The Unreasonable Effectiveness of Text Embedding Interpolation for Continuous Image Steering

Yigit Ekin*
Reve

Yossi Gandelsman
Reve

Abstract

Continuous image editing requires fine-grained control over semantic attributes in text-conditioned generative models. Recent methods obtain such control through trainable adapters, test-time optimization, or architecture-specific design choices. In this work, we revisit steering the text-encoder space as a simpler alternative for open-vocabulary slider construction. We show that, as text-conditioned generative models become increasingly capable, linear directions in their frozen text-encoder spaces can provide a competitive interface for continuous control when carefully constructed. To make this interface practical, we introduce an automated pipeline that constructs continuous sliders from just a prompt and a concept, eliminating manual data curation and coefficient tuning. First, a language model generates balanced contrastive prompt pairs. Then, difference-of-means over the frozen text encoder estimates a concept direction, and an LLM-assisted process selects the prompt tokens to steer. Finally, an elastic range search adaptively calibrates the slider coefficient interval. As the process includes only changing the text encoder hidden states, the same pipeline can be applied across different text-conditioned generative backbones without denoiser-specific modifications. Across diverse set of edit types, our method outperforms prior training-free baselines and approaches the edit compliance of trained controllers while avoiding per-concept and per-backbone training. Our results suggest that a carefully designed text-space steering is a surprisingly strong practical baseline for open-vocabulary continuous editing. Especially in a rapidly evolving generative model ecosystem, where repeatedly adapting specialized controllers to new backbones can be a substantial burden. Code is available at https://github.com/YigitEkin/diffusion-sliders.

1 Introduction

Text-conditioned generative models have become a standard interface for visual creation and editing [33, 36, 35, 26, 17, 19, 44, 13, 46, 42]. Natural-language prompts can specify rich target states, but they do not expose a direct control variable for edit strength. For example, a user may want to gradually increase a smile, make an image more photorealistic, or control how crowded a scene becomes. Continuous editing sliders address this limitation by turning a semantic edit into a controllable coefficient.

Recent methods achieve continuous control through trained sliders, instruction-level controllers, test-time optimization, guidance-level mechanisms, or architecture-specific interventions [10, 7, 47, 23, 16, 3, 15, 43, 8]. While effective, these approaches typically rely on additional mechanisms that must be trained or re-validated each time for a specific network or an edit concept. In an ecosystem where new generative backbones and editors appear rapidly and repeatedly, adapting these networks to such slider mechanisms can become a practical burden. Motivated by this, we ask: *Can automated slider construction be achieved without sacrificing edit strength, slider continuity, or setup efficiency?*

*Work done while an intern at Reve.

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

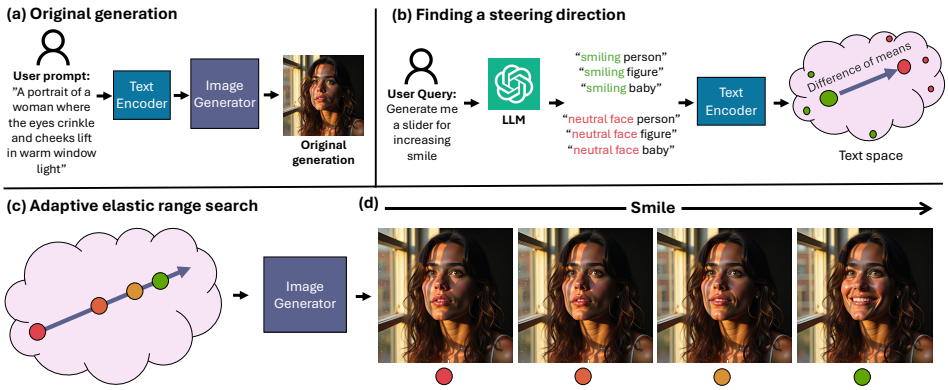

Figure 1: **Our framework.** Given a user-defined concept and an input prompt, our method constructs a continuous editing slider without additional training or manual slider coefficient tuning. It first generates balanced contrastive prompt pairs and estimates a concept direction in the frozen text-encoder space. It then selects the input-prompt tokens to steer and calibrates a prompt-specific coefficient range with elastic range search. Scaling the resulting text-encoder direction produces a continuous edit trajectory.

We answer this question by revisiting the simplest available control space: the frozen text-conditioning representation used by the generator. As text-conditioned generators become increasingly capable, linear directions in their frozen text-encoder spaces can serve as effective continuous controls [24]. Rather than training a controller, optimizing image-specific embeddings, or modifying denoiser internals, we construct sliders directly over the generator's existing text-conditioning interface.

While this space is capable of producing successful continuous edits, the idea alone does not yield a usable slider. The end user must still manually construct the concept direction, isolate the correct prompt tokens to steer, and blindly guess the steering magnitude. These human-in-the-loop choices directly affect edit quality and make the method difficult to use across prompts, concepts, and new backbones. Our goal is to automate these choices through a three-step construction protocol: First, a language model generates balanced contrastive prompt pairs, which are encoded by the frozen text encoder to estimate a debiased difference-of-means concept direction. Second, an LLM-assisted rule selects the input-prompt tokens to steer, reducing unintended changes to unrelated context. Third, an *elastic range search* calibrates the steering coefficients for the given prompt and concept by redistributing candidate slider points according to perceptual gaps between adjacent generations, proposing a slider coefficient range to overcome under and over-steering.

This pipeline allows us to evaluate not only the efficacy of text-space steering, but also the practical usability of automated construction. Our calibrated text-space steering consistently outperforms existing training-free baselines across local, global, and stylization edits. Furthermore, when applied to highly capable generative backbones, it closely matches the edit quality of dedicated controllers while entirely bypassing the need for per-concept or per-model training. We further analyze token selection, prompt-pool size, LLM dependence, runtime, and cross-backbone behavior, showing how much of the slider-construction process can be shifted from manual or trained setup to an automated procedure over the frozen text-conditioning interface.

Our contributions can be summarized as follows: **(1) Automated construction pipeline:** We introduce a fully automated, three-step pipeline that generates contrastive data, isolates target tokens, and calibrates coefficient ranges to create usable sliders directly from a user prompt and concept. **(2) Text-conditioning-only continuous control:** We demonstrate that carefully constructed linear directions in frozen text-encoder spaces provide a competitive interface for local, global, and stylization edits, with evidence that this approach becomes stronger on more capable editing backbones. **(3) Comprehensive setup analysis:** We ablate our core components of token selection, dataset size, LLM sensitivity, and analyze runtime cost to clarify exactly when automated text-space steering is a practical alternative to trained mechanisms.

2 Related Work

We provide an extended related-work discussion in Appendix A; here we focus on the distinctions most relevant to automated slider construction and training-free control.

Continuous slider-based editing. Text-guided editing models such as InstructPix2Pix [5], LEDITS++ [4], Qwen-Image [44], Flux-Kontext [19], and Flux2 [18] enable strong instruction-based editing, but lack calibrated continuous control over edit strength. Recent slider-like methods address this in different ways: Concept Sliders / Flux-Slider [10] train concept-specific LoRA adapters and use the adapter scale as the coefficient; PromptSlider and TextSlider [7, 38] learn low-rank adaptations in the text encoder for per-concept control; SliderEdit [47] trains a low-rank adapter for unified instruction-level control in text encoder space; Kontinuous Kontext [23] trains a strength-conditioned projection on synthetic edit data; and AdaOr [43] controls intensity by shifting the classifier-free guidance origin. While effective, these methods couple slider construction to a specific concept, target model, text encoder, or editing pipeline. In contrast, our method constructs a calibrated slider from a user concept and prompt by intervening only the text-conditioning representation space, without training a module, modifying generator weights, or fixing the coefficient range by hand.

Text, latent, and representation-space steering. A related line of work manipulates learned representations rather than training a full editor. StyleCLIP [25] and CLIP-PAE [49] identify semantic directions in CLIP space for continuous manipulation, but are tied to Generative Adverserial networks [11] rather than modern diffusion or flow-matching backbones. Imagic [16] optimizes text embeddings for editing, but requires per-image test-time optimization and generator fine-tuning. Baumann et al. [3] identify token-level semantic directions for attribute control, refined at test time with gradient correction over diffusion noise predictions. SAEdit [15] discovers token-level steering directions via sparse autoencoders, adding an auxiliary representation-learning stage, and FluxSpace [8] offers training-free control restricted to architecture-specific internals of Flux-style models. Closest to ours is text-space steering, but rather than optimizing embeddings, learning auxiliary representations, or modifying denoiser internals, we use the frozen text-conditioning space as a common interface and automate every step from a user concept to a calibrated slider.

Automated slider construction. Most continuous-control methods leave at least one stage of slider construction manual, trained, or model-specific: data design, token or representation selection, controller training, or per-prompt coefficient calibration. Our method targets the full process: an LLM generates balanced contrastive pairs to estimate the concept direction, an LLM-assisted rule selects the input-prompt tokens to steer, and elastic range search adaptively calibrates the coefficient bounds per prompt and concept. Thus, our distinction is not only that the method is training-free, but that direction estimation, token selection, and range calibration are unified into a single automated slider-construction pipeline.

3 Automated training-free continuous steering

We present an automated training-free method for continuous steering of text-conditioned visual generation. Given a user-defined concept (e.g., *happy*↔*sad*, *photorealistic*↔*cartoon*), we use a Large Language Model to produce a small, debiased set of contrastive prompt pairs and compute a steering vector via token-level pooling and difference-of-means (Section 3.1). Relevant tokens are automatically selected using an LLM-assisted procedure (Section 3.2), and an elastic range search determines effective steering coefficients for smooth, continuous control (Section 3.3). Together, these components yield a simple framework for controllable generation without additional model training or manual slider tuning.

3.1 Finding a steering direction

We start by presenting how language conditioning is used in flow matching models with a text encoder. Then, we explain how to extract steering direction in the representation space of the text encoder.

Text-conditioned flow matching. We consider a text-conditioned flow-matching generative model that evolves a sample $\mathbf{x}_t$ according to a velocity field $\mathbf{v}_\theta$, conditioned on text prompt $\mathbf{p}$ encoded via

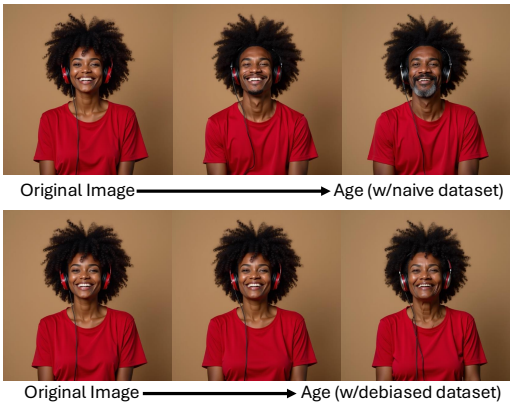

Figure 2: **Illustration of bias inheritance in steering.** When the age direction is computed from a biased dataset (e.g., predominantly old men), the resulting steering vector entangles gender with age. Consequently, age manipulations not only modify apparent age but also introduce unintended gender-related changes, revealing the dataset's underlying bias.

text encoder $\mathbf{E}$:

$$\dot{\mathbf{x}}_t = \mathbf{v}_\theta(\mathbf{x}_t, t, \mathbf{E}(\mathbf{p})), \tag{1}$$

$$\mathbf{E}(\mathbf{p})' = \mathbf{E}(\mathbf{p}) + \alpha\, \mathbf{d}_s, \tag{2}$$

where $\dot{\mathbf{x}}_t$ denotes the instantaneous change of $\mathbf{x}_t$ at time t, $\mathbf{d}_s$ is the learned steering vector for the target concept (defined in Eq. (3)), and α controls its steering strength. To modulate a target semantic concept (e.g., *smiling*, *realism*), we substitute $\mathbf{E}(\mathbf{p})'$ into Eq. 1, perturbing the generative trajectory along $\mathbf{d}_s$.

