image manifold. In *SIGGRAPH*, 2023.

Rishubh Parihar, Or Patashnik, Daniil Ostashev, Venkatesh Babu Radhakrishnan, Daniel Cohen-Or, and Kuan-Chieh Jackson Wang. Kontinuous kontext: Continuous strength control for instruction-based image editing. In *CVPR*, 2026.

Sam Sartor, Iliyan Georgiev, Michael Fischer, Valentin Deschaintre, and Pieter Peers. Overpainting: Localized context-aware diffusion image editing. *arXiv preprint arXiv:2609.10811*, 2026.

Prafull Sharma, Varun Jampani, Yuanzhen Li, Xuhui Jia, Dmitry Lagun, Fredo Durand, Bill Freeman, and Mark Matthews. Alchemist: Parametric control of material properties with diffusion models. In *CVPR*, pp. 24130–24141. IEEE, 2024.

Shelly Sheynin, Adam Polyak, Uriel Singer, Yuval Kirstain, Amit Zohar, Oron Ashual, Devi Parikh, and Yaniv Taigman. Emu edit: Precise image editing via recognition and generation tasks. In *CVPR*, 2024.

Yujun Shi, Chuhui Xue, Jun Hao Liew, Jiachun Pan, Hanshu Yan, Wenqing Zhang, Vincent YF Tan, and Song Bai. DragDiffusion: Harnessing diffusion models for interactive point-based image editing. In *CVPR*, 2024.

Zhenxiong Tan, Songhua Liu, Xingyi Yang, Qiaochu Xue, and Xinchao Wang. OminiControl: Minimal and universal control for diffusion transformer. In *ICCV*, 2025.

Jianyi Wang, Kelvin CK Chan, and Chen Change Loy. Exploring clip for assessing the look and feel of images. In *AAAI*, 2023.

Sheng-Yu Wang, Yotam Nitzan, Aaron Hertzmann, Jun-Yan Zhu, Eli Shechtman, Alexei A Efros, and Richard Zhang. The many senses of visual similarity: A text-prompted image perceptual metric. *arXiv preprint arXiv:2607.18237*, 2026.

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng-ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, et al. Qwen-image technical report. *arXiv preprint arXiv:2508.02324*, 2025.

Arman Zarei, Samyadeep Basu, Mobina Pournemat, Sayan Nag, Ryan A Rossi, and Soheil Feizi. SliderEdit: Continuous image editing with fine-grained instruction control. In *CVPR*, 2026.

Kai Zhang, Lingbo Mo, Wenhu Chen, Huan Sun, and Yu Su. MagicBrush: A manually annotated dataset for instruction-guided image editing. *NeurIPS*, 2023a.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *ICCV*, 2023b.

Zicheng Zhang, Tengchuan Kou, Shushi Wang, Chunyi Li, Wei Sun, Wei Wang, Xiaoyu Li, Zongyu Wang, Xuezhi Cao, Xiongkuo Min, et al. Q-eval-100k: Evaluating visual quality and alignment level for text-to-vision content. In *CVPR*, 2025.

## A LLM USAGE STATEMENT

We used LLMs for grammar correction and to rewrite and polish parts of the manuscript. LLMs also assisted with writing code, and Qwen3-VL-30B (Bai et al., 2025) was used to generate the text instructions of our training dataset (Section 3). All LLM outputs were reviewed by the authors, who verified their correctness and revised them where necessary. The authors take full responsibility for the content of this paper.

## B FURTHER RELATED WORK DISCUSSION

Section 2 covers slider methods for continuous image editing. Due to space constraints, we extend that discussion here with the broader trends in text-based image editing that our backbone builds on, and with interactive editing paradigms that give users control along axes other than strength.

**Text-based image editing.** Early text-driven editors operated by inverting a real image into a pre-trained diffusion model's noise space and then manipulating the generation process directly through cross-attention injection (Hertz et al., 2023), partial re-noising and re-denoising (Meng et al., 2022), or latent optimization (Kawar et al., 2023). These methods require no editing-specific training data and can be strikingly precise, but each edit costs a fresh inversion and often per-image optimization, and the target has to be specified as a new caption or embedding rather than a direct instruction. Instruction-tuned editors removed this overhead by training a single feed-forward model on (image, instruction, edited image) triples: InstructPix2Pix (Brooks et al., 2023) bootstrapped such a dataset from GPT-3 and Prompt-to-Prompt, MagicBrush (Zhang et al., 2023a) replaced synthetic triples with human-annotated ones and added multi-turn edits, and Emu Edit (Sheynin et al., 2024) scaled this recipe across a wide range of edit types with task embeddings. These editors generalize well and require only a single instruction at inference, but still rely on a many-step diffusion sampler, and expose no notion of edit strength: the model produces one edit, not a family of them. The most recent instruction-based editors move this recipe onto rectified-flow diffusion transformers with few-step distillation (Labs et al., 2025; Wu et al., 2025), which is what makes decoding to pixel space inside the training loop (central to our method, Section 3) affordable in the first place.

**Interactive image editing.** A separate line of work gives users control over where or how an edit happens, rather than only what it is. Spatial conditioning methods such as ControlNet (Zhang et al., 2023b) or OminiControl (Tan et al., 2025) let a user supply a sketch, edge map, or depth map alongside the prompt. Mask-based editors (Avrahami et al., 2022; Lugmayr et al., 2022; Sartor et al., 2026) instead let the user restrict an edit to a region, leaving the rest of the image untouched; UniSlider extends this idea to a soft mask rather than a binary one to control the locality of an edit. Point-based editing takes this further. DragGAN (Pan et al., 2023) and DragDiffusion (Shi et al., 2024) let a user drag a handle point on an image toward a target location. Other work gives the user control over scene attributes directly such as the exact color (Butt et al., 2026) or material (Baykal et al., 2026) of an object, or which object appears at all (Chen et al., 2024). Together these give the user control over the location, geometry, and attributes of an edit. None of them, however, address its magnitude. Once a region, a point, or an attribute value is fixed, the edit is still applied all-or-nothing. This is the axis sliders operate on, and where UniSlider is positioned (see Section 2)

# C FURTHER METHOD DETAILS

**Implementation details for reproduction.** We train a single LoRA adapter on the attention projections of every transformer block of FLUX.2-Klein[1], rank $r = 32$ at 4B and $r = 16$ at 9B. The strength $s$ is not a network input but the adapter scale, applied as $(1 - s)$. At $s = 1$ the adapter vanishes and the frozen backbone produces its own edit, which serves as the training target. Supervision therefore requires only (image, instruction) pairs, never ground-truth intermediates.

We train for 2000 steps on our training dataset with AdamW, learning rate $10^{-4}$, gradient clipping at 1.0, batch size 1, and a 768-pixel long side; each step draws one example and one $s \sim \mathcal{U}(0, 1]$. One choice matters for reproduction. The noise vector is drawn once per example and reused at every $s$ at inference time, so a sweep is one trajectory. Inference uses four steps with $\sigma_{\text{end}} = 1/28$. Training a 4B adapter takes about two hours on one NVIDIA RTX PRO 6000 and yields an 8 MB adapter.

Frames shown for UniSlider are placed by the adaptive sampling algorithm. Other baselines use a uniform grid over their own documented control range or parameter. All methods are compared at 11 frames.

**Adaptive sampling.** Let $\phi(s) = D(x_{\text{src}}, x_s)/D(x_{\text{src}}, x_{\text{edit}})$ be the fraction of the total perceptual distance covered at strength $s$, measured with DreamSim. A perceptually uniform slider satisfies $\phi(s) = s$, so for $n$ frames we want samples at $\phi = 0, \frac{1}{n-1}, \ldots, 1$. Since $\phi$ can only be observed by decoding images to pixel space, every evaluation costs a generation and we must work within a fixed budget.

We start from the two endpoints, $x_{\text{src}}$ and the default edit $x_{\text{edit}}$, and render the midpoint $x_{s=0.5}$. We then measure $D$ between adjacent frames and repeatedly split the largest gap in $\phi$ until the budget is exhausted. Rather than splitting at the midpoint in $s$, we fit a monotone PCHIP spline through the measurements of $\phi$, invert it to locate the *perceptual* midpoint of that gap, and render there. This targets the strength at which the trajectory is expected to be halfway through the gap in perceptual terms, which is where a uniform slider would place its next stop.

Measuring $\phi$ during sampling has a second benefit We can detect when a newly rendered frame is perceptually too close to one we already have and leave it out. Users are better served by fewer frames than by a slider with visually repeated stops. We therefore discard a new point for display if it falls within $\varepsilon$ of a neighbor in $\phi$, while keeping it as a measurement so that it still sharpens the spline. We set $\varepsilon$ as a fraction of $D(x_{\text{src}}, x_{\text{edit}})$ (0.03) rather than as an absolute distance, since $D$ varies by over an order of magnitude across edits. Algorithm 1 summarizes the procedure.

Adaptive sampling adds no parameters and no training. It requires only the ability to render at an arbitrary strength and a distance with which to measure, so it applies to any continuous editor, and we show in Section 6 that it also improves the baselines. Its one assumption is that $\phi$ is monotone. If a trajectory doubles back, the largest gap has no well-defined perceptual midpoint and placement degrades towards uniform sampling. The overhead is negligible in practice. Rendering 11 frames with Flux-Klein-4B takes 7.02 s on average and 7.16 s with adaptive sampling, an increase of 2%, measured on a NVIDIA RTX Pro 6000.