LLM-generated contrastive pairs. To compute steering directions, we construct a set of K contrastive prompt pairs $\{(p_i^+, p_i^-)\}_{i=1}^K$, each differing only in the target concept while preserving all other semantic content. For example, for the concept *sad↔happy*, the corresponding pair may be (*"A portrait of a sad person"*, *"A portrait of a happy person"*). These contrastive pairs are automatically generated by a large language model, which ensures that changes are concept-isolated and contextually consistent, eliminating the need for manual dataset design or bias-prone selection. The full query prompt for dataset generation can be seen in appendix Section I.1.

Debiasing via style-token pooling. As we rely on an LLM-produced dataset to compute the steering direction $\mathbf{d}_s$, a biased dataset can lead to entangled attribute directions. For example, when the contrastive prompts used for constructing the age direction also correlate age with gender (e.g., *"A shot of a young woman"* and *"A shot of an old man"*), the resulting steering vector jointly alters both attributes when manipulating the age direction (top row in Fig. 2). Such entanglement is undesirable, as the goal is to obtain a steering vector that controls only the intended attribute. To mitigate this, we prompt the LLM to generate balanced contrastive pairs and *only utilize concept-relevant tokens*. E.g., averaging the hidden features of the *young* and *old* token spans in the pair (*"A shot of a young person"* / *"A shot of an old person"*) leads to a more disentangled edit (bottom row in Fig. 2).

More precisely, let S_i^+ and S_i^- denote the style-token spans for each pair i. The pooled features $\mathbf{E}(\mathbf{p})_{i\pm} = \frac{1}{|S_i^\pm|}\sum_{j\in S_i^\pm}\mathbf{E}(\mathbf{p_i})[\mathbf{j}]$ isolate the desired attribute from unrelated tokens, yielding cleaner and more concept-specific directions.

Difference-of-means steering vectors. After obtaining the pooled features for each contrastive pair, we compute the difference-of-means vector using the final output representation of the text encoder as

$$\mathbf{s} = \frac{1}{K}\sum_{i=1}^{K}\mathbf{E}(\mathbf{p})_{i,+} - \frac{1}{K}\sum_{i=1}^{K}\mathbf{E}(\mathbf{p})_{i,-} \tag{3}$$

which captures the semantic displacement between the positive and negative poles of the target concept. To ensure consistent magnitude across different concepts and prevent scaling bias during inference, the resulting vector is ℓ_2-normalized: $\mathbf{d}_s = \frac{\mathbf{s}}{||\mathbf{s}||_2}$.

The normalized direction $\mathbf{d}_s$ defines a single, global steering axis applied exclusively to the encoder's final output embedding, derived directly from the encoder's final latent space without any additional training.

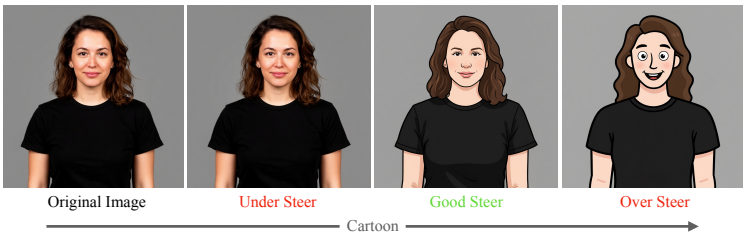

Figure 3: **Effect of steering magnitude on edit strength.** Weak steering produces minimal visual change, while large magnitudes lead to semantic drift. Our elastic range search algorithm automatically identifies the steering range that achieves perceptually consistent edits. In this case, a balanced cartoon stylization.

3.2 Single-image steering

Given an image generated from an input text, our method modifies the text embeddings of LLM-selected tokens according to the learned steering vector.

LLM-assisted token selection. We utilize an LLM to choose which tokens in the free-form input prompt should be steered for a given concept. We find that accurate token selection for adding the intended steering vector (e.g., adding *smile* steering vector to the token corresponding to *man*) is crucial for producing semantically aligned edits. Modification of irrelevant tokens (e.g., stopwords) may introduce either undesired edits or no changes at all, depending on the selection. To only steer concept-relevant tokens while covering a broad spectrum of edits, we group edits as *local*, *global* or *stylization* and prompts are classified as *implicit* or *explicit*. *local* edits modify localized attributes of the main subject while preserving subject identity and the overall scene (e.g., *smile*, *age*); *global* edits induce global or geometric changes that affect many regions at once, such as environment or pose (e.g., *winter* $\leftrightarrow$ *summer*, *standing* $\leftrightarrow$ *sitting*); and *stylization* edits primarily change the rendering style (e.g., *cartoon*, *anime style*). *implicit* prompts are prompts that do not contain the target attribute that is intended to be edited (e.g., "a man" for *smile*), while *explicit* prompts directly include it (e.g., "a sad man" for *smile*).

Rule-based strategy guided by an LLM. We automate token selection by prompting an LLM with a simple set of rules: *Stylization edits:* implicit prompts $\rightarrow$ steer main subject nouns (e.g., "cat"); explicit prompts $\rightarrow$ steer style tokens (e.g., "cartoon", "photorealistic"). *Global edits:* implicit prompts $\rightarrow$ steer the main subject nouns (e.g., "dog" for *standing*); explicit prompts $\rightarrow$ steer only the global edit tokens (e.g., "standing", "winter", "close-up"). *Local edits:* implicit prompts $\rightarrow$ steer the main subject nouns (e.g., "man" for *smile*); explicit prompts $\rightarrow$ steer only the attribute tokens (e.g., "sad", "old"). This rule-based LLM-guided procedure removes the need for manual annotation and ensures that steering modifies only semantically meaningful tokens, aligning edit scope with the intended concept. The full query prompt for token selection and inference details for specific edit types can be found in appendix Sections J and M.

3.3 Continuous steering

We aim to create a continuum of image edits by automatically determining the steering strength range that produces smooth and perceptually consistent edits. Most existing methods select this range $\alpha \in [\alpha_{\min}, \alpha_{\max}]$ through manual inspection, which can lead to undesired results: weak steering produces almost no change, while large magnitudes cause artifacts (see Fig. 3). Therefore, we present an algorithm that adaptively finds the slider coefficient range for each concept, inspired by elastic band search [12].

Data-driven initialization. The initial steering interval is estimated directly from the contrastive dataset used to construct the steering vector. Given positive pairs p_i^+ and the normalized steering direction $\mathbf{d}_s$, we compute $\alpha_{\max} = \max_i \langle \mathbf{d}_s, \mathbf{E}(\mathbf{p_i^+}) \rangle$, anchoring the search to the actual semantic variation of the concept in the embedding space. If the text encoder supports further extrapolation without causing semantic drift, we apply a simple lookup: generate an image at $2\alpha_{\max}$ and expand if the perceptual distance to the unsteered image remains within a reasonable range, repeating until

the distance exceeds the threshold or the maximum number of steps is reached. In practice, over the text encoders used in modern generative models [31, 2, 18, 30], at most 3 extrapolation steps are needed, adding negligible overhead.

Elastic-band optimization. The range $[\alpha_{\min}, \alpha_{\max}]$ is split into N control points $\{\alpha_i\}_{i=1}^{N}$, each corresponding to a generated image, connected by virtual springs whose tension $f(\alpha_i, \alpha_{i+1})$ encodes the perceptual difference between the images generated at α_i and α_{i+1}, measured by DreamSim [9]. The algorithm alternates between two operations: **MOVE** — each interior point α_i is shifted toward the neighbor with the larger normalized distance $f(\alpha_i, \alpha_{i+1})$ or $f(\alpha_{i-1}, \alpha_i)$, with a cosine-scheduled step size scaled by the imbalance factor $|f(\alpha_{i-1}, \alpha_i) - f(\alpha_i, \alpha_{i+1})|$; and **EXPAND** — a midpoint is inserted whenever a perceptual gap exceeds a threshold, increasing density in regions of steep change. All shifts are clamped to avoid point order flips, and control points are rendered in parallel to reduce runtime. The search proceeds until no further movement or expansion occurs. The resulting endpoints $(\alpha_{\min}, \alpha_{\max})$ represent the smallest and largest steering magnitudes that produce perceptually smooth and semantically consistent edits, replacing manual tuning with a data-driven procedure that generalizes across edit categories. Detailed pseudocode for elastic range search is provided in Section E.

4 Experiments

We evaluate our method on *local* (*smile*, *age*, *frostiness*, *wet texture*, *rusty surface*), *global* (*winter*, *nighttime*, *rainy weather*, *sitting*, *crowded scene*, *zoom-out camera*), and *stylization* (*cartoon*, *ghibli style*, *anime*, *photorealistic render style*) edits.

For comparisons on a common text-to-image backbone, we evaluate against SAEdit [15], FluxSpace [8], and Flux-Sliders [10] on FLUX.DEV [17]. Since Kontinuous Kontext [23] and SliderEdit [47] are built on the stronger image-to-image FLUX-KONTEXT [19] pipeline, we also evaluate our method on QWEN-IMAGE-EDIT [44], a strong instruction-based editing backbone. These cross-backbone results test our hypothesis that text-space steering becomes more competitive on stronger editing backbones; because they conflate method and backbone effects, we additionally report a controlled comparison in which all methods share the same backbone, inputs, and seeds, including our method on FLUX-KONTEXT (Section P).For all experiments, we use a debiased contrastive pool of $K = 100$ pairs. We observe sufficient embedding quality at $K = 50$ but use $K = 100$ for the remainder of the work to be safe, and find diminishing improvements beyond that (0.98 cosine similarity compared to 2000 examples). A more detailed analysis of contrastive tuple count K is provided in Appendix G. Appendices M and K provide inference details and dataset composition.

Magnitude Increment Deviation (MID_{dist}) We use DreamSim [9] to measure content and identity preservation, and ΔVQA [20] to quantify perceptual edit success. To evaluate continuous slider behavior, we additionally report ΔVQA trajectories and our proposed Magnitude Increment Deviation (MID) metric over six slider points. Because low MID can also result from weak edits, we interpret MID jointly with ΔVQA rather than as a standalone quality measure. An ideal continuous image editing pipeline should distribute semantic change across the slider in proportion to perceptual change. To measure this, we generate N edited images at uniformly spaced strengths $\alpha_i = \frac{i}{N-1}\alpha_{\max}$, where $\alpha_0 = 0$ and $\alpha_{N-1} = \alpha_{\max}$. For adjacent slider steps, we compute semantic increments $\Delta v_i = \left|\Delta\text{VQA}(I_{\alpha_{i+1}}) - \Delta\text{VQA}(I_{\alpha_i})\right|$ and perceptual increments $\Delta d_i = \text{LPIPS}(I_{\alpha_{i+1}}, I_{\alpha_i})$ for $i \in \{0, \ldots, N-2\}$. We normalize them into distributions $p_i = \frac{\Delta v_i}{\sum_{j=0}^{N-2} \Delta v_j + \varepsilon}$ and $q_i = \frac{\Delta d_i}{\sum_{j=0}^{N-2} \Delta d_j + \varepsilon}$, and define $\text{MID}_{\text{dist}} = \frac{1}{2}\sum_{i=0}^{N-2} |p_i - q_i|$. Lower MID_{dist} indicates better continuity: semantic change is distributed more consistently with perceptual change across the slider. Because Δv_i takes absolute differences, MID measures the *evenness* of semantic change relative to perceptual change, not its *direction*: an increasing and a decreasing trajectory with the same increment magnitudes receive the same score. MID therefore does not establish semantic monotonicity, which we measure separately with ρ, the mean per-example Spearman correlation between slider position and ΔVQA, and %mono, the fraction of trajectories whose ΔVQA never drops by more than 0.01 between adjacent slider steps (Section P).

4.1 Quantitative Results

Tab. 1 compares training-based and training-free continuous editing methods using edit success (ΔVQA$\uparrow$) and slider continuity (MID$\downarrow$). DreamSim–ΔVQA trade-offs across edit strengths are shown in Fig. 4, and additional DreamSim and per-category results are provided in Sections N and O.