**Limitations.** We expand on the three limitations discussed in Section 7.

UniSlider interpolates between the input and the default edit of the frozen backbone, so it cannot correct the backbone's failures. Figure 7 shows an example with Flux.2-Klein-4B. Besides inflating the ball, the default edit changes its colors, and since every intermediate frame moves towards this output, the unrequested color change is carried along the whole slider. Conversely, UniSlider benefits directly from stronger backbones, since adapting it to a new few-step editor only requires retraining the adapter (Appendix H).

Our slider is uniform under $D$, which is only a proxy for human perception. DreamSim is among the metrics that best correlate with human judgments of edit magnitude (Table 2), but the correlation is not perfect, and any disagreement translates directly into non-uniformity as perceived by the user. In Fig. 7, DreamSim cannot separate the requested change (inflating the ball) from unrelated ones. It places the second frame on the diagonal because a white patch on the right part of the ball has turned

[1] `to_q`, `to_k`, `to_v`, `to_out.0`, `add_q_proj`, `add_k_proj`, `add_v_proj`, `to_add_out`.

**Algorithm 1** Adaptive sampling of the strength axis

**Require:** input $x_{\text{src}}$, default edit $x_{\text{edit}}$, trained adapter, budget $M$, threshold $\varepsilon$
**Ensure:** strengths $s_1 < \cdots < s_n$, shown at slider values $u_k = \frac{k-1}{n-1}$

$x_0 \leftarrow x_{\text{src}}$, $x_1 \leftarrow x_{\text{edit}}$, $\mathcal{S} \leftarrow \{0, 1\}$
render $x_{0.5}$; $\mathcal{S} \leftarrow \mathcal{S} \cup \{0.5\}$; $\mathcal{S}_{\text{disp}} \leftarrow \mathcal{S}$
**while** $|\mathcal{S}| < M$ **do**
  $(s_a, s_b) \leftarrow$ adjacent pair in $\mathcal{S}$ with the largest $\phi(s_b) - \phi(s_a)$
  $\hat{\phi} \leftarrow$ monotone PCHIP fit to $\{(s, \phi(s))\}_{s \in \mathcal{S}}$
  $s_{\text{new}} \leftarrow \hat{\phi}^{-1}\left(\frac{1}{2}(\phi(s_a) + \phi(s_b))\right)$ ▷ perceptual midpoint
  render $x_{s_{\text{new}}}$, measure $\phi(s_{\text{new}})$; $\mathcal{S} \leftarrow \mathcal{S} \cup \{s_{\text{new}}\}$
  **if** $\min_{s \in \mathcal{S}_{\text{disp}}} |\phi(s_{\text{new}}) - \phi(s)| \geq \varepsilon$ **then**
    $\mathcal{S}_{\text{disp}} \leftarrow \mathcal{S}_{\text{disp}} \cup \{s_{\text{new}}\}$ ▷ otherwise, measurement only
  **end if**
**end while**
**return** $\mathcal{S}_{\text{disp}}$ in increasing order




Figure 7: **Backbone and DreamSim failure case.** The backbone's default edit changes the colors of the ball, and this change is carried along the whole slider. DreamSim places the second frame on the diagonal because of a small color change on the right of the ball, although the ball is not yet inflated.

red, although the ball is still deflated. The slider is thus uniform under DreamSim, but a human observer sees a dead zone. Text-conditioned metrics such as TPIPS could, in principle, measure only the change related to the instruction, but with 14 billion parameters, TPIPS is too expensive for interactive use. Since UniSlider is agnostic to the choice of $D$ in both training and sampling, it would directly benefit from more efficient text-conditioned or better-calibrated perceptual metrics.

Adaptive sampling estimates the remapping $g$ from a fixed budget of renders (typically the number of frames shown by the slider), so it approaches uniformity under $D$ without reaching it exactly. To measure the remaining gap, we compare it with a dense reference. We render the trajectory at 51 equispaced strengths, fit a monotone PCHIP spline to $\phi(s)$, invert it at the target slider values $u_k = k/10$, and render the resulting frames, for a total of 60 renders. Because it samples the trajectory densely, this reference closely approximates the exact remapping. Figure 8 compares both on one edit. The dashed red curve is the fitted $\phi(s)$, which corresponds to our Vanilla slider without remapping ($s = u$). The blue curve shows adaptive sampling with only 11 renders, and the solid red curve shows the frames rendered by the dense reference at each $u_k$. Note that we only show five images due to space constraints. The Vanilla slider already stays relatively close to the diagonal but follows an S-shape. Adaptive sampling brings the curve much closer to the diagonal, and the dense reference matches it almost exactly. For instance, its frame at $u = 0.5$ lies exactly halfway, in perceptual distance, between the input and the default edit. The remaining gap of adaptive sampling is small compared with the non-uniformity of the Vanilla slider. The frames produced by adaptive sampling are also visually very similar to those of the dense reference, while requiring less than one-fifth as many renders. Adaptive sampling therefore offers a favourable trade-off for interactive use, and its budget can be increased when exact uniformity matters more than latency.

## D Further training dataset details

Training needs only (image, instruction) pairs: the edited image at $s$=1 is the frozen backbone's default output, so no edited image, no intermediate, and no human annotation is required. We assemble 300 such pairs from MIT-Adobe-FiveK (Bychkovsky et al., 2011). Source photographs

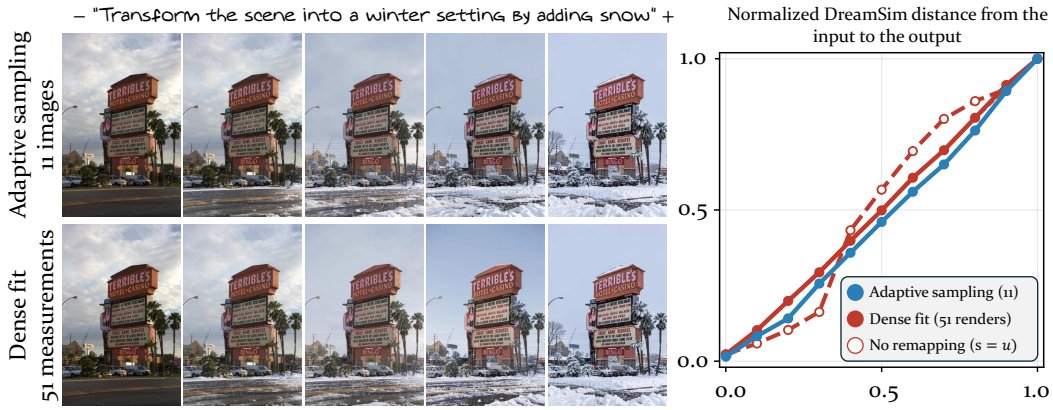


Figure 8: **Adaptive sampling vs. dense reference. Left:** five of the eleven frames obtained with adaptive sampling (top) and with the dense reference, which inverts a PCHIP spline fitted to 51 measurements (bottom). **Right:** normalized DreamSim distance from the input to each frame against the slider value $u$. Without remapping (dashed), adaptive sampling (blue) and dense fit (red).

are drawn from the FiveK pool after removing every image used by our benchmark, so the training set and the evaluation set share no image.

Instructions are generated through a three-stage VLM pipeline. Each sampled photograph is first described neutrally and screened for content sufficiency: blank walls, plain sky, water surfaces and abstract textures are discarded, since they offer nothing an edit can act on. Surviving images are paired with a concept seed sampled from a fixed pool of 500 seeds spanning art medium, artist style, cinematic and videogame looks, color tone and art movement (global); person and object attributes; and count increase, addition or resizing, atmosphere, vegetation and urban density (structural). A second stage writes an instruction conditioned on the image and its seed, and a judge stage accepts or rejects it.

One constraint on instruction wording is specific to this setting and worth stating. The instruction must not encode any degree, range, percentage or mechanism. No *"from 1 to 5"*, no *"by 30%"*, no *"slightly"*. Strength is carried entirely by $s$, and an instruction that names its own intensity would make the text and the slider compete to specify the same quantity. The resulting instructions are correspondingly short: a median of eight words (range 4–17), e.g. *"Add more cars to the street"*.

We over-generate and filter. 540 candidates against a target of 300, leaving room for rejections. Figure 9 shows nine examples of the image–text instruction pairs of our training dataset.

## E Further benchmark details

**Motivation.** Current slider methods are evaluated on small, method-specific test sets that make cross-paper comparison unreliable. Kontinuous Kontext (Parihar et al., 2026) evaluates on a held-out split of its own morphing dataset, which is unfair to methods that were not trained on that kind of data. SliderEdit (Zarei et al., 2026) builds a facial-editing benchmark, which is too domain-specific to say anything about performance elsewhere. Neither benchmark is public. The alternative some methods take is to reuse an existing image-editing benchmark: FlowSlider (Endo et al., 2026) and VeloEdit (Li et al., 2026) both draw images from PIE-Bench (Ju et al., 2024), but PIE-Bench targets one-shot edits and includes operations such as object removal or addition that have no meaningful intermediate strength as an object is either there or it is not. Reusing it for a slider benchmark inherits that mismatch. A further, more practical problem cuts across all of these: training-free methods are steered by a pair of prompts (positive and negative), while trained sliders take a single instruction, so any shared benchmark has to supply both prompt formats for every example or silently favor one family of methods over the other. No existing benchmark does this. Together these gaps (small and non-public test sets, evaluation domains too narrow to generalize from, and a benchmark format built for a different kind of edit) are why we build a dedicated benchmark for continuous image editing.