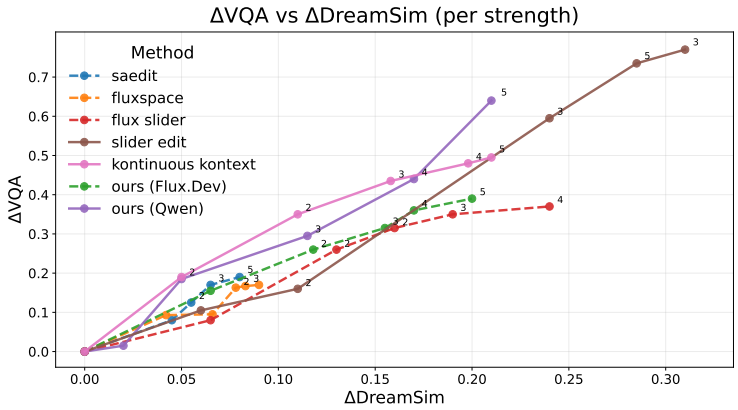

Figure 4: **Edit-strength/preservation trade-off.** ΔVQA versus DreamSim across steering strengths. Points closer to the upper-left achieve stronger edits with lower perceptual deviation. Dashed curves denote training-based methods; solid curves denote training-free methods.

Metric	Training-based				Training-free				
	SAEdit	Flux Slider	SliderEdit	Kontinuous Kontext	FluxSpace	Ours (First)	Ours (All)	Ours (Flux)	Ours (Qwen)
ΔVQA↑	0.18	0.36	**0.75**	0.48	0.15	0.15	0.44	0.38	*0.63*
MID↓	*0.36*	0.41	0.50	0.45	**0.33**	0.36	0.51	0.39	0.42

Table 1: **Quantitative comparison.** We report edit strength (ΔVQA↑) and slider continuity (MID↓). Best performing model is displayed in **bold** and second best is displayed in *italic*

Training-based methods built on the FLUX.KONTEXT backbone achieve the strongest edit compliance, with SliderEdit obtaining the highest ΔVQA. However, this comes with larger perceptual deviation and reduced slider smoothness, reflecting a trade-off between edit strength and gradual controllability. In contrast, training-free representation-manipulation methods such as FluxSpace and SAEdit obtain lower DreamSim and MID values, but this partly reflects under-editing: these methods often remain close to the original image and struggle with stronger global or stylization edits.

Our method occupies a favorable middle regime. On FLUX.DEV, it improves edit compliance over prior training-free baselines while maintaining competitive continuity. It also got 60% preference on our conducted user study between Flux-Based models for slider evaluation (See Section T). When applied to the stronger QWEN-IMAGE-EDIT backbone, the same text-space construction increases ΔVQA to 0.63, approaching trained controllers while avoiding per-concept or per-backbone controller training. Together with the trade-off curve in Fig. 4.

4.2 Qualitative Results

Fig. 5 compares Kontinuous Kontext, SliderEdit, FluxSpace, and our method across stylization, global scene editing, and local attribute editing. For photorealism, Kontinuous Kontext shows an unstable progression with visible artifacts, SliderEdit changes abruptly despite achieving the target effect, and FluxSpace under-edits. For crowdedness, Kontinuous Kontext and SliderEdit can produce the desired change but do not provide a smooth slider-like trajectory, while FluxSpace largely fails to realize the edit. For age editing, our method produces a gradual progression, while the baselines either saturate early or provide limited edit strength. Additional qualitative comparisons with more methods are provided in the supplementary material.

4.3 Runtime and Setup Cost

Our pipeline has two stages that are computed once per edit concept and cached, and two that are recomputed for every new prompt. We measure each stage averaged over 100 examples. Per concept, contrastive-pair generation takes 67.50 ± 9.77 s using GPT-4.1-mini [22] and token pooling takes 1.09 ± 0.36 s; both are saved and reused across all prompts that target the same concept. Per prompt, token selection takes 3.29 ± 2.71 s using Qwen3-8B (Thinking) [45] and elastic range search takes 25.75 ± 7.85 s, generating 13.92 ± 3.87 images per search with a 30-step generator. In total,

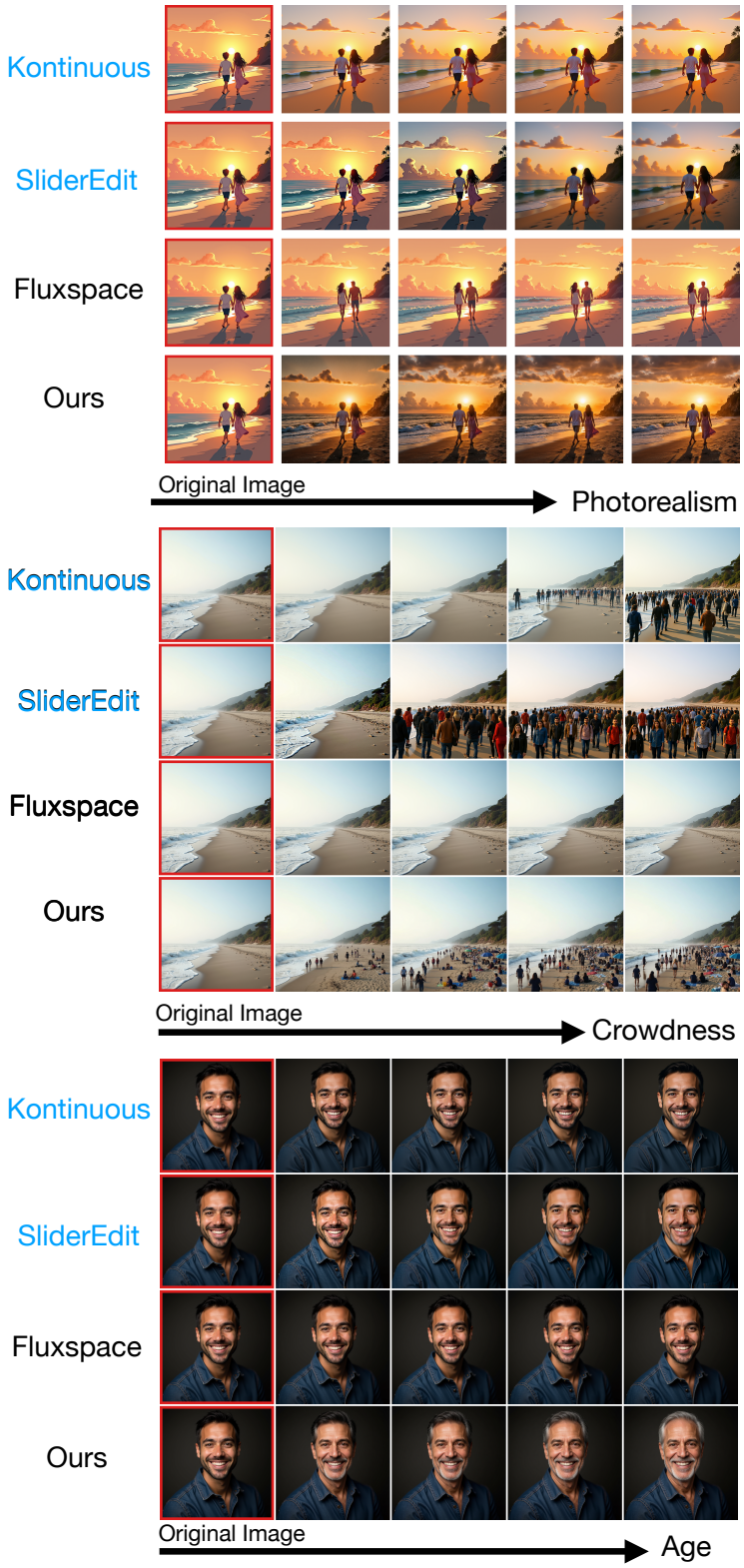

Figure 5: **Qualitative results.** We compare continuous edit trajectories for stylization (photorealism), global scene change (crowdedness), and local attribute editing (age). The leftmost images with red borders are the initial generations, and edit strength increases from left to right. Methods shown in blue are training-based.

Range selection	ΔVQA↑	MID↓	ρ ↑	%mono↑	Gen.↓
Fixed range	0.30	**0.43**	**0.71**	55.4	**1**
Data-driven initialization	0.33	0.49	0.39	62.0	**1**
Binary search	0.49	0.47	0.59	70.4	6
Elastic search (ours)	**0.51**	0.45	0.58	**72.1**	16

Table 2: **Range-selection ablation.** Controlled FLUX-KONTEXT evaluation (300 examples, Section P); only the range-selection procedure varies. Gen. is the average number of images generated per prompt to construct the range.

constructing a new slider costs $\sim$69 s once per concept, after which each new prompt is steered in $\sim$29 s. Elastic-search hyperparameters are listed in Section L.

4.4 Ablations and Construction Analysis

Effect of token selection. Our method selects which prompt tokens to steer using an LLM. To test whether this selection matters, we compare three variants in Table 1: **Ours (First Token)** steers only the first prompt token, **Ours (All Tokens)** steers every token in the prompt, and **Ours (Flux.dev)**, our full method, steers only the LLM-selected tokens. Steering only the first token under-edits, because that token is rarely aligned with the target concept (DreamSim 0.06, weak edits). Steering all tokens over-edits, because the intervention becomes global and shifts unrelated content (DreamSim 0.34, large drift). Steering the LLM-selected subset balances both regimes (DreamSim 0.20), confirming that targeted token selection is necessary.

Effect of elastic range search. To isolate the contribution of elastic range search, we compare it with three alternatives on the controlled FLUX-KONTEXT evaluation, keeping all other components fixed (Table 2). **Fixed range** applies a single empirically selected global range to all prompts, without prompt-specific calibration. **Data-driven initialization** uses the maximum projection onto the concept direction (Section 3.3) without extrapolation or adaptive search. **Binary search** keeps our initialization and extrapolation stages and replaces only the elastic-band optimization with a binary search over the range endpoints. The two adaptive variants clearly outperform the non-adaptive ranges in edit strength and strict monotonicity, supporting prompt-specific calibration of steering strength. The fixed range attains the lowest MID and highest ρ, but at a substantially weaker edit (ΔVQA 0.30), consistent with the under-editing regime discussed in Section 4.1. Among the adaptive variants, elastic search achieves higher ΔVQA, lower MID, and higher %mono, while binary search reduces construction cost from 16 to 6 generations; this is a quality–cost trade-off between adaptive optimizers. Both costs are incurred once per prompt–concept pair, and faster adaptive search remains a promising direction (Section 5).

LLM dependence. Our method uses an LLM at two stages, which raises the question of whether LLM supervision is reliable or not. For token selection, we evaluate this on a held-out set of 200 prompts spanning Local, Global, and Stylization edits in both Implicit and Explicit prompt modes, with human-annotated gold tokens in Table 3 (for details see Section B). Cross-model agreement is high: in thinking mode, Qwen3-8B [45] and Qwen3.5-35B-A3B [28] agree at Jaccard† $\mathbf{0.899}$ under 1-token tolerance, and averaging over all six pairwise combinations gives a mean Jaccard† of 0.819. Thinking mode consistently improves alignment with gold (Jaccard†: 8B 0.648 $\rightarrow$ 0.712; 35B 0.729 $\rightarrow$ 0.767). We use Qwen3-8B [45] in thinking mode throughout, since it matches the 35B MoE [28] in cross-model agreement and is smaller on the total number of parameters. For contrastive-pair generation, by contrast, smaller LLMs tended to produce repetitive or poorly-debiased examples in our experiments, whereas larger models such as GPT-4.1 [22] and Qwen3.6-27B [29] reliably generated debiased datasets. We therefore use GPT-4.1-mini [22] for dataset construction throughout the paper.

Model	Mode	Jacc.†
Qwen3-8B	think	0.712
Qwen3-8B	no-think	0.648
Qwen3.5-35B-A3B	think	0.767
Qwen3.5-35B-A3B	no-think	0.729

Table 3: **Token-selection LLM sensitivity.** Token-set Jaccard† (1-token tolerance) against ground truth (gold) labels classification accuracy on 200 prompts. Mean pairwise Jaccard across models: $\mathbf{0.819}$.