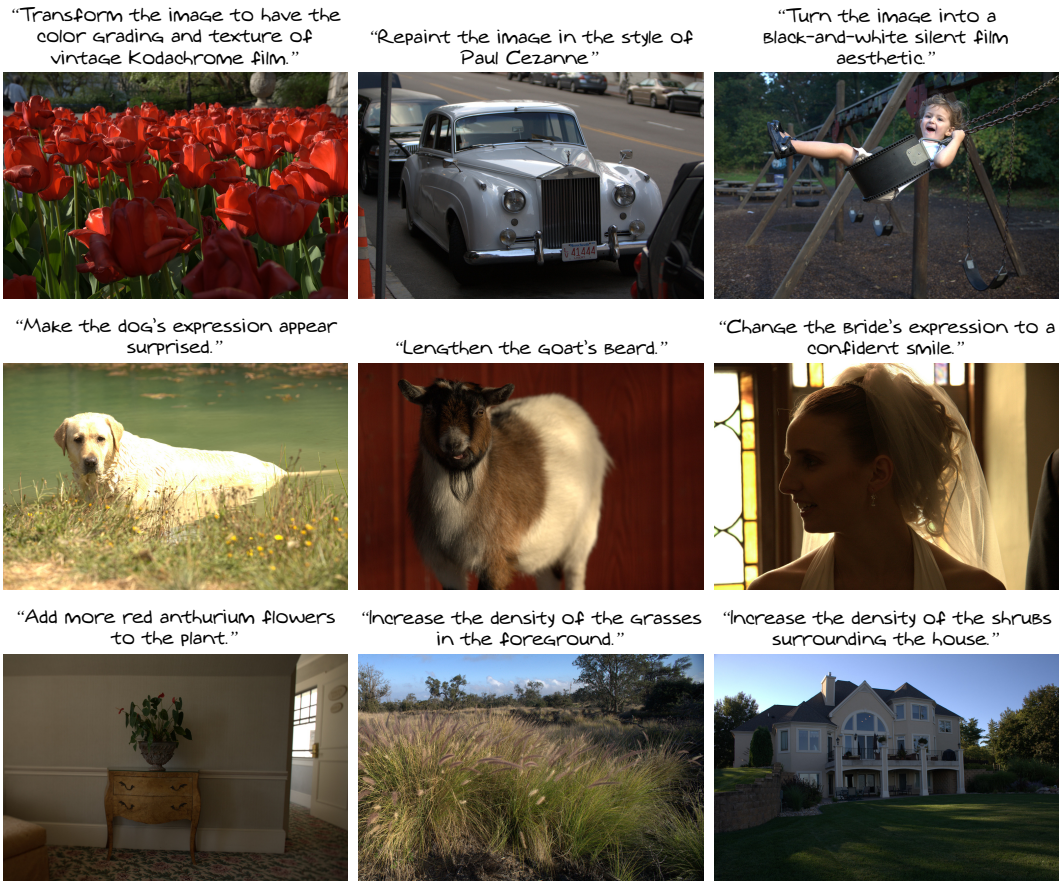


Figure 9: **Training data.** Nine of the 300 (image, instruction) pairs. These two things are the entire supervision, the target at $s{=}1$ is the frozen backbone's default edit, so no edited image and no intermediate is required. Instructions carry no degree or mechanism (no "slightly", no "by 30%"), since strength is carried entirely by $s$; they are correspondingly short, a median of eight words.

**Images and instructions.** The benchmark holds 300 examples, 100 each of *global*, *attribute* and *structural* edits. Source photographs come from MIT-Adobe FiveK (Bychkovsky et al., 2011) at up to 1600 pixels on the long side, in both orientations, and are disjoint from the training set. The training data was sampled only after every benchmark image had been excluded, so no photograph appears in both. We only use the MIT-Adobe-FiveK as a good pool of high-quality and diverse images.

Each example carries an instruction, and neutral descriptions of the input and of the intended result for training-free methods. Benchmark instructions are markedly longer and more specific than the training ones, a median of 18 words against 8, because they must pin down a single intended outcome precisely enough to be scored in the edit fidelity section. We show examples from our benchmark when comparing UniSlider with other slider methods.

**Metrics.** Every method is rendered at $K = 11$ slider positions $s_0 < s_1 < \cdots < s_{10}$, normalized so that $s_0$ is the method's minimum strength and $s_{10}$ its full edit. We write $x_{\text{src}}$ for the input image and $x_{s_k}$ for a method's own output at position $s_k$.

Because methods render at different native resolutions ($768 \times 512$ to $1600 \times 1067$), every image is resampled to a common 768-pixel long side before any metric is computed. Without this, pixel-space and no-reference quality scores would partly measure render size rather than edit quality.

A benchmark for continuous editing needs to characterize six properties, and we group the metrics accordingly:

- **Edit fidelity.** Does the intended edit happen?
- **Identity preservation.** Is everything else preserved?
- **Endpoint consistency.** Does the slider return the input unchanged at its minimum setting?
- **Image quality.** Are the output images free of degradation?
- **Strength monotonicity.** Does the slider move consistently in one direction?
- **Perceptual uniformity.** Are its increments perceptually even?

None of these six questions has a single correct answer. A continuous edit has no ground truth as infinitely many intermediate trajectories satisfy the same instruction at the same strength, and no reference image exists to compare against. This is why every metric used is either no-reference or judges a property directly from the pair of images or the trajectory itself, rather than measuring distance to a fixed target the way conventional image editing benchmarks do. With these six questions in hand, we now define the metrics used for each.

**Edit fidelity: CLIP directional similarity ($\mathrm{CLIP}_{\mathrm{dir}} \uparrow$).** Following Gal et al. (2022) and other slider methods (Parihar et al., 2026; Zarei et al., 2026; Endo et al., 2026), we measure whether the image moves in the direction the text asks for, rather than merely towards the target caption:

$$\mathrm{CLIP}_{\mathrm{dir}}(s) = \cos\Big(E_I(x_s) - E_I(x_0),\ E_T(c_{\mathrm{edit}}) - E_T(c_{\mathrm{src}})\Big), \tag{7}$$

with $E_I, E_T$ the CLIP image and text encoders and $c_{\mathrm{src}}, c_{\mathrm{edit}}$ the source and edited captions. By default, $\mathrm{CLIP}_{\mathrm{dir}}$ reports the value at full strength ($s_{10}$). However, we also use edit fidelity metrics at all strength values to measure strength monotonicity. It is included for comparability with prior work, but it is a weak discriminator, insensitive to whether the edit is localized or plausible.

**Edit fidelity: VQA edit success ($\mathrm{VQA}_{\mathrm{edit}} \uparrow$).** Vision–language models are increasingly used as evaluators for fine-grained visual tasks. Kumari et al. (2026) showed they can optimize an editing model without ground truth data from nothing but two questions: whether the instructed edit was executed, and whether the edited image preserves the identity and content of the input image. We reuse this idea, but only for evaluation. A VLM (`Qwen3-VL-8B-Instruct`, Bai et al. (2025)) is shown the input and the edited image together and asked whether the instructed change has been applied. We compute the probability from the "Yes" token $P(\mathrm{Yes})$ directly from the first-token logits rather than sampling an answer, which is deterministic, costs a single forward pass, and gives a continuous score instead of a binary one.

We use the text prompt: "*The first image is the original. The second is an edited version of it. Has this edit been applied to the second image: {instruction}"? Answer with only Yes or No.*", and compute the scores by:

$$\mathrm{VQA}_{\mathrm{edit}} = P(\mathrm{Yes} \mid q_{\mathrm{edit}}). \tag{8}$$

**Identity preservation: VQA collateral change ($\mathrm{VQA}_{\mathrm{id}} \downarrow$).** Similarly to Kumari et al. (2026) and for the $\mathrm{VQA}_{\mathrm{edit}}$ metric, we use the same model, on the same image pair, but this time we ask wether anything other than the instructed edit has changed. Specifically the prompt is: "*The first image is the original. The second is an edited version of it. Besides the intended edit ("instruction"), has anything ELSE in the image changed — the background, the layout, other objects, the colors, or the identity of the subject? Answer with only Yes or No.*" Note the polarity, we ask for evidence of damage instead of preservation. We do so, because the positive form ("is the rest unchanged?") saturates, as the model will call almost anything unchanged, and separates a strong from a weak-identity method by only a few percent.

**Endpoint consistency: Identity at zero strength ($\mathrm{RMSE}@s{=}0 \downarrow$).** The root-mean-square pixel error between $x_{s=0}$ and $x_{\mathrm{src}}$. A continuous editor should return the input unchanged at zero strength. Any residual error is an offset the user cannot remove.

**Image quality.** Continuous editors can satisfy every other metric while degrading the image, so we add no-reference quality measures evaluated at all strength levels and averaged. In Table 1 we report MUSIQ (Ke et al., 2021), a multi-scale transformer predictor with strong agreement with

human opinion scores. However, in Table 5 we also report CLIP-IQA (Wang et al., 2023), which responds to semantic degradation that MUSIQ can miss. We do not use NIQE (Mittal et al., 2012) as it is tuned for compression and blur statistics and scores diffusion artifacts poorly.

**Strength Monotonicity: Violation rates ($\mathcal{V}\downarrow$).** A slider should move steadily towards the edit. A trajectory that doubles back is a control the user cannot reason about, and, unlike uneven pacing, it cannot be repaired by re-sampling the slider positions. For a sequence of images from an slider $s_0, \ldots, s_{10}$, we obtain their corresponding metric values $v_0, \ldots, v_{10}$ that should ideally increase with $s$ monotonically. We report the violation rate of the metric values $v_k$:

$$\mathcal{V}(v) = \frac{1}{K-1} \left| \{\, k : v_{k+1} - v_k < -\tau \,\} \right|, \tag{9}$$

with a small dead-band $\tau$ so that noise around a flat segment is not counted.

We compute the violation rate on the VQA edit success ($\mathcal{V}_{\text{VQA}}$) and LPIPS ($\mathcal{V}_{\text{LPIPS}}$) on Table 1. For the former, each $v_k$ the metric computes a probability of edit success, while for the latter we compute the LPIPS$(x_{\text{src}}, x_k)$, which should also increase monotonically. We report the violation rate on the DreamSim values ($\mathcal{V}_{\text{DreamSim}}$) and using the Q-Eval metric ($\mathcal{V}_{\text{Qeval}}$) (a metric that measures image quality and alignment with text instructions) in Table 5.

**Perceptual uniformity: Coefficient of variation (CV$\downarrow$).** The paper's central claim is that a slider should behave like a physical dial, changing the image proportionally to how far the handle moves. In generative sliders this change is not directly measurable, so we quantify it with a perceptual metric. With consecutive-frame distances $\Delta_k = D(x_{s_k}, x_{s_{k+1}})$, a perfectly uniform slider has all $\Delta_k$ equal, so we report their coefficient of variation

$$\text{CV}(\Delta) = \frac{\sigma(\Delta)}{\mu(\Delta)}, \tag{10}$$

which is scale-free and therefore comparable across methods whose total edit magnitudes differ. Equivalently, the cumulative curve $D(x_{\text{src}}, x_s)/D(x_{\text{src}}, x_{\text{edit}})$ plotted against $s$ should follow the diagonal, as in Fig. 4. Flat segments are dead zones and steep segments are jumps.

We report CV under DreamSim because as seen in Table 2 is the perceptual metric that relates closer to human perception. However, to avoid circularity as UniSlider optimizes directly to reduce $\text{CV}_{\text{DSim}}$, we also report $\text{CV}_{\text{TPIPS}}$ in Table 5.

Uniformity must never be read without fidelity. A method that changes almost nothing has trivially even increments and will top this column while failing to edit at all. We observed exactly this failure mode among the baselines. We therefore always present CV beside $\text{VQA}_{\text{edit}}$, and regard a uniformity score as meaningful only above a minimum fidelity. This occurs for other metrics as well and, that is why, the benchmark reports all six properties together.

**SliderEdit Flux.2-Klein implementation.** SliderEdit (Zarei et al., 2026) exposes edit strength through a low-rank adapter whose scale acts as a signed slider. We use the *global* variant (GST-LoRA), which allocates a single slider to the whole instruction and is the setting matching our benchmark, where every example carries one instruction. The objective is self-supervised and requires no ground-truth edited images. The adapter is trained so that the velocity predicted with the adapter active and the edit instruction matches the velocity predicted by the frozen model under the neutral instruction "keep the image the same",

$$\mathcal{L} = \left\| T_{\theta+\Delta}(z_t, t, c) - T_\theta(z_t, t, c_\varnothing) \right\|_2^2, \qquad z_t = (1-\sigma_t)\, z_{\text{img}} + \sigma_t\, \epsilon, \tag{11}$$

where $\Delta$ denotes the adapter at scale 1 and $c_\varnothing$ the neutral instruction. The adapter therefore learns to *cancel* the edit, which fixes the sign convention of the slider: $\alpha = +1$ suppresses the edit, $\alpha = 0$ recovers the frozen model's own edit, and $\alpha < 0$ over-drives it. At evaluation we map benchmark strength $s \in [0, 1]$ to $\alpha = 1 - 2s$, so that $s = 0$ is maximal suppression and $s = 1$ maximal amplification, with the frozen edit at the midpoint; the $s = 0$ frame is rendered by the model rather than copied from the input. A single noised timestep is scored per step, so no sampling trajectory is unrolled.

The original work adapts FLUX.1-Kontext and Qwen-Image-Edit. To separate the contribution of the slider formulation from that of the backbone, we retrain it on FLUX.2-Klein-4B, the same base

model, training set and schedule used by our method. We keep the published hyper-parameters: rank-16 LoRA (3.9M parameters) on the eight attention projections ($W_q, W_k, W_v, W_o$ and their context counterparts), learning rate $5 \times 10^{-5}$, gradient clipping at 1.0, and 1000 optimisation steps at an effective batch size of 6, trained in about one hour on a single NVIDIA A40 80GB GPU. Three changes follow from the backbone and the evaluation protocol. First, FLUX.2-Klein-4B is not guidance-distilled, so the guidance scale of 3.5 used for FLUX.1-Kontext has no corresponding input and is omitted. Second, Klein encodes prompts without pooled projections and assembles latents through a different packing path, so the conditioning tensors are rebuilt accordingly while playing identical roles in the loss. Third, the reference implementation center-crops inputs to $1024^2$ so that a batch can be stacked; because our benchmark requires the input aspect ratio to be preserved and our training images are of mixed orientation, we instead train at batch size 1 with gradient accumulation 6 on aspect-preserving resizes, which keeps the effective batch unchanged while avoiding a train/test mismatch in field of view.

# F Additional results

**Additional benchmark quantitative results.** Table 4 reports the benchmark results separated by edit type. Here we also include our own implementation of SliderEdit on Flux-Klein-4B ($^K$), which shares the backbone with UniSlider and therefore returns the same image at full strength. Under this scenario, the two differ in what they optimize: SliderEdit supervises the velocity estimates, whereas UniSlider imposes uniformity directly in pixel space. UniSlider obtains the best or second-best result on every edit category and metric except two. As noted above, these metrics should not be read in isolation, since a method can score well on one of them while producing a poor slider overall. What matters is that UniSlider is consistently at the top across all of them.

Because continuous image editing is highly ill-posed and offers no ground truth to compare against, we report a further set of metrics in Table 5. For edit fidelity we add Q-Eval (Zhang et al., 2025), which jointly measures visual quality and alignment with the text instruction, together with its violation rate ($\mathcal{V}_{\text{Qeval}}$). We also report a combined score that multiplies the two VQA-based metrics, $\text{VQA}_{\text{edit}} \times (1 - \text{VQA}_{\text{id}})$ ($\uparrow$), so that a method is rewarded only when it performs the requested edit and leaves the rest of the image intact. UniSlider performs best on all of these metrics.

**Benchmark qualitative results.** Figure 12 shows four benchmark examples under several methods, annotated with per-frame metrics. The top example compares UniSlider with Firefly Tune, reporting $\text{VQA}_{\text{edit}}$ and $\text{VQA}_{\text{id}}$ at every frame. Both track the state of the edit closely. Firefly Tune struggles to build up the snow coverage and its $\text{VQA}_{\text{edit}}$ jumps from 0.0 to 1.0 near the end of the range, whereas UniSlider transitions smoothly. On $\text{VQA}_{\text{id}}$, Firefly Tune introduces a color shift at maximum strength and thus rises the metric value to 0.2, while the metric stays flat throughout for UniSlider. Recall that $\text{VQA}_{\text{id}}$ asks the VLM whether anything *other* than the instructed edit has changed, so lower is better and 0.0 is ideal.

The second example compares UniSlider with FlowSlider and DiffusionSteering. FlowSlider introduces substantial artifacts, particularly at low strengths, which is visible in its $\text{VQA}_{\text{id}}$ curve. DiffusionSteering fails to interpolate between the input and the edit at all and its $\text{VQA}_{\text{edit}}$ stays high across the entire range. UniSlider yields a smooth interpolation, resulting in a smooth and monotonic $\text{VQA}_{\text{edit}}$ curve while keeping $\text{VQA}_{\text{id}}$ flat.

The bottom two examples compare UniSlider with Firefly Tune and SliderEdit, plotting the perceptual distance from the input at each frame. Both baselines show dead zones and jumps, while UniSlider stays close to the ideal diagonal.

**Benchmark user study details.** We ran a two-alternative forced-choice (2AFC) study comparing the seven methods. On each trial, a participant saw two full 11-frame strips of the same example, produced by two different methods, with method identities hidden, and left/right position randomised. Participants were asked to evaluate the results by answering two questions: which slider is better overall and which slider strip is more evenly spread. The latter measures the property central to this work, while the former is left deliberately unconstrained, so that a method winning on uniformity alone cannot also claim general preference by construction. In total, 25 participants contributed 881 pairwise judgments.