4.5 Cross-Backbone Behavior

Since the intervention is applied at the text-conditioning input, the same construction extends across generators with compatible text-encoder interfaces: when models share a text encoder we reuse the steering vectors directly, otherwise we reconstruct the direction in the corresponding frozen text space. We report Qwen-Image-Edit [44] and Flux [17] quantitatively, with qualitative Qwen-Image-Edit results in Fig. 5 and additional qualitative results on Flux [17], Flux2 [18], and Wan2.1 [42] in the supplementary material. As all these backbones use different text-encoder families [18, 31, 2], it shows that the controllable semantic structure exploited by our method is not tied to a single text-encoder backbone. Preliminary experiments show our slider also works on video diffusion backbones (Section Q), but it is too early to call this a full video extension; reliable calibration awaits stronger video backbones and a DreamSim-equivalent perceptual metric, details of how sliders are constrcuted for these methods are given in Section M.

5 Discussion, Limitations & Future Work

Our work introduces a training-free framework for controllable image editing through token-aware steering and adaptive range calibration. We conclude by discussing limitations and future directions.

Limitations. Our approach assumes that the generator can produce both endpoints of a concept spectrum. This assumption can fail when the model has strong biases. For example, if the model consistently favors anatomically common structures such as five-fingered hands, it may fail to generate valid six-fingered hands even under direct prompting (see Appendix Section T). In such cases, the corresponding steering direction cannot be reliably realized. Additionally, while token selection can be handled by a relatively small reasoning model, contrastive dataset generation still benefits from a stronger LLM to avoid repetitive examples and maintain diversity in the debiased pairs. Elastic range search enforces perceptual rather than semantic constraints, so its DreamSim thresholds govern failure modes: a conservative maximum threshold can under-steer, whereas a permissive one can admit over-steering or unrelated changes. Token selection can also vary between plausible tokens or edit categories; since selections are explicit, users can inspect and adjust them (Section T). Calibration also has a cost: elastic range search takes $\sim$26 s and $\sim$14 generations per prompt (Section 4.3), and since the valid range is prompt–concept specific, it must be repeated for every new prompt. This cost cannot simply be removed, as the non-adaptive alternatives in Table 2 under-edit and produce fewer monotonic trajectories, indicating that prompt-specific calibration is necessary rather than incidental.

Future Work. Better and faster LLMs in the future would improve token selection and dataset creation. Since adaptive calibration is necessary but accounts for most of the per-prompt cost, faster adaptive range search is the most direct route to interactive slider construction. Our ablation already points to a quality–cost frontier: binary endpoint search reduces construction from 16 to 6 generations at a small loss in edit strength and monotonicity (Table 2), suggesting that cheaper optimizers can retain most of the benefit of adaptivity. Promising directions include evaluating candidate points with few-step or distilled generators and reduced resolution, warm-starting the search from ranges found for semantically similar prompts or concepts, and amortizing calibration by predicting the range directly from prompt and concept features. A complementary direction is making the search semantically aware: augmenting its perceptual objective with a semantic signal, such as a VQA or vision–language score, would allow it to reject semantic reversal, saturation, and unrelated changes during calibration rather than only at evaluation time. Preliminary results show the same text-space construction transfers to video generators (Section Q), though reliable elastic-range calibration awaits stronger video backbones and a DreamSim-equivalent perceptual metric. More broadly, we hope this work encourages controllable generation systems that replace manual dataset curation and repeated fine-tuning with interpretable, automatically constructed controls.

References

[1] Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction. *Advances in Neural Information Processing Systems*, 37:136037–136083, 2024.

[2] 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.

[3] Stefan Andreas Baumann, Felix Krause, Michael Neumayr, Nick Stracke, Melvin Sevi, Vincent Tao Hu, and Björn Ommer. Continuous, subject-specific attribute control in t2i models by identifying semantic directions. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 13231–13241, 2025.

[4] Manuel Brack, Felix Friedrich, Katharia Kornmeier, Linoy Tsaban, Patrick Schramowski, Kristian Kersting, and Apolinário Passos. Ledits++: Limitless image editing using text-to-image models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8861–8870, 2024.

[5] Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 18392–18402, 2023.

[6] Shoufa Chen, Chongjian Ge, Yuqi Zhang, Yida Zhang, Fengda Zhu, Hao Yang, Hongxiang Hao, Hui Wu, Zhichao Lai, Yifei Hu, et al. Goku: Flow based video generative foundation models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 23516–23527, 2025.

[7] Pin-Yen Chiu, I Fang, Jun-Cheng Chen, et al. Text slider: Efficient and plug-and-play continuous concept control for image/video synthesis via lora adapters. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 613–622, 2026.

[8] Yusuf Dalva, Kavana Venkatesh, and Pinar Yanardag. Fluxspace: Disentangled semantic editing in rectified flow transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2025.

[9] Stephanie Fu, Netanel Y Tamir, Shobhita Sundaram, Lucy Chai, Richard Zhang, Tali Dekel, and Phillip Isola. Dreamsim: learning new dimensions of human visual similarity using synthetic data. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pages 50742–50768, 2023.

[10] Rohit Gandikota, Joanna Materzyńska, Tingrui Zhou, Antonio Torralba, and David Bau. Concept sliders: Lora adaptors for precise control in diffusion models. In *European Conference on Computer Vision*, pages 172–188. Springer, 2024.

[11] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[12] Graeme Henkelman and Hannes Jónsson. Improved tangent estimate in the nudged elastic band method for finding minimum energy paths and saddle points. *The Journal of Chemical Physics*, 113(22):9978–9985, 12 2000. ISSN 0021-9606. doi: 10.1063/1.1323224. URL https://doi.org/10.1063/1.1323224.

[13] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in neural information processing systems*, 35: 8633–8646, 2022.

[14] Zeyinzi Jiang, Zhen Han, Chaojie Mao, Jingfeng Zhang, Yulin Pan, and Yu Liu. Vace: All-in-one video creation and editing. *arXiv preprint arXiv:2503.07598*, 2025.

[15] Ronen Kamenetsky, Sara Dorfman, Daniel Garibi, Roni Paiss, Or Patashnik, and Daniel Cohen-Or. Saedit: Token-level control for continuous image editing via sparse autoencoder. *arXiv preprint arXiv:2510.05081*, 2025.

[16] 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6007–6017, 2023.

[17] Black Forest Labs. Flux. https://github.com/black-forest-labs/flux, 2024.

[18] Black Forest Labs. FLUX.2: Frontier Visual Intelligence. https://bfl.ai/blog/flux-2, 2025.

[19] 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.

[20] Zhiqiu Lin, Deepak Pathak, Baiqi Li, Jiayao Li, Xide Xia, Graham Neubig, Pengchuan Zhang, and Deva Ramanan. Evaluating text-to-visual generation with image-to-text generation. In *European Conference on Computer Vision*, pages 366–384. Springer, 2024.

[21] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

[22] OpenAI. Introducing GPT-4.1 in the API, April 2025. URL https://openai.com/index/gpt-4-1/. Accessed: 2025-08-11.

[23] Rishubh Parihar, Or Patashnik, Daniil Ostashev, R Venkatesh Babu, Daniel Cohen-Or, and Kuan-Chieh Wang. Kontinuous kontext: Continuous strength control for instruction-based image editing. *arXiv preprint arXiv:2510.08532*, 2025.

[24] Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=UGpGkLzwpP.

[25] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. Styleclip: Text-driven manipulation of stylegan imagery. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2085–2094, 2021.

[26] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. SDXL: Improving latent diffusion models for high-resolution image synthesis. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=di52zR8xgf.

[27] Yifu Qiu, Zheng Zhao, Yftah Ziser, Anna Korhonen, Edoardo Maria Ponti, and Shay Cohen. Spectral editing of activations for large language model alignment. *Advances in Neural Information Processing Systems*, 37:56958–56987, 2024.

[28] Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

[29] Qwen Team. Qwen3.6-27B: Flagship-level coding in a 27B dense model, April 2026. URL https://qwen.ai/blog?id=qwen3.6-27b.

[30] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[31] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[32] Nate Rahn, Pierluca D'Oro, and Marc G Bellemare. Controlling large language model agents with entropic activation steering. In *ICML 2024 Workshop on Mechanistic Interpretability*.

[33] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022.

[34] Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. Steering llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15504–15522, 2024.

[35] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[36] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35:36479–36494, 2022.

[37] Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. Flashattention-3: Fast and accurate attention with asynchrony and low-precision. *Advances in Neural Information Processing Systems*, 37:68658–68685, 2024.

[38] Deepak Sridhar and Nuno Vasconcelos. Prompt sliders for fine-grained control, editing and erasing of concepts in diffusion models. In *European Conference on Computer Vision*, pages 17–31. Springer, 2024.

[39] DecartAI Team. Lucy edit: Open-weight text-guided video editing. 2025. URL https://d2drjpuinn46lb.cloudfront.net/Lucy_Edit__High_Fidelity_Text_Guided_Video_Editing.pdf.

[40] Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering language models with activation engineering. *arXiv preprint arXiv:2308.10248*, 2023.

[41] Hieu M Vu and Tan M Nguyen. Angular steering: Behavior control via rotation in activation space. *arXiv preprint arXiv:2510.26243*, 2025.

[42] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

[43] Alon Wolf, Chen Katzir, Kfir Aberman, and Or Patashnik. Continuous control of editing models via adaptive-origin guidance. *arXiv preprint arXiv:2602.03826*, 2026.

[44] 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.

[45] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[46] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, Da Yin, Yuxuan.Zhang, Weihan Wang, Yean Cheng, Bin Xu, Xiaotao Gu, Yuxiao Dong, and Jie Tang. Cogvideox: Text-to-video diffusion models with an expert transformer. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=LQzN6TRFg9.

[47] Arman Zarei, Samyadeep Basu, Mobina Pournemat, Sayan Nag, Ryan Rossi, and Soheil Feizi. Slideredit: Continuous image editing with fine-grained instruction control. *arXiv preprint arXiv:2511.09715*, 2025.

[48] Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody H Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. Sglang: Efficient execution of structured language model programs. *Advances in neural information processing systems*, 37: 62557–62583, 2024.

[49] Chenliang Zhou, Fangcheng Zhong, and Cengiz Öztireli. Clip-pae: Projection-augmentation embedding to extract relevant features for a disentangled, interpretable and controllable text-guided face manipulation. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–9, 2023.

The Unreasonable Effectiveness of Text Embedding Interpolation for Continuous Image Steering

Appendix

Additional qualitative results and video extension demonstrations are provided in the supplementary material zip accompanying this submission.

A Extended Related Work

Text-conditioned visual generation. Text conditioning has become the standard interface for modern visual generation across diffusion-based, flow-matching-based, and video generative models [33, 36, 35, 26, 13, 17, 19, 42, 44, 14, 6, 39]. These models translate natural-language prompts into conditioning representations that guide the denoising or flow trajectory. Recent flow-matching models [21, 17, 19, 42, 44, 14, 6, 39] have further improved visual fidelity, prompt adherence, and sampling efficiency by learning continuous velocity fields that map noise to data through an ordinary differential equation. Despite these advances, text prompts remain an indirect control interface: they specify a desired output state, but they do not expose an explicit coefficient for smoothly varying the strength of a semantic attribute. Our work is complementary to improvements in the generator backbone. Rather than modifying the image or video generator architecture, training objective, denoising network, or flow trajectory, we operate on the text-conditioning representation that is already provided to the generator. This design is motivated by the observation that, as text-conditioned generators become increasingly capable, their frozen text-conditioning spaces can provide useful semantic directions for continuous control when those directions are carefully constructed and calibrated.

Instruction-based image editing. Instruction-guided editing models such as InstructPix2Pix [5], LEDITS++ [4], Qwen-Image [44], Flux-Kontext [19], and Flux2 [18] enable strong text-driven image editing from free-form instructions. These models are highly effective for specifying an edit target, such as "make the image snowy" or "turn the dog into a statue." However, they typically do not provide calibrated continuous control over edit strength. Changing the wording of a prompt or adjusting classifier-free guidance can sometimes affect the output, but these controls are not designed to produce smooth, monotonic, and semantically isolated edit trajectories. This motivates slider-based editing methods, which aim to expose an explicit coefficient that controls the strength of a semantic edit. Our work focuses on the construction of such sliders, rather than on training a new instruction-following editor.

Concept-specific slider methods. Concept Sliders/Flux-Slider [10] train concept-specific LoRA adapters and use the adapter scale as a continuous control coefficient. This provides an intuitive mechanism: increasing the LoRA scale increases the strength of the learned concept. However, each new concept typically requires a concept-specific training procedure and the resulting control is tied to the target generator and adapter formulation. The trained LoRA can be cached after training, but adapting the method to new concepts or backbones still requires a new setup stage. Our method differs by constructing the steering direction from LLM-generated contrastive prompt pairs in the frozen text-encoder space, without training a concept-specific adapter or modifying generator weights. The slider coefficient acts on text-conditioning representations rather than on learned adapter weights.

Text Slider [7] and Prompt Slider [38] reduces the cost of concept-slider training by learning low-rank adaptations in the text encoder rather than in the diffusion model. This is closer to our philosophy because it recognizes the text encoder as a useful control interface. However, Text Slider still performs LoRA-style fine-tuning for the target concept and text encoder. It therefore remains a trained slider method: the continuous control is obtained by scaling learned low-rank parameters. In contrast, our method does not fine-tune the text encoder. It estimates a direction directly from frozen text-encoder representations using a difference-of-means construction, selects target prompt tokens, and calibrates the steering range at inference time. Thus, while both methods exploit text-encoder structure, Text Slider learns a low-rank controller, whereas we construct and apply a training-free steering vector.

Instruction-level slider methods. SliderEdit [47] addresses a different but related problem: fine-grained continuous control over individual instructions in instruction-based image editing. Rather than training a separate slider for each semantic concept, SliderEdit trains low-rank adapters that

disentangle components of a multi-part edit instruction and expose continuous control over instruction strength. This is powerful for instruction-based editing, especially when the goal is to adjust the relative contribution of several instructions. However, it is still a trained adapter-based framework and its controller is learned for the target editing setup. Our method instead targets open-vocabulary slider construction from only a concept string and an input prompt. We do not train a global instruction controller or adapter. Instead, for each requested slider, we generate contrastive prompt pairs, estimate a concept direction in the frozen text encoder, select the relevant prompt tokens, and calibrate the steering coefficient range.

Kontinuous Kontext [23] introduces continuous strength control for instruction-based image editing by training a strength-conditioned projection module on synthetic edit data. This directly addresses the lack of continuous control in strong editing models such as Flux-Kontext. The method can achieve strong edit compliance because the controller is trained for the target editing architecture and strength-control formulation. However, this also means that the continuous-control mechanism depends on a trained module and a dataset generation procedure. Our method is complementary: instead of learning a strength-conditioned projector for an instruction editor, we ask how far the frozen text-conditioning representation itself can be used as the control space. This avoids training a new projector and makes the method easier to apply when new text-conditioned generators appear.

AdaOr [43] controls edit intensity by modifying the classifier-free guidance origin. Standard classifier-free guidance interpolates between conditional and unconditional predictions, but in editing models the unconditional prediction can behave as an arbitrary manipulation rather than as an identity-preserving origin. AdaOr replaces or interpolates this origin with an identity-conditioned adaptive origin, enabling smoother transitions from the input image to the edited output. This is an important guidance-level approach to continuous edit control. Our method differs in both the intervention point and the construction objective. AdaOr modifies the guidance formulation of the editing model, whereas we modify the text-conditioning representation before it is passed to the generator. AdaOr controls the strength of a given text-guided edit through an adaptive guidance origin; our method constructs an open-vocabulary concept slider by estimating a semantic direction, selecting target tokens, and calibrating the coefficient interval. Thus, AdaOr is a guidance-origin method for edit-intensity control, while our approach is a text-space steering method for automated slider construction.

Text, latent, and representation-space editing. A broader line of work manipulates learned representation spaces instead of training a full editor from scratch. StyleCLIP [25] identifies semantic directions in CLIP space and uses them to manipulate StyleGAN latent codes. CLIP-PAE [49] similarly uses CLIP-guided latent directions for semantic editing. These methods established that CLIP-like representation spaces contain meaningful semantic directions, but they are primarily tied to GAN inversion or GAN-style latent editing pipelines. They do not directly address continuous slider construction for modern text-conditioned diffusion or flow-matching generators, where the text encoder output serves as conditioning for an iterative generative process. Our method inherits the general idea that semantic directions can be useful, but applies it to the frozen text-conditioning space of contemporary text-conditioned generators and focuses on automatic construction, token targeting, and coefficient calibration.

Imagic [16] is closer to modern text-conditioned editing. It optimizes a text embedding to reconstruct and edit a specific input image, followed by generator fine-tuning for image-specific editing. Imagic demonstrated that optimized textual representations can support meaningful edits and interpolation-like behavior. However, it requires image-specific optimization and generator fine-tuning, making it fundamentally different from our training-free setup. Our method does not optimize a prompt embedding for a particular image and does not fine-tune the generator. Instead, it estimates a reusable concept direction from contrastive text pairs and applies that direction to selected tokens in the input prompt.

Baumann et al. [3] identify token-level semantic directions in CLIP space for continuous subject-specific attribute control. Their method further refines the directions with test-time correction using diffusion noise predictions across timesteps. This provides a stronger correction mechanism than purely linear text-space steering, but it introduces test-time gradient-based refinement and depends on diffusion-model feedback. Our method deliberately avoids such correction. We use the frozen text encoder to estimate a direction and rely on token selection plus elastic range search to make the linear intervention usable. The distinction is therefore not that we are the first to use semantic directions,

but that we construct a calibrated slider without gradient correction, generator feedback optimization, or auxiliary training.

SAEdit [15] discovers steering directions by learning sparse autoencoders over textual embeddings. This yields interpretable and steerable latent factors, but it requires an auxiliary representation-learning stage. The learned sparse autoencoder is tied to the representation distribution of the chosen text encoder and must be trained before it can be used to discover directions. Our method avoids this auxiliary training stage. We estimate a direction directly by difference-of-means over frozen text-encoder representations generated from balanced contrastive prompt pairs. This makes the method simpler to instantiate on a new compatible text-conditioned backbone.

FluxSpace [8] provides training-free editing by manipulating internal activations of Flux-style models. It is an important training-free baseline because it avoids adapter training and can produce smooth controls for certain edits. However, its intervention is architecture-specific: it depends on accessing and manipulating particular internal representations of the Flux generator. Our method instead intervenes only in the text-conditioning representation. This makes the control mechanism independent of the denoiser or flow-transformer internals, although it still depends on having access to the text encoder and its output embeddings.

Language model steering. Steering vectors have been widely studied for controlling language model behaviors such as sentiment, formality, toxicity, refusal, or truthfulness [34, 40, 24, 1, 27, 41, 32]. These methods typically identify linear directions in the residual stream of an autoregressive language model and add them to selected hidden states to influence next-token prediction. A common setup uses activations associated with the final token or a small set of behavior-relevant positions. Our setting is different. The text encoder output is not itself decoded autoregressively; instead, it is passed as conditioning context to an image or video generator [31, 2, 18]. Semantic information is distributed across token embeddings, and the effect of steering depends strongly on which prompt tokens are modified. Therefore, last-token steering from language models does not directly transfer to text-conditioned visual generation. We adapt the general steering-vector idea to this setting by constructing a concept direction in the frozen text encoder and applying it to LLM-selected prompt tokens. This makes token targeting a central component of our pipeline.

B LLM Sensitivity Analysis for Token Selection

This section expands the headline numbers reported in the main paper (Table 3). We describe the held-out evaluation set, the metrics, the inference setup, and the per-stratum breakdown, and discuss two failure modes uncovered during the analysis.

B.1 Held-Out Evaluation Set

Construction. We curate 200 *(prompt, concept)* pairs spanning the three concept types our pipeline supports — Local Edit, Stylization Edit, and Global Edit — and the two prompt-grounding modes, Implicit (no concept-axis word in the prompt) and Explicit (a synonym, antonym, or opposite-pole word for the concept is present). All edits split across both modes. The exact cell counts are listed in Table 4. Prompts are written in the descriptive "image-prompt" style typical of slider deployment, including rendering metadata such as camera, lighting, and atmosphere phrases, so the evaluation stress-tests long-prompt behavior rather than only short-form selection.

Edit type	Implicit	Explicit	Total
Local Edit	40	40	80
Stylization Edit	30	30	60
Global Edit	30	30	60
Total			**200**

Table 4: Per-cell composition of the held-out token-selection evaluation set.

Annotation. Gold target tokens were produced by a human annotator following our token selection system prompt.

B.2 Metrics

For each example, we compare the model's predicted token set A against the gold token set B after lower-casing and punctuation stripping. We report:

- **Jaccard.** $J(A, B) = |A \cap B|/|A \cup B|$, with the convention $J(\emptyset, \emptyset) = 1$.
- **Relaxed Jaccard ($\tilde{J}$).** $\tilde{J}(A, B) = \min(|A \cap B| + 1, |A \cup B|)/|A \cup B|$. This allows a single-token mismatch per example without penalty. The relaxation is motivated by long descriptive prompts containing several mutually plausible content words: gold breaks ties arbitrarily, and a single-token disagreement on otherwise overlapping sets is judgment, not semantic disjointness. $\tilde{J}$ collapses to standard J when $|A \cap B| \geq |A \cup B| - 1$.
- **Concept-type (CT) accuracy.** Fraction of examples where the predicted CT label matches gold (3-class: Local / Stylization / Global).

For inter-model agreement, we average J (and separately $\tilde{J}$) computed pairwise on the same examples.

B.3 Models and Inference Configuration

We evaluate two open-weights reasoning LLMs at distinct capacity points:

- **Qwen3-8B** [45]: dense, 8B parameters.
- **Qwen3.5-35B-A3B** [28]: mixture-of-experts, 35B total / ∼3B active per token.

Each model is evaluated in both reasoning modes (*thinking* and *no-thinking*, controlled by the chat-template flag), giving four (model, mode) configurations. Both are served via SGLang [48] on H100 GPUs with greedy decoding (temperature 0). Generation budgets are mode-dependent: no-think runs cap at 1,024 tokens; Qwen3-8B think at 4,096 tokens; and Qwen3.5-35B-A3B think at 32,768 tokens, to accommodate the larger model's substantially longer reasoning traces.

B.4 Per-Model Results

Table 5 reports per-model accuracy and aggregated inter-model agreement.

Model	Mode	Jaccard ↑	Relaxed $\tilde{J}$ ↑	CT acc ↑
Qwen3-8B	think	0.574	0.712	**0.870**
Qwen3-8B	no-think	0.430	0.648	0.925
Qwen3.5-35B-A3B	think	0.469	0.767	0.685
Qwen3.5-35B-A3B	no-think	0.597	0.729	**0.960**
Mean pairwise (4-way)		0.591	**0.819**	—

Table 5: **Token-selection LLM sensitivity — per-model breakdown.** Per-model Jaccard and relaxed Jaccard ($\tilde{J}$, 1-token tolerance) against gold, concept-type accuracy. Bottom row: mean pairwise inter-model agreement averaged over the six pairwise combinations of the four (model, mode) configurations.

Standard J against gold ranges from 0.430 to 0.597 across the four conditions; under 1-token relaxation $\tilde{J}$ tightens to 0.648–0.767. Mean pairwise inter-model agreement is 0.591 exact and 0.819 relaxed.

B.5 Pairwise Inter-Model Agreement

Table 6 disaggregates mean pairwise $\tilde{J}$ into the six pairwise combinations underlying the bottom-row average in Table 5.

B.6 Stratified Breakdown

Table 7 stratifies mean pairwise $\tilde{J}$ by edit type and prompt-grounding type to localize the source of remaining disagreement.

	8B think	8B no-think	35B think	35B no-think
8B think	—	0.753	**0.899**	0.869
8B no-think		—	0.723	0.771
35B think			—	0.897
35B no-think				—

Table 6: **Pairwise relaxed Jaccard ($\tilde{J}$).** Cross-model same-mode pairs are bolded: in thinking mode the two models agree at $\tilde{J} = 0.899$, and in no-think mode at $\tilde{J} = 0.771$. Same-model cross-mode agreement is similarly tight (0.753 for Qwen3-8B, 0.897 for the 35B). Mixed model-and-mode pairs sit between.