| | Method | Edit Fidelity | | Identity | Endpoint | IQA | Monotonicity | | Uniformity |
|---|---|---|---|---|---|---|---|---|---|
| | | $\text{CLIP}_{\text{dir}}\uparrow$ | $\text{VQA}_{\text{edit}}\downarrow$ | $\text{VQA}_{\text{id}}\downarrow$ | RMSE↓ | MUSIQ↑ | $\mathcal{V}_{\text{VQA}}\downarrow$ | $\mathcal{V}_{\text{LPIPS}}\downarrow$ | CV-DSim↓ |
| Global | K. Kontext | 0.097 | 0.414 | 0.076 | 0.020 | 62.86 | 0.042 | 0.061 | 0.985 |
| | Diff. Steering | 0.124 | 0.660 | 0.025 | 0.182 | 61.91 | 0.128 | 0.348 | 0.986 |
| | FlowSlider | 0.095 | 0.242 | 0.143 | 0.047 | 64.44 | 0.045 | 0.167 | 0.673 |
| | VeloEdit | 0.106 | 0.470 | 0.053 | 0.021 | 54.19 | 0.070 | 0.022 | 0.782 |
| | SliderEdit | 0.106 | 0.412 | 0.037 | 0.081 | 62.78 | 0.068 | 0.132 | 1.000 |
| | SliderEdit$^K$ | **0.141** | 0.898 | 0.024 | 0.036 | 63.66 | 0.044 | 0.109 | 1.243 |
| | PrecisionFlux | 0.111 | 0.748 | **0.018** | 0.094 | **65.84** | **0.022** | 0.434 | 1.582 |
| | UniSlider | **0.141** | **0.905** | 0.023 | **0.016** | 65.71 | 0.038 | **0.008** | **0.643** |
| Attribute | K. Kontext | 0.057 | 0.414 | 0.154 | 0.021 | 65.06 | 0.053 | 0.173 | 1.247 |
| | Diff. Steering | **0.089** | 0.685 | 0.128 | 0.097 | 65.40 | 0.078 | 0.391 | 1.246 |
| | FlowSlider | 0.069 | 0.388 | 0.433 | 0.044 | 66.28 | 0.077 | 0.190 | **0.625** |
| | VeloEdit | 0.056 | 0.450 | 0.134 | 0.023 | 55.87 | 0.078 | 0.117 | 1.151 |
| | SliderEdit | 0.066 | 0.412 | **0.072** | 0.028 | 63.82 | **0.039** | 0.313 | 1.221 |
| | SliderEdit$^K$ | 0.083 | 0.718 | 0.118 | 0.032 | 68.67 | 0.051 | 0.151 | 1.158 |
| | PrecisionFlux | 0.074 | 0.634 | 0.164 | 0.096 | 68.27 | 0.047 | 0.513 | 1.162 |
| | UniSlider | 0.083 | **0.720** | 0.117 | **0.017** | **69.39** | 0.047 | **0.016** | 0.664 |
| Structural | K. Kontext | 0.080 | 0.461 | 0.253 | 0.022 | 65.09 | 0.041 | 0.114 | 1.228 |
| | Diff. Steering | 0.109 | **0.786** | 0.348 | 0.174 | 66.11 | 0.077 | 0.432 | 0.904 |
| | FlowSlider | 0.106 | 0.337 | 0.558 | 0.050 | 65.54 | 0.080 | 0.137 | 0.726 |
| | VeloEdit | 0.075 | 0.490 | 0.265 | 0.024 | 55.74 | 0.069 | 0.094 | 1.056 |
| | SliderEdit | 0.090 | 0.495 | **0.200** | 0.036 | 63.97 | 0.033 | 0.308 | 1.375 |
| | SliderEdit$^K$ | **0.112** | 0.765 | 0.231 | 0.038 | 68.92 | 0.034 | 0.141 | 1.065 |
| | PrecisionFlux | 0.095 | 0.626 | 0.227 | 0.098 | 68.51 | **0.024** | 0.517 | 1.388 |
| | UniSlider | **0.112** | 0.771 | 0.236 | **0.019** | **69.46** | 0.052 | **0.024** | **0.648** |

Table 4: **Benchmark results by edit type.** The same metrics as Tab. 1, split over the three edit categories (100 examples each, 300 total). $^K$ marks a method running on the FLUX.2-klein-4B backbone, i.e. the same base as ours, separating backbone advantage from method.

| Method | Edit Fid. | IQA | Monotonicity | | Uniformity | Fidelity × Preservation |
|---|---|---|---|---|---|---|
| | QEval | CLIPIQA↑ | $\mathcal{V}_{\text{DreamSim}}\downarrow$ | $\mathcal{V}_{\text{Qeval}}\downarrow$ | CV-TPIPS↓ | $\text{VQA}_{\text{edit}} \times (1-\text{VQA}_{\text{id}})\uparrow$ |
| KontinuousKontext | 0.660 | 0.488 | 0.0007 | 0.167 | 0.438 | 0.357 |
| DiffusionSteering | 0.759 | 0.489 | 0.0008 | 0.196 | 0.468 | 0.592 |
| FlowSlider | 0.607 | 0.503 | 0.0004 | 0.175 | 0.420 | 0.196 |
| VeloEdit | 0.682 | 0.402 | 0.0005 | 0.168 | 0.427 | 0.407 |
| SliderEdit | 0.671 | 0.469 | 0.0009 | 0.147 | 0.433 | 0.386 |
| SliderEdit$^K$ | **0.763** | 0.553 | 0.0011 | 0.126 | 0.393 | 0.702 |
| FireFly Tune | 0.739 | 0.524 | 0.0011 | 0.125 | 0.392 | 0.584 |
| UniSlider | **0.763** | **0.572** | **0.0002** | **0.112** | **0.372** | **0.706** |

Table 5: **Additional metrics.** Averaged over the 300 benchmark examples. Averaged over all benchmark examples.

We scale the pairwise outcomes with Thurstone's Case V model, with confidence intervals from a bootstrap over participants. Mosteller's $\chi^2$ does not reject the model on either question. UniSlider leads both scales by a wide margin: $+0.88\,[+0.76, +1.07]$ on uniformity against $+0.23$ for the next method, and $+0.82\,[+0.69, +1.01]$ overall. The two scales correlate at $r = 0.92$, so the method judged most evenly spread is also the one judged better overall. Figure 10 shows a screenshot of the user study (right) and the plotted Thurston Case V (left).

**Further UniSlider examples.** Figures 13 to 16 present additional examples of UniSlider for global/stylization, attribute, facial-expression, and structural edits, respectively. Across all edit categories, UniSlider consistently produces high-quality results.

**Additional masking examples.** Figure 17 presents examples of UniSlider with spatial masking. For local edits, such as "Make it low poly 3D" or "Make it moldy", the mask selects the object to be edited. For global edits, such as "Make it full of ivy" or "Make wood look old and weathered", it

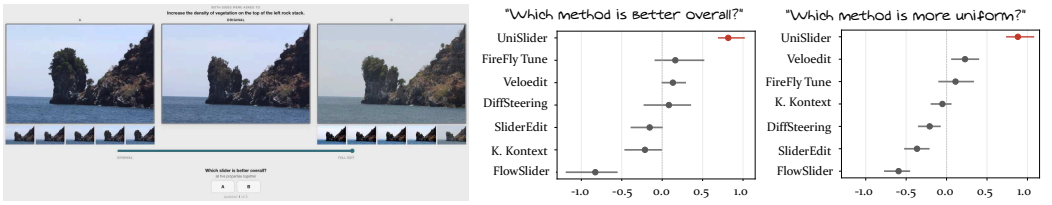


Figure 10: **Benchmark 2AFC user study. Left:** a screenshot of the study interface, in which annotators drag a slider back and forth to compare methods A and B at matching strengths. **Right:** Thurstone Case V values with confidence intervals from the pairwise judgments, for the two questions asked: which method is better overall, and which is more uniform. A total of 25 observers participated in the study making a total of 881 examples.




Figure 11: **Perceptual metric 2AFC user study. Left:** two examples of the rendered images at different strengths. **Right:** PLCC and SRCC for all evaluated metrics. A total of 15 observers participated in the study making a total of 538 annotations.

restricts the edit to a semantically coherent region of the image. In particular, the two bottom rows show that, since the mask takes values in $[0, 1]$, it can create smooth transitions between edited and preserved regions, as well as arbitrary shapes, as in the bear's fur.

**Further 2D examples.** Figures 18 and 19 present further 2D results, in which two edits are composed into a grid. As in Fig. 6, the first instruction spans the horizontal axis, and each of its outputs is then used as the input to UniSlider with the second instruction, which spans the vertical axis. Since UniSlider is trained on a single instruction at a time and never sees composed edits, this composition requires no additional training. The examples cover different type of edits. In all of them, the grids remain approximately uniform along both axes, and the two edits progress largely independently. Because the composition is sequential, the two edits are not interchangeable in general, as the second one is applied to the output of the first.

## G Further Analysis

**Perceptual metrics correlation with human judgment.** Learned perceptual metrics such as DreamSim (Fu et al., 2023) and TPIPS (Wang et al., 2026) are trained on relative human judgments. Annotators are shown two candidates and a reference and asked which candidate is closer, never by how much. Their values are therefore calibrated for ranking, and nothing in their training guarantees that a distance twice as large corresponds to a change that looks twice as big. Perceptual uniformity requires precisely that, so we validate it directly with a user study.

Every uniformity measure in this paper rests on a perceptual distance $D$, so we check that $D$ agrees with people when measuring strength before using it to score methods. We built 15 scenes, each with a concept (e.g. *"make it winter"*) rendered at four prompt-specified magnitudes from subtle to very strong, and collected 538 two-alternative forced-choice judgements. We show the original image

| | Edit Fidelity | | Identity | Endpoint | IQA | Monotonicity | | Uniformity | Efficiency | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $\text{CLIP}_{\text{dir}}\uparrow$ | $\text{VQA}_{\text{edit}}\downarrow$ | $\text{VQA}_{\text{id}}\downarrow$ | RMSE$\downarrow$ | MUSIQ$\uparrow$ | $\mathcal{V}_{\text{VQA}}\downarrow$ | $\mathcal{V}_{\text{LPIPS}}\downarrow$ | CV-DSim$\downarrow$ | Time(s) | Param |
| Qwen-Edit-Dist | 0.102 | 0.693 | 0.158 | 0.059 | 67.21 | 0.055 | 0.031 | 0.641 | 32.4 | 29B |
| FireRed-1.0-Dist | 0.109 | 0.802 | 0.114 | 0.183 | 67.93 | 0.089 | 0.052 | 1.038 | 68.7 | 29B |
| Flux-Klein-4B | 0.112 | 0.799 | 0.125 | 0.018 | 67.09 | 0.046 | 0.016 | 0.652 | 7.2 | 8B |
| Flux-Klein-9B | 0.114 | 0.820 | 0.141 | 0.018 | 68.19 | 0.051 | 0.018 | 0.748 | 14.8 | 17B |
| $r = 4$ | 0.112 | 0.799 | 0.125 | 0.020 | 66.94 | 0.052 | 0.018 | 0.661 | 7.2 | 1M |
| $r = 8$ | 0.112 | 0.799 | 0.125 | 0.018 | 67.02 | 0.048 | 0.018 | 0.660 | 7.2 | 2M |
| $r = 16$ | 0.112 | 0.799 | 0.125 | 0.018 | 66.98 | 0.048 | 0.018 | 0.654 | 7.2 | 4M |
| $r = 32$ | 0.112 | 0.799 | 0.125 | 0.018 | 67.09 | 0.046 | 0.016 | 0.652 | 7.2 | 78M |

Table 6: **Editing backbone and LoRA rank ablations. Top:** UniSlider trained on four editing backbones. **Bottom:** adapter rank swept on Flux-Klein-4B, our default setting ($r$=32, shown in both blocks). Time is the wall-clock cost of generating an 11-frame slider on an RTX Pro 6000; Params counts the full model in the top block and the trainable adapter in the bottom one.

and two edits and ask: "which edit changed the image more?". For each observer, nine pairs were repeated to measure self-consistency, and discard the observers that voted differently in duplicated pairs. Figure 11 (right) shows two of the 15 examples.