Stratum	n	**Pairwise $\tilde{J}$**
by edit type		
Local	80	**0.903**
Stylization	60	0.770
Global	60	0.755
by prompt-grounding type		
Explicit	70	0.868
Implicit	70	0.823
N/A (Global)	60	0.755

Table 7: **Stratified pairwise relaxed Jaccard ($\tilde{J}$).** Local edits, where the steering target is unambiguous, are effectively solved across all four configurations. Stylization and Global edits absorb the residual disagreement: these prompts contain multiple plausible style or content descriptors, and gold resolves the tie arbitrarily.

Two patterns emerge: (i) **Local edits are essentially solved** ($\tilde{J} = 0.903$): when the steering target is a single subject noun (Implicit) or a clearly demarcated attribute word (Explicit), all four configurations converge. (ii) **Stylization and Global edits absorb the residual** ($\tilde{J} = 0.770, 0.755$): these prompts contain multiple mutually plausible content or style adjectives, gold annotation resolves the ambiguity by picking one, and even the relaxed metric cannot recover full agreement when *which two adjectives* qualify is itself a judgment call.

B.7 Failure-Mode Observations

Asymmetry between Jaccard and CT accuracy under reasoning. Thinking mode improves token-set Jaccard for both models but *degrades* concept-type classification accuracy, especially for the 35B ($0.960 \rightarrow 0.685$, -0.275). On inspection, the mechanism is that the verbose reasoning trace occasionally rewrites the model's intermediate CT label before emitting it; shorter no-think generations are more likely to commit to the first correct classification. CT accuracy directly selects the downstream rule (Local vs. Stylization vs. Global) at steering time, making this asymmetry relevant in principle to deployment. In practice, however, most of these misclassifications confuse Local with Global edits, which yield the same selected tokens on the affected prompts, so the effective downstream impact is smaller than the raw -0.275 would suggest.

C Elastic-Range Search Hyperparameters

To determine a reliable steering range $[\alpha_{\min}, \alpha_{\max}]$ for each concept, we use the elastic-range search procedure described in the main paper. This procedure iteratively samples, expands, and adjusts a set of control points along the steering axis so that (i) edits progress smoothly, and (ii) the perceptual deviation from the reference image increases at an interpretable and consistent rate. For all experiments reported here, these hyperparameters are used with the FLUX.DEV backbone.

For transparency, we describe the hyperparameters used by the search algorithm and explain their role in simple terms.

1. Shared search setup. The following hyperparameters are shared across both local edits and global/stylization edits:

- **height = 512, width = 512:** All images used during elastic-range search are generated and evaluated at a resolution of 512×512.
- **target_gap = 0.01:** The search aims to keep the DreamSim distance between neighboring control points close to this target value, so that the discovered slider produces a reasonably smooth progression of edits.
- **maximum_number_of_iterations = 25:** The maximum number of refinement iterations allowed for the search.
- **maximum_number_of_points = 10:** A hard cap on the total number of control points, preventing the search from growing excessively expensive.
- **starting_number_of_points = 4:** The search begins from four initial steering values before refinement.
- **lam = 1.0:** Balances the update objective used during point adjustment.
- **move_fraction = 1.0:** Controls how aggressively interior points are moved during each update step. In practice, this makes the procedure responsive while still stable.
- **inference_batch_size = 20:** Number of candidate images generated together during each search step.
- **embed_batch_size = 64:** Batch size used when computing perceptual embeddings for similarity-based evaluation.

2. Local edit configuration. For local edits, we use:

- **minimum_similarity = 0.05**
- **max_similarity = 0.15**
- **expand_threshold = 0.6:** a midpoint is inserted only when the largest gap exceeds the target gap by 60%, since local edits expect coarser-grained refinement.
- **epsilon = 0.25:** minimum separation enforced between adjacent control points.

The lower minimum similarity for local edits reflects the fact that these edits should remain relatively close to the original image while still producing a visible semantic change. In other words, the search accepts milder deviations from the reference image because local edits are expected to preserve most of the original content and layout.

3. Global and stylization edit configuration. For global edits and stylization edits, we use:

- **minimum_similarity = 0.15**
- **max_similarity = 0.30**
- **expand_threshold = 0.05:** a tighter expansion criterion for finer-grained refinement of the perceptually larger range.
- **epsilon = 0.1:** smaller minimum spacing between adjacent control points than for local edits, allowing denser sampling.

Compared with local edits, the higher minimum similarity threshold reflects the fact that global and stylization changes are expected to induce a stronger perceptual deviation from the original image. These edits typically affect a larger portion of the image, such as the overall appearance, illumination, or rendering style, so the valid slider range is searched in a region with more pronounced change.

D Competing Methods

We compare the continuous editing capabilities of our method with SAEdit [15], Flux-Slider [10], FluxSpace [8], SliderEdit, Kontinuous Kontext, and our image-editing variant based on Qwen Image Edit. Below, we explain the setup used for each of these methods along with implementation details where necessary.

Unless otherwise noted, SAEdit, Flux-Slider, and FluxSpace are evaluated using the standard evaluation prompt set described in K. In contrast, SliderEdit, Kontinuous Kontext, and Ours (Qwen

Image Edit) are evaluated using an instruction-based prompt format, since these are image-editing methods rather than pure text-to-image generation methods. For these instruction-based methods, we first generate the source images from the prompts in the dataset using FLUX.DEV, and then use these generated images as the image-conditioning input for the editing model. The editing instructions are written in the form: *"make the scene ⟨edit⟩ without changing the overall format"* or *"without changing the background and the identity of the [person,⟨object_of_interest⟩ make the [person,⟨object_of_interest⟩, ⟨edit⟩"* depending on the edit and the object of interest.

SAEdit For SAEdit, we use their released code implementation. As there is no clear guideline released by the authors on how to construct a dataset for calculating steering vectors, we feed their released example datasets to an LLM and ask it to generate the remaining concepts in a similar format, following the dataset structure described in K. For token steering, we steered the same token selections as our method.

FluxSlider Although Concept Sliders was originally developed for UNet-based diffusion models [10, 26], we use the extension of this method to rectified-flow transformers [17] for a fairer comparison. Specifically, we use the Flux slider implementation released by the authors[2]. For training, we follow the default hyperparameters provided in the official implementation:

- Learning rate: 0.002
- Learning rate warmup steps: 200
- Training steps: 1000
- Batch size: 1
- LoRA rank: 16
- LoRA alpha: 1

During quantitative evaluation, we use a maximum slider scale of 5. For dataset generation when training sliders, we use the system prompt provided in their official codebase[3].

FluxSpace FluxSpace is already a training-free method [8]. Therefore, we directly use their released implementation for quantitative evaluation without modification. For both local and stylization edits, we use the default hyperparameters from their public code[4].

For **stylization-based edits**, the default settings are:

- edit start timestep = 0
- edit stop timestep = 3
- edit global scale = 0.8
- edit content scale = 4.0
- attention threshold = 0

For **local edits**, the default settings are:

- edit start timestep = 3
- edit stop timestep = 30
- edit global scale = 0.5
- edit content scale = 5.0
- attention threshold = 0.5

During quantitative evaluation, we use a content scale of 5.

[2] https://github.com/rohitgandikota/sliders/tree/main/flux-sliders
[3] https://github.com/rohitgandikota/sliders/blob/main/GPT_prompt_helper.ipynb
[4] https://github.com/gemlab-vt/FluxSpace/blob/main/demo.ipynb

	Cosine Similarity
Smile	0.03
Anime	-0.01

Table 8: **Alignment with SAEdit steering vectors.** Cosine similarity between our training-free steering vectors and those produced by SAEdit's learned sparse autoencoder for two representative edits. Higher similarity indicates that both methods capture comparable semantic directions despite fundamentally different construction approaches.

SliderEdit SliderEdit is an image-editing-based method, so unlike text-to-image methods, it first applies the edit and then suppresses it to recover a controllable range. Accordingly, for comparison we evaluate SliderEdit using the scales [-1, -0.6, -0.2, 0.2, 0.6, 1]. Here, the negative direction corresponds to suppressing the applied edit, and we use the scale -1 when reporting the maximum ΔVQA result. As with the other image-editing baselines, we use the same instruction-based prompts in the form described, and the input image is the FLUX.DEV-generated image corresponding to the original dataset prompt.

Kontinuous Kontext Kontinuous Kontext is also evaluated in the image-editing setting using the same instruction-based prompts as SliderEdit and Ours (Qwen Image Edit). We first generate the source images from the dataset prompts using FLUX.DEV and provide these as image-conditioned inputs. For this method, the slider values are sampled uniformly as `np.linspace(0, 1, num=n_edit_steps)`. During quantitative evaluation, we use the value 1 as the maximum edit strength.

Ours (Qwen Image Edit) For our Qwen Image Edit variant, we use the same instruction-based prompt format as the other image-editing methods. As with SliderEdit and Kontinuous Kontext, we first generate the original images from the dataset prompts using FLUX.DEV and then use these images as the image conditioning input. This ensures that SliderEdit, Kontinuous Kontext, and Ours (Qwen Image Edit) are compared under the same editing setup, while SAEdit, Flux-Slider, FluxSpace and ours (Flux) are evaluated on the standard prompt-based benchmark described in evaluation dataset description Section K.

E Detailed Implementation for the Elastic Band Search

An example pseudocode implementation of the elastic search is provided in Listing 1.

F Alignment analysis with SAEdit vectors

SAEdit [15] extracts steering directions using a learned sparse autoencoder, while our method derives steering vectors in a fully training-free manner. To examine how these approaches relate, we compute the cosine similarity between SAEdit's vectors and ours for two representative edits (*smile*, *anime*). As shown in Table 8, the similarities are close to zero. This indicates that the two methods discover *different* directions for the same concept, highlighting that there are multiple valid ways to construct an effective steering vector, even when the numerical directions do not coincide.

G Effect of Contrastive-Pair Count

We analyze how the number of LLM-generated contrastive pairs affects the stability of the estimated steering direction. For each contrastive-pair pool size $K \in \{1, 10, 50, 100, 500, 1000\}$, we compute the cosine similarity between the resulting steering vector and a reference vector estimated with $K = 2000$ pairs. As shown in Fig. 6, the direction stabilizes rapidly: cosine similarity increases to ≈ 0.97 at $K = 50$, ≈ 0.99 at $K = 100$, and above 0.99 for $K \geq 500$. We therefore use $K = 100$ in the main experiments as a practical trade-off between direction stability and construction cost.