We recover a Bradley–Terry scale per scene from these judgments and correlate it with each metric's distance to the input, reporting both Pearson (PLCC) and Spearman (SRCC) correlations in Table 2. SRCC measures whether a metric orders edits as humans do, while PLCC measures whether its values scale linearly with perceived change, which is what perceptual uniformity requires. DreamSim and TPIPS agree best with human judgments and are effectively tied, followed by LPIPS and CLIP . Pixel-space RMSE is clearly behind and a color-histogram distance further still. For every metric PLCC and SRCC are within $0.01$ of each other, so the metrics that rank edits correctly also scale close to linearly with perceived magnitude, which is what justifies using $D$ as a measure of strength rather than only as a ranking signal. See Figure 11 (left) for a plot of the results.

## H Ablation studies

**Different baseline editors.** The top block of Table 6 reports UniSlider trained on different base editing models: a distilled Qwen-Image-Edit, an 8-step distilled FireRed-1.0-Edit, and Flux.2-Klein at both 4B and 9B. All four reach comparable edit fidelity and identity preservation, which is expected since these are largely inherited from the backbone at full strength. The differences appear in the slider-specific properties. Both Flux-Klein models are clearly ahead on monotonicity and uniformity, while FireRed-1.0 is the weakest and also fails to return the input at zero strength. We attribute this to its eight-step schedule. The objective is imposed on decoded pixels, so the gradient is backpropagated through twice as many sampling steps as for the four-step models, which makes the optimization considerably harder.

We adopt Flux-Klein-4B for UniSlider. Flux-Klein-9B is slightly better on edit fidelity and image quality, but it is worse on uniformity, the property we optimize for, while being twice as large and twice as slow. At 8B parameters and 7.2 s for 11 frames, Flux-Klein-4B is also the most efficient of the four, generating a full slider four times faster than Qwen-Image-Edit and nine times faster than FireRed-1.0.

**LoRA rank.** The bottom block of Table 6 sweeps the adapter rank $r \in \{4, 8, 16, 32\}$ on Flux-Klein-4B. Edit fidelity and identity preservation are unchanged across the sweep, and uniformity improves only marginally with rank, with no measurable cost in inference time. This is consistent with the capacity analysis of Section 4. Rank governs how many image–instruction pairs the adapter can hold a uniform trajectory for, but on the full dataset the residual non-uniformity is recovered by adaptive sampling rather than by additional capacity. We use $r = 32$ throughout, as it is best on every metric at a cost of 7.9M trainable parameters, under $0.1\%$ of the backbone.

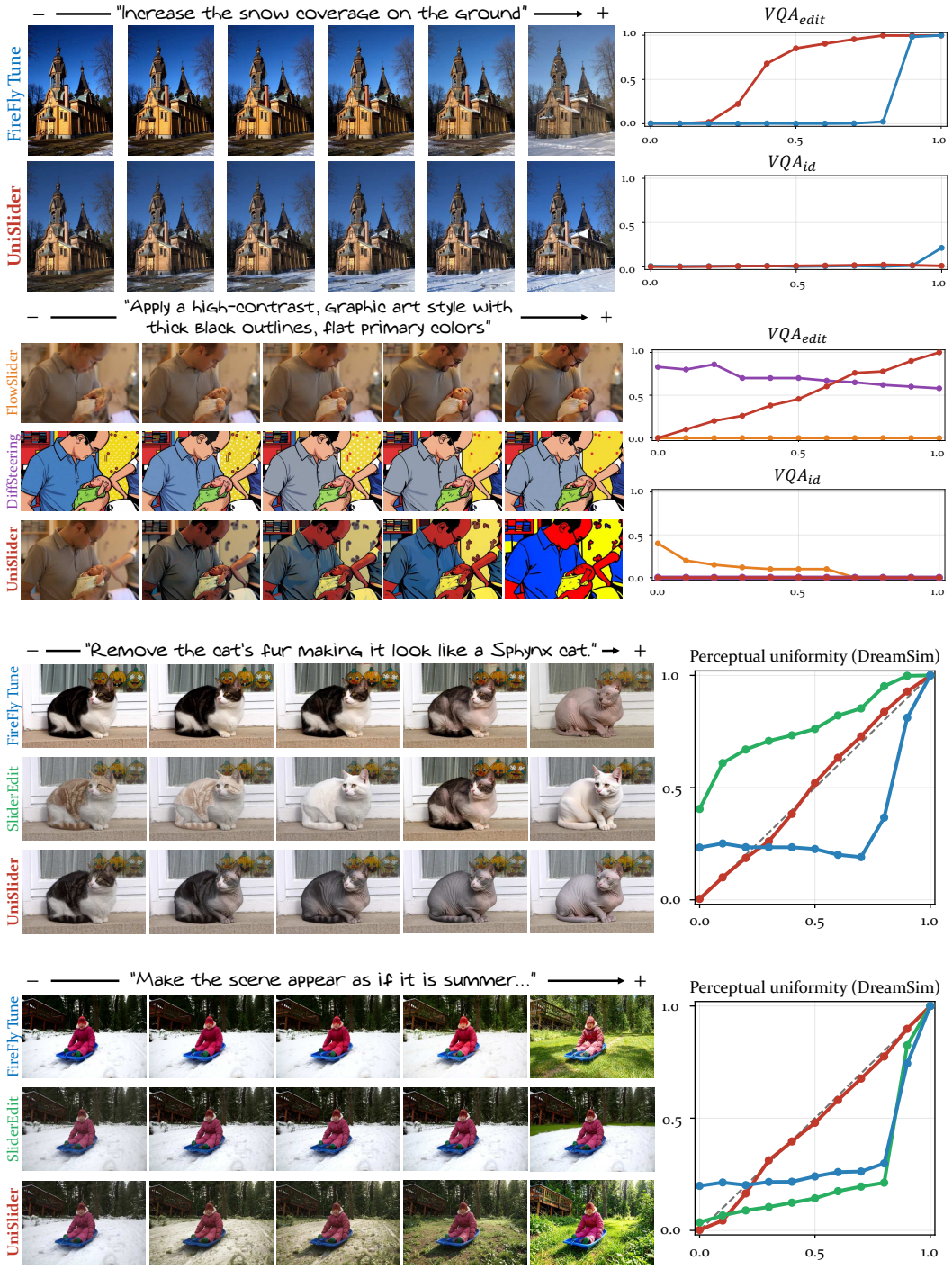


Figure 12: **Benchmark comparisons with per-frame metrics.** The first two examples compare UniSlider against Firefly Tune (top) and against FlowSlider and DiffusionSteering (second), annotating each frame with its $\text{VQA}_{\text{edit}}$ and $\text{VQA}_{\text{id}}$ values. The last two compare UniSlider against Firefly Tune and SliderEdit, plotting instead the perceptual distance from the input at each frame, so that a flat segment marks a dead zone and a steep one a jump. Note how UniSlider achieves better uniformity across all different types of edits.

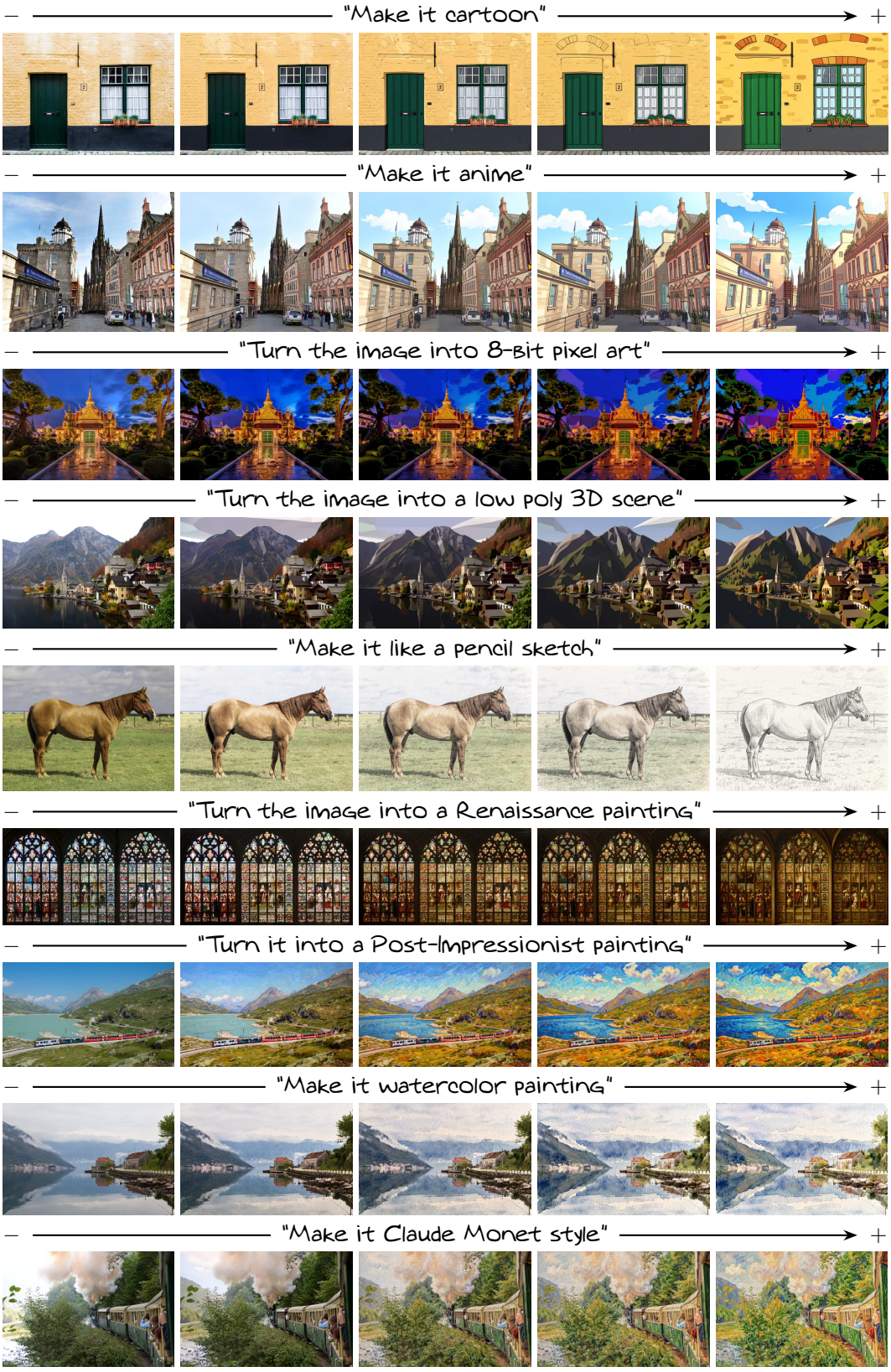


Figure 13: **Additional UniSlider results with changes in visual style.** From left to right input image, all the intermediate strength levels and the default output of Flux-Klein-4B. All the input images are from the DIV2K (Agustsson & Timofte, 2017) dataset.

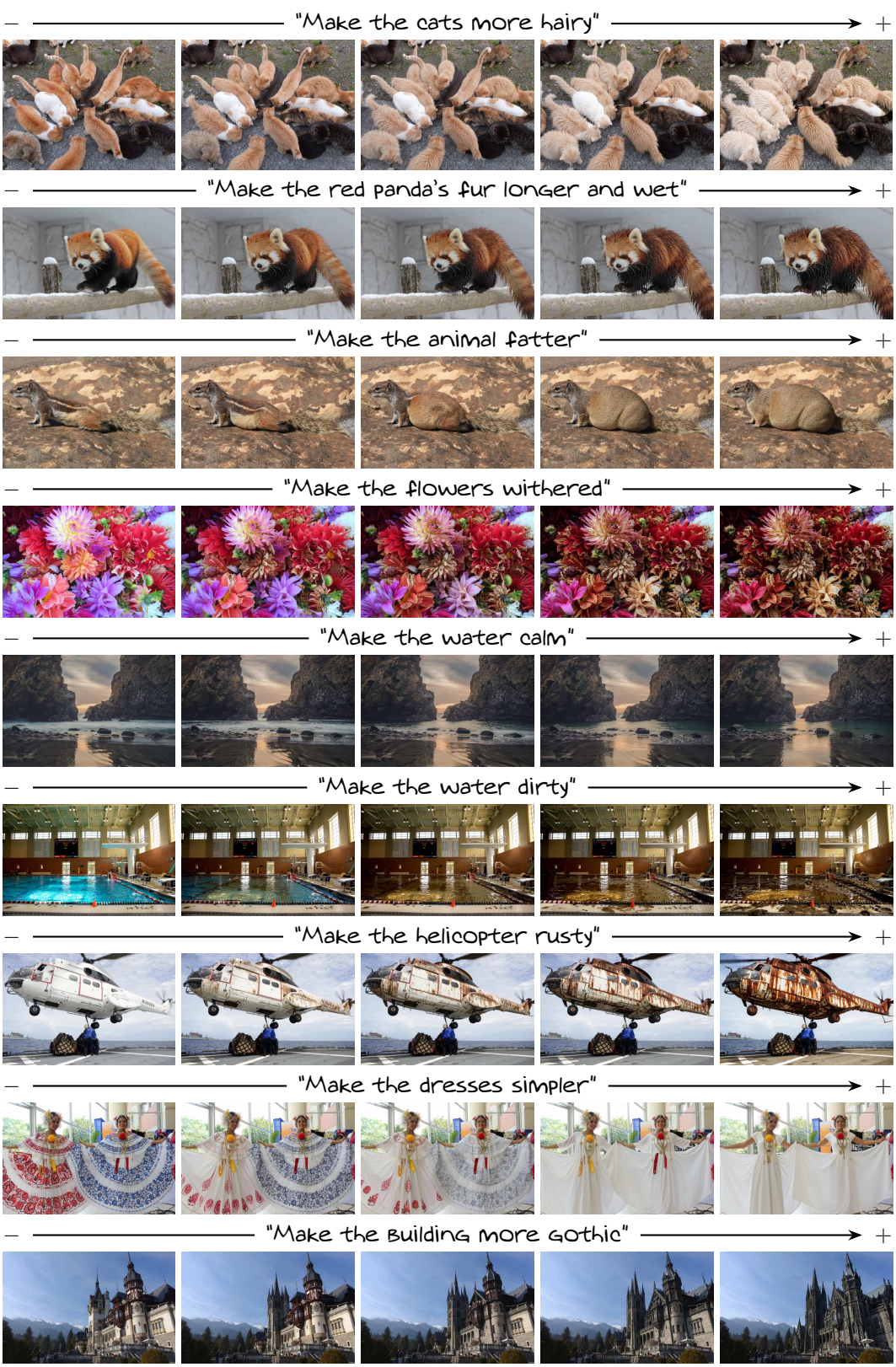


Figure 14: **Additional UniSlider results with attribute edits.** From left to right input image, all the intermediate strength levels and the default output of Flux-Klein-4B. All the input images are from the DIV2K (Agustsson & Timofte, 2017) dataset.

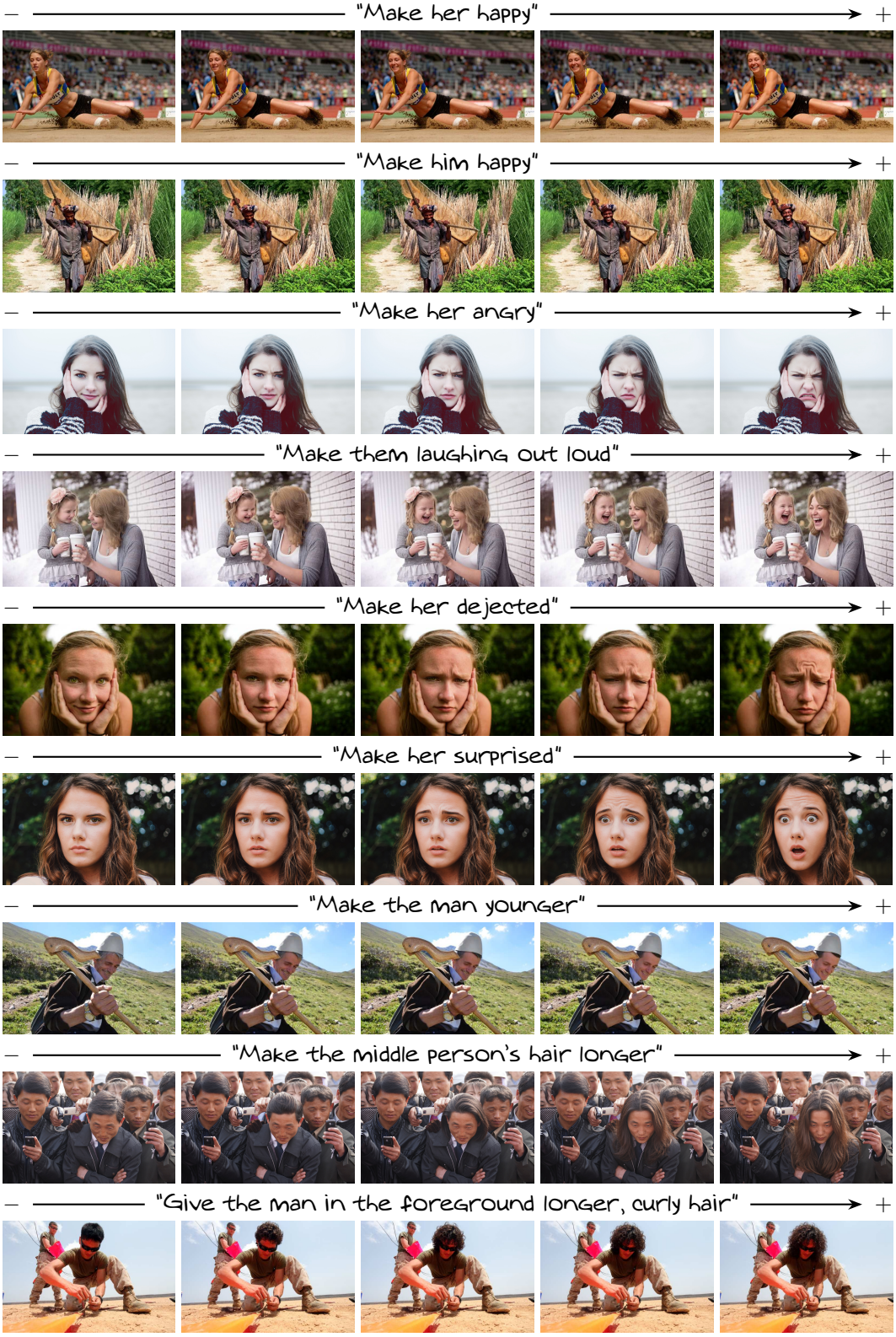


Figure 15: **Additional UniSlider results with structural edits.** From left to right input image, all the intermediate strength levels and the default output of Flux-Klein-4B. All the input images are from the DIV2K (Agustsson & Timofte, 2017) dataset.