Listing 1: **Elastic Range Search**

```python
def elastic_band_search(p, g, dist, a_min, a_max, target_gap, eta, lam, eps,
                        N0, Nmax, Texpand, T, move_fraction, sim_min, sim_max):
    """
     Args:
       p: input prompt; g: image generator where g(p, a) applies steering a;
       dist: perceptual distance function; a_min, a_max: steering range;
       target_gap: desired perceptual gap; eta: step-size schedule;
       lam: left/right imbalance gain; eps: minimum spacing;
       N0, Nmax: initial/max control-point count; Texpand: expansion threshold;
       T: max iterations; move_fraction: min required movement;
       sim_min, sim_max: allowed similarity range to the unsteered image.
     Returns:
       X_valid: control points that satisfy the sim
    """
    X = linspace(0, a_max, N0)
    move_threshold = move_fraction * eps
    for t in range(1, T + 1):
        # Compute normalized perceptual gaps between neighbors
        G = []
        for i in range(len(X) - 1):
            gap = dist(g(p, X[i]), g(p, X[i + 1]))
            G.append(gap / target_gap)
        # EXPAND: insert midpoint where gap is largest
        k = argmax(G)
        if G[k] > Texpand and len(X) < Nmax:
            mid = 0.5 * (X[k] + X[k + 1])
            X = X[:k + 1] + [mid] + X[k + 1:]
            continue
        # MOVE: adjust interior points to equalize gaps
        moved = False
        base_step = eta(t)
        for i in range(1, len(X) - 1):
            L, R = G[i - 1], G[i]
            direction = -1 if L > R else +1
            step = base_step * (1.0 + lam * abs(L - R))
            if direction < 0:
                new_a = max(X[i-1] + eps, X[i] - step)
            else:
                new_a = min(X[i+1] - eps, X[i] + step)
            if abs(new_a - X[i]) >= move_threshold:
                X[i] = new_a
                moved = True
        if not moved:
            break
    # Filter control points by similarity to the unsteered image
    X_valid = []
    for a in X:
        sim = dist(g(p, 0.0), g(p, a))
        if sim_min <= sim <= sim_max:
            X_valid.append(a)
    return X_valid
% end
```

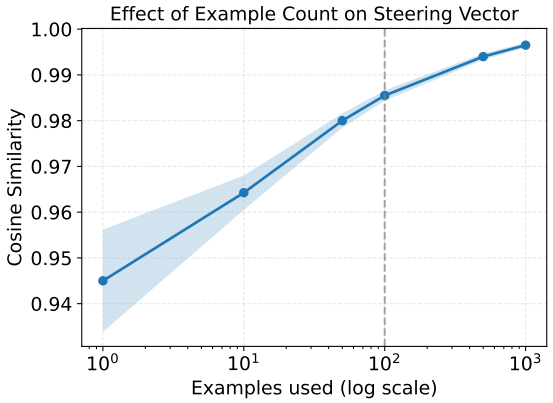

Figure 6: **Effect of contrastive-pair count on steering-vector stability.** We report cosine similarity between steering vectors estimated with K contrastive pairs and a reference vector estimated with $K = 2000$ pairs. The direction stabilizes quickly, reaching 0.99 similarity at $K = 100$, with only marginal improvement from larger pools. Embeddings are extracted from T5 [31] Encoder.

H Additional qualitative results

Additional qualitative results are provided in the attached zip of the submission in the supplementary material

I System prompts for token selection and dataset construction

We rely on two system prompts: (1) one for constructing contrastive text pairs used to estimate steering vectors, and (2) one for selecting which tokens in a free-form prompt should be steered at inference time. The full system prompts can be seen in Sections I.1 and J

I.1 Prompt for contrastive dataset generation

```
STEER_CONCEPT = "concept here"
NUMBER_OF_EXAMPLES = 100

prompt = f"""
You are an advanced data generation assistant.

Your task is to create a contrastive dataset of
{NUMBER_OF_EXAMPLES} examples for
computing a steering vector.

The steering concept to focus on is: {STEER_CONCEPT}

Output exactly {NUMBER_OF_EXAMPLES} JSON objects, one
per line (JSON Lines), with
no list brackets, no extra commentary, and no markdown.
Each line must be:
{{"pos_style": "<positive identifier>", "neg_style":
"<negative identifier>",
"pos": "<positive full sentence>", "neg": "<negative
full sentence>"}}

---
### 1. Internal Analysis (YOUR FIRST STEP)

Before generating, analyze {STEER_CONCEPT}:
* Is it an Abstract Style? (e.g., "photorealistic vs
cartoon",
"bright vs dark", "metal vs wood"). These can apply
to any subject.
* Is it a Subject-Specific Attribute? (e.g., "smiling
vs neutral" [faces],
"ripe vs unripe"
[fruit], "young vs old" [living beings/objects]). These
are tied to a class of subjects.

Based on your analysis, you MUST follow the correct
rules from Section 2.

---
### 2. Generation Guidelines (STRICT)

A. Universal Rules (Apply to ALL concepts):
* PARALLELISM: The two sentences in a pair MUST share
the same syntactic
skeleton and content words (subject, setting,
composition, perspective).
* MINIMAL DELTA: The ONLY differences between
"pos" and "neg"
are the minimal tokens
that express the concept contrast (e.g.,
"smiling" <-> "neutral").
```

```
* STYLE NEUTRALITY: Do NOT change rendering domain,
lighting, camera, or layout.
* IDENTIFIERS: Use the SAME "pos_style" and "neg_style"
identifiers for ALL

{NUMBER_OF_EXAMPLES} lines.
These identifiers MUST also appear in the
corresponding sentences.

B. Content & Subject Rules (CHOOSE A or B based on
your Analysis):

[RULE SET A] For ABSTRACT STYLES (e.g., cartoon, bright):
* SUBJECT: You MUST vary subjects and settings widely (e.g.,
objects, landscapes, animals, architecture, indoor/outdoor).
* GOAL: Decouple the style from any one context.
* SAFETY: Avoid specific identities (age, gender, race)
if people/animals are used. Use neutral terms ("person",
"animal").

[RULE SET B] For SUBJECT-SPECIFIC ATTRIBUTES (e.g.,
smiling, ripe):
* SUBJECT: You MUST focus *only* on the relevant
subject class
(e.g., "person" or "face" for smiling; "fruit" or
"plant" for ripe).
Do NOT use inanimate objects like statues for concepts like
"smiling".
* GOAL: Isolate the attribute's effect on its specific
subject.
* SAFETY: To avoid bias *within* the subject class,
use neutral, general terms
(e.g., "person," "face," "figure," "human"). Do NOT
specify age, gender, race,
or ethnicity unless it is the *target concept itself*.

---
### 3. Quality Checks (must pass):
* Exactly {NUMBER_OF_EXAMPLES} lines; each is
valid JSON.
* "pos" and "neg" are grammatical, depictable,
and differ ONLY by the
minimal concept tokens.
* The rules from Section 2 (A and B) have
been correctly followed.

---
### 4. Examples

* Example for "bright vs dark" (Abstract Style - Rule A):
{{"pos_style": "bright", "neg_style": "dark",
"pos": "A bright living room with large windows.",
"neg": "A dark living room with large windows."}}
* Example for "smiling vs neutral"
(Subject-Specific - Rule B):
{{"pos_style": "smiling", "neg_style": "neutral",
"pos": "A photorealistic portrait of a person with
a smiling expression.", "neg":
"A photorealistic portrait of a person with a ,
neutral expression."}}

Now generate {NUMBER_OF_EXAMPLES} JSON
objects that represent the
{STEER_CONCEPT} contrast following these
instructions, one per line.
"""
```

J Prompt for token selection at inference

```
You are an expert token selection assistant. Your job is to identify the
exact tokens from a PROMPT that should be steered, based on a steering
CONCEPT.

DEFINITIONS

CONCEPT TYPE:
\t•\tLocal Edit: A trait that applies to a specific subject (e.g.,
"smile", "age", "ripe", "sad", "rusty", "frosty").
\t•\tStylization Edit: A rendering style (e.g., "photorealistic",
"cartoon", "anime", "dark").
\t•\tGlobal Edit: A change to the entire scene's context or environment
(e.g., "winter", "summer", "crowded").

PROMPT TYPE:
\t•\tImplicit: The PROMPT contains NO word that lies on the same semantic
axis as the CONCEPT -- no synonym, antonym, near-synonym, or pole word.
\t•\tExplicit: The PROMPT contains at least one word that lies on the
same semantic axis as the CONCEPT. This INCLUDES:
\t\t-\tsynonyms or near-synonyms (e.g., "happy" with CONCEPT "smile";
"old" with CONCEPT "age").
\t\t-\tantonyms or opposite poles (e.g., "sad" with CONCEPT "smile";
"polished" with CONCEPT "rusty"; "clean" with CONCEPT "dirty"; "young"
with CONCEPT "age"; "clear" with CONCEPT "frosty"; "dry" with CONCEPT
"wet").
\t\t-\tstylistic poles (e.g., "photorealistic" with CONCEPT "cartoon";
"anime" with CONCEPT "realistic").
\t\tIf any such word is present in the PROMPT, the PROMPT is Explicit --
even when the word is the antonym or opposite pole of the CONCEPT.



RULES (Apply in order)
\t1.\tIf the CONCEPT is a "Global Edit":
\t\t•\t-> Output EVERY meaningful content word from the PROMPT --
adjectives ("red", "wooden", "busy", "narrow"), nouns ("cabin",
"ceremony", "stalls"), and significant modifiers. Do NOT pick only the
"main" subject; the steering needs the full scene.
\t\t•\tA Global Edit answer typically has 3-5 tokens, sometimes more. If
you find yourself outputting only 1-2 tokens for a Global Edit, you are
probably missing adjectives -- re-read the PROMPT.
\t\t•\tExclude articles, prepositions, conjunctions, and pure
scene-action verbs ("driving", "standing", "holding", "floating") unless
they are part of a named object or phrase.
\t2.\tIf the CONCEPT is a "Stylization Edit":
\t\t•\tIf the PROMPT is Explicit:
\t\t\t-> Output ALL explicit style/appearance words present in the PROMPT
(e.g., "photorealistic", "cinematic", "cartoon", "detailed", "realistic",
"flat", "vector", "toy-like"). If the PROMPT contains MORE than one style
word on the same axis as the CONCEPT, output ALL of them -- not just one.
\t\t•\tIf the PROMPT is Implicit:
\t\t\t-> Output the main subject noun(s) that the style can logically
apply to (e.g., "man", "lighthouse", "forest"). When a noun phrase has a
head noun and a modifying noun (e.g., "fishing boat", "park bench"),
include both unless the modifier is clearly a category disambiguator
only.
\t3.\tIf the CONCEPT is a "Local Edit":
\t\t•\tIf the PROMPT is Explicit:
\t\t\t-> Output only the explicit descriptive or attribute words present
in the PROMPT that lie on the same axis as the CONCEPT (e.g., "sad",
"old", "polished", "clean", "ripe"). This applies whether the word is a
synonym OR an antonym of the CONCEPT.
\t\t•\tIf the PROMPT is Implicit:
```

\t\t\t-> Output only the subject nouns that the local attribute can naturally attach to (e.g., "man", "woman", "child", "face", "fruit").
\t\tPrefer the main human, animal, or animate entity; if none, choose the most central object noun in the description.
\t4.\tHARD CONSTRAINTS (must always hold):
\t\t•\tEvery TOKEN you output MUST appear verbatim in the PROMPT. Do not invent or rephrase.
\t\t•\tIf the CONCEPT word itself (or a hyphenated/compound form of it like "frost-coated" for "frosty", or "sun-bleached" for "sunny") appears in the PROMPT, that occurrence IS the explicit pole and you SHOULD output it. Picking it is correct, NOT a violation.
\t\t•\tA non-empty PROMPT always has at least one valid token to output. Refusing to answer / returning an empty TOKENS list is itself a failure mode.
\t\t•\tExclude punctuation and purely functional words (articles, prepositions, conjunctions).
\t\t•\tReturn the minimal set of tokens required by the rules above -- but for Global Edits the minimal set is "all content words", not "the most important word".
\t5.\tLONG / DESCRIPTIVE PROMPTS -- IGNORE RENDERING METADATA:
\t\tLong prompts often contain extensive padding describing HOW the image is shot/lit/styled, not WHAT is in the scene. EXCLUDE all of these from your TOKENS output:
\t\t•\tLighting / atmosphere words: "soft afternoon light", "soft window light", "moody lighting", "natural lighting", "warm tones", "muted colors", "gentle shadows", "ambient", "atmospheric".
\t\t•\tCamera / framing / shot words: "documentary camera perspective", "carefully composed scene", "wide shot", "close-up", "from above", "low angle", "eye-level", "depth of field", "shallow focus".
\t\t•\tImage-quality / generic-style metadata: "high-resolution visual", "detailed image", "subtle background detail", "balanced composition", "calm composition", "clean composition", "realistic environmental composition", "professional rendering".
\t\t•\tGeneric narrative scaffolding: "an image of", "a scene depicting", "a high-resolution visual showing", "a carefully composed scene of", "a detailed image of", "with X and Y and Z" appendix clauses.
\t\tThese words are NOT scene content. They are not Global Edit content words, not Stylization style cues (unless the CONCEPT is itself about lighting/camera/style), and not Local Edit attribute words.
\t\tRule of thumb: the TOKENS answer for a long, descriptively-padded prompt should be the SAME as for a short, plain version of the same prompt. If a long prompt expands "a man in a park" into "a high-resolution visual of a man in a park, with soft afternoon light, a documentary camera perspective, and balanced natural lighting", the answer is unchanged -- only the man+park content survives.