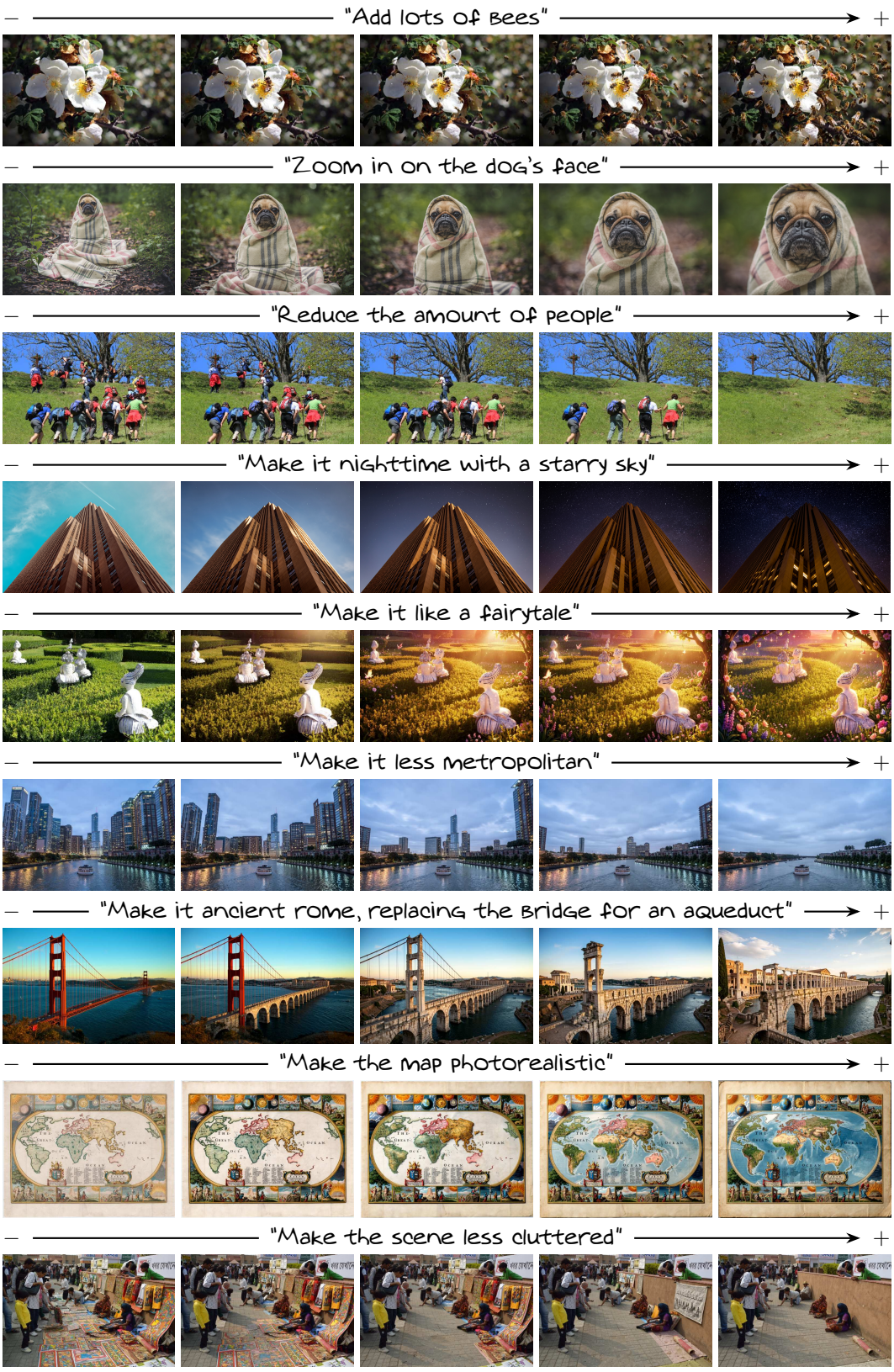


Figure 16: **Additional UniSlider results with structural edits.** From left to right input image, all the intermediate strength levels and the default output of Flux-Klein-4B. All the input images are from the DIV2K (Agustsson & Timofte, 2017) dataset.

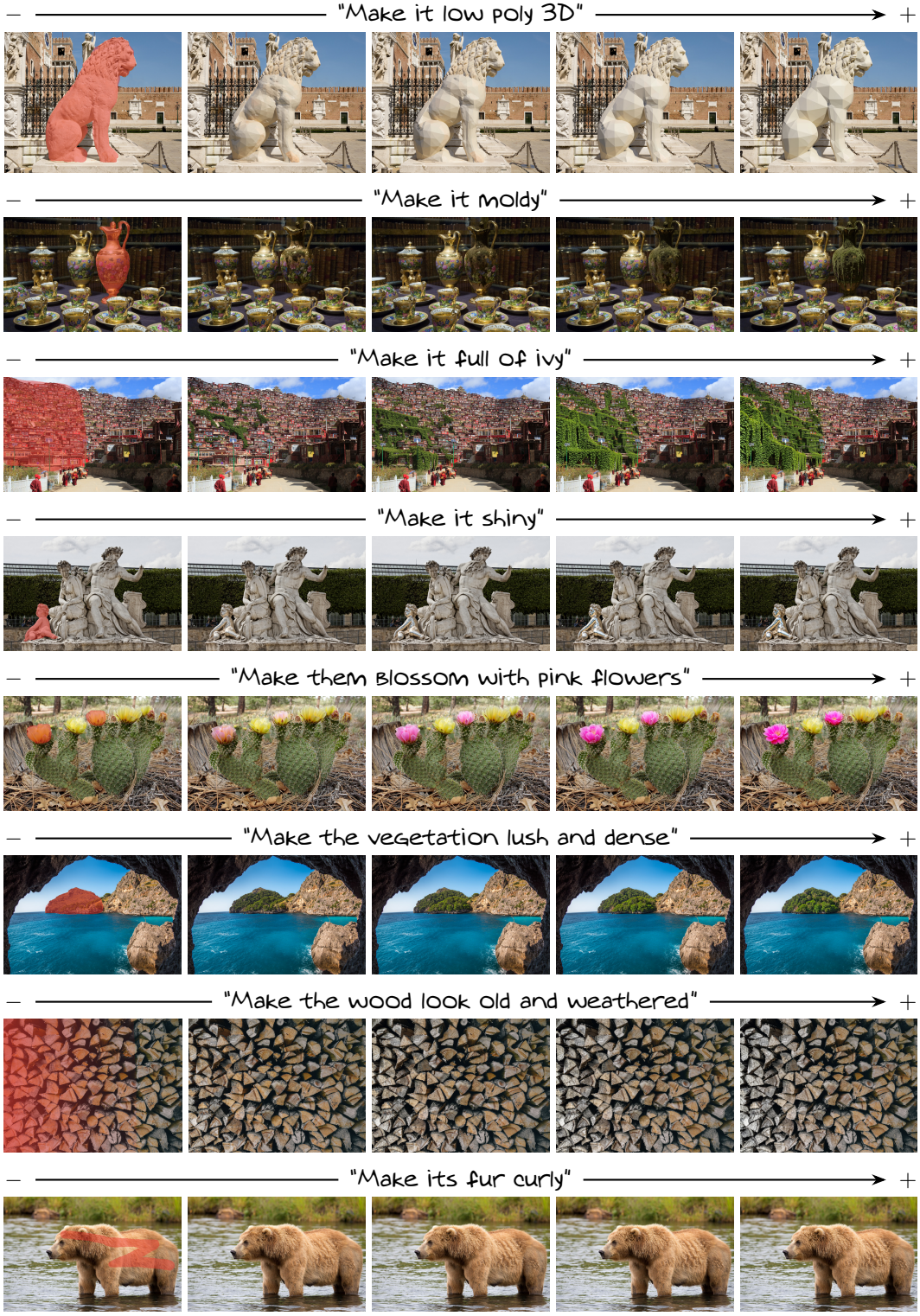


Figure 17: **Additional UniSlider with masking.** From left to right input image, all the intermediate strength levels and the default output of Flux-Klein-4B. All the input images are from the DIV2K (Agustsson & Timofte, 2017) dataset.




Figure 18: **Additional 2D UniSlider results.** From left to right input image, all the intermediate strength levels and the default output of Flux-Klein-4B. All the input images are from the DIV2K (Agustsson & Timofte, 2017) dataset.




Figure 19: **Additional 2D UniSlider results.** From left to right input image, all the intermediate strength levels and the default output of Flux-Klein-4B. All the input images are from the DIV2K (Agustsson & Timofte, 2017) dataset.