EXAMPLES

Global Edit (output ALL content words, including adjectives -- typically 3+ tokens):
\t•\tPROMPT: "a woman in a park", CONCEPT: "winter" (Global Edit)
\t•\tOUTPUT:
\t\tCONCEPT_TYPE: Global Edit
\t\tTOKENS: woman park
\t•\tPROMPT: "a red car driving along a mountain road", CONCEPT: "winter" (Global Edit -- "driving" is a filler verb, drop it; keep adjective "red")
\t•\tOUTPUT:
\t\tCONCEPT_TYPE: Global Edit
\t\tTOKENS: red car mountain road
\t•\tPROMPT: "a busy market filled with fruit stalls", CONCEPT: "nighttime" (Global Edit -- keep adjective "busy")
\t•\tOUTPUT:

```
\t\tCONCEPT_TYPE: Global Edit
\t\tTOKENS: busy market fruit stalls
\t•\tPROMPT: "a wooden cabin beside a lake", CONCEPT: "winter" (Global
Edit)
\t•\tOUTPUT:
\t\tCONCEPT_TYPE: Global Edit
\t\tTOKENS: wooden cabin lake

Stylization Edit (output ALL style words when multiple are present):
\t•\tPROMPT: "a photorealistic lighthouse on a cliff", CONCEPT: "cartoon"
(Stylization Edit, Explicit -- single style word)
\t•\tOUTPUT:
\t\tCONCEPT_TYPE: Stylization Edit
\t\tTOKENS: photorealistic
\t•\tPROMPT: "a detailed realistic fishing boat floating near the
harbor", CONCEPT: "cartoon" (Stylization Edit, Explicit -- TWO style
words on the realism axis: "detailed" and "realistic")
\t•\tOUTPUT:
\t\tCONCEPT_TYPE: Stylization Edit
\t\tTOKENS: detailed realistic
\t•\tPROMPT: "a flat vector bowl of oranges on a kitchen table", CONCEPT:
"oil painting" (Stylization Edit, Explicit -- "flat" and "vector" are
both stylistic descriptors)
\t•\tOUTPUT:
\t\tCONCEPT_TYPE: Stylization Edit
\t\tTOKENS: flat vector
\t•\tPROMPT: "a toy-like dinosaur on a desk", CONCEPT: "photorealistic"
(Stylization Edit, Explicit -- "toy-like" is a hyphenated style word)
\t•\tOUTPUT:
\t\tCONCEPT_TYPE: Stylization Edit
\t\tTOKENS: toy-like
\t•\tPROMPT: "a lighthouse on a cliff", CONCEPT: "cartoon" (Stylization
Edit, Implicit)
\t•\tOUTPUT:
\t\tCONCEPT_TYPE: Stylization Edit
\t\tTOKENS: lighthouse

Local Edit (synonym/same-pole Explicit):
\t•\tPROMPT: "a portrait of a sad man", CONCEPT: "smile" (Local Edit,
Explicit -- "sad" is the antonym of "smile")
\t•\tOUTPUT:
\t\tCONCEPT_TYPE: Local Edit
\t\tTOKENS: sad
\t•\tPROMPT: "a ripe tomato on the vine", CONCEPT: "age" (Local Edit,
Explicit -- "ripe" is on the age axis)
\t•\tOUTPUT:
\t\tCONCEPT_TYPE: Local Edit
\t\tTOKENS: ripe

Local Edit (antonym/opposite-pole Explicit -- common pitfall):
\t•\tPROMPT: "a polished steel bridge crossing a narrow river", CONCEPT:
"rusty" (Local Edit, Explicit -- "polished" is the antonym of "rusty")
\t•\tOUTPUT:
\t\tCONCEPT_TYPE: Local Edit
\t\tTOKENS: polished
\t•\tPROMPT: "a clear window overlooking a city street", CONCEPT:
"frosty" (Local Edit, Explicit -- "clear" is the antonym of "frosty")
\t•\tOUTPUT:
\t\tCONCEPT_TYPE: Local Edit
\t\tTOKENS: clear
\t•\tPROMPT: "a clean metal gate at the entrance of a garden", CONCEPT:
"rusty" (Local Edit, Explicit -- "clean" lies on the same axis as
"rusty/dirty")
\t•\tOUTPUT:
\t\tCONCEPT_TYPE: Local Edit
\t\tTOKENS: clean
```

```
Local Edit (CONCEPT word literally in PROMPT -- output it; do NOT
refuse):
\t•\tPROMPT: "a wet leather boot on a studio floor", CONCEPT: "wet"
(Local Edit, Explicit -- "wet" is the CONCEPT itself, it appears in the
PROMPT, so output it)
\t•\tOUTPUT:
\t\tCONCEPT_TYPE: Local Edit
\t\tTOKENS: wet
\t•\tPROMPT: "a frosty glass bottle on a table", CONCEPT: "frosty" (Local
Edit, Explicit -- "frosty" is the CONCEPT itself; output it)
\t•\tOUTPUT:
\t\tCONCEPT_TYPE: Local Edit
\t\tTOKENS: frosty
\t•\tPROMPT: "a frost-coated truck in a quiet lot", CONCEPT: "frosty"
(Local Edit, Explicit -- "frost-coated" is a hyphenated compound on the
frost axis; pick the hyphenated word verbatim)
\t•\tOUTPUT:
\t\tCONCEPT_TYPE: Local Edit
\t\tTOKENS: frost-coated

Local Edit (Implicit):
\t•\tPROMPT: "a portrait of a man", CONCEPT: "smile" (Local Edit,
Implicit)
\t•\tOUTPUT:
\t\tCONCEPT_TYPE: Local Edit
\t\tTOKENS: man
\t•\tPROMPT: "a tomato on the vine", CONCEPT: "age" (Local Edit,
Implicit)
\t•\tOUTPUT:
\t\tCONCEPT_TYPE: Local Edit
\t\tTOKENS: tomato
\t•\tPROMPT: "a steel bridge crossing a narrow river", CONCEPT: "rusty"
(Local Edit, Implicit -- no word on the rust axis is in the prompt)
\t•\tOUTPUT:
\t\tCONCEPT_TYPE: Local Edit
\t\tTOKENS: bridge

Long descriptive prompts -- same answers as the short version (rendering
metadata ignored):
\t•\tPROMPT: "a high-resolution visual of portrait of a man in soft
window light, with soft afternoon light, subtle background detail, and a
calm documentary camera perspective", CONCEPT: "smile" (Local Edit,
Implicit -- only "man" is scene content; the rest is shot/lighting
padding)
\t•\tOUTPUT:
\t\tCONCEPT_TYPE: Local Edit
\t\tTOKENS: man
\t•\tPROMPT: "a carefully composed scene showing a red car driving along
a mountain road, with cinematic lighting, distant background detail, and
a calm documentary camera perspective", CONCEPT: "winter" (Global Edit --
drop all camera/lighting padding; keep "red car mountain road")
\t•\tOUTPUT:
\t\tCONCEPT_TYPE: Global Edit
\t\tTOKENS: red car mountain road
\t•\tPROMPT: "a detailed image of a fishing boat floating near the
harbor, with muted colors, gentle shadows, and a balanced natural
composition", CONCEPT: "cartoon" (Stylization Edit, Implicit -- "muted
colors", "gentle shadows", "balanced composition" are RENDERING metadata,
NOT style descriptors of the same axis as cartoon. Pick the subject
noun.)
\t•\tOUTPUT:
\t\tCONCEPT_TYPE: Stylization Edit
\t\tTOKENS: fishing boat

WRONG OUTPUTS -- do NOT produce these:
```

```
\t•\tPROMPT: "a mango on a kitchen counter", CONCEPT: "ripe" -> WRONG:
"TOKENS: ripe" (the word "ripe" is NOT in the prompt). CORRECT: "TOKENS:
mango".
\t•\tPROMPT: "a steel bridge crossing a narrow river", CONCEPT: "rusty"
-> WRONG: "TOKENS: rusty" (the word "rusty" is NOT in the prompt).
CORRECT: "TOKENS: bridge".
\t•\tPROMPT: "a high-resolution visual of a man in a park, with soft
afternoon light and a documentary camera perspective", CONCEPT: "winter"
(Global Edit) -> WRONG: "TOKENS: visual man park soft afternoon light
documentary camera perspective" (camera/lighting metadata is NOT scene
content). CORRECT: "TOKENS: man park".
\t•\tPROMPT: "a detailed image of a fishing boat near the harbor, with
cinematic lighting and muted colors", CONCEPT: "cartoon" (Stylization
Edit, Implicit) -> WRONG: "TOKENS: detailed cinematic muted" (these are
rendering metadata, NOT on the cartoon/realism axis when used as
scene-padding). CORRECT: "TOKENS: fishing boat".


YOUR TASK

Analyze the following PROMPT and CONCEPT using this logic. Output exactly
two lines and nothing else:
Line 1: CONCEPT_TYPE: <one of: Local Edit, Stylization Edit, Global Edit>
Line 2: TOKENS: <the tokens separated by a single space, each one taken
verbatim from the PROMPT>
Do not add any commentary, explanation, or punctuation beyond this
format.\
"""
```

K Evaluation Dataset

We construct our evaluation prompt set to cover a broad range of edit behaviors while keeping the underlying scene content diverse. The dataset used in our experiments contains a total of **198 prompts**, grouped into three high-level categories: *local edits*, *global edits*, and *stylization edits*.

The local edit category contains edits that modify a restricted semantic attribute while preserving the overall scene identity and composition. The global edit category contains edits that affect scene-level properties or broader image characteristics, often requiring coordinated changes across multiple regions of the image. The stylization category contains edits that primarily change the rendering style or visual domain of the image while preserving its semantic content. This structure allows us to evaluate whether a single continuous steering framework can generalize across qualitatively different types of edits.

Local edits. Targeted attribute changes that leave most of the image unchanged. The dataset includes the edit types *smile*, *age*, *frostiness*, *wet texture*, and *rusty surface*.

Some example prompts for local edit types from our dataset are given below:

- **Smile:**
  - "A professional passport-style headshot of a man wearing a navy suit against a neutral studio background, closed lips and a straight mouth, looking directly at the camera."
  - "A woman seated at a wooden dining table inside a softly lit apartment, looking at the camera with a neutral expression, lips closed, relaxed face."
- **Age:**
  - "A portrait of a woman standing beside a bookshelf in a cozy living room."
  - "A man standing outdoors in front of a mountain landscape."
- **Rust level:**
  - "A vintage car parked on a quiet countryside road."
  - "A metal gate standing at the entrance of an apartment."

- **Wetness:**
  - "A paved road stretching through a rural landscape."
  - "A wooden dock extending into a calm lake."

Global edits. Scene-wide or geometric transformations that require coherent changes across multiple regions. The dataset includes the edit types *winter*, *nighttime*, *rainy weather*, *sitting*, *crowded scene*, and *zoom-out camera*.

Some example prompts for global edit types from our dataset are given below:

- **Season:**
  - "A small countryside house surrounded by trees and a grassy yard."
  - "A mountain landscape with a lake and pine trees in the distance."
- **Time of day:**
  - "A busy downtown street with tall office buildings and traffic lights."
  - "A coastal harbor with fishing boats docked near wooden piers."
- **Rain intensity:**
  - "A person walking along a city sidewalk carrying a backpack."
  - "A parked car along a residential street with trees and houses."
- **Crowd density:**
  - "A city square surrounded by shops and cafes with wide walking space."
  - "A train station platform with benches and signage."
- **Camera zoom:**
  - "A portrait of a woman standing in a large open field with mountains behind her."
  - "A man standing on a rooftop overlooking a city skyline."
- **Pose sitting/standing:**
  - "A woman inside a living room near a wooden chair and a small side table."
  - "A standing man in a park near a bench under a large tree."
- **Frostiness:**
  - "A wooden cabin surrounded by pine trees."
  - "A metal railing along a city bridge."

Stylization edits. Changes to the visual appearance or artistic rendering while preserving semantic structure. The dataset includes the edit types *cartoon*, *ghibli style*, *anime*, and *photorealistic render style*.

Some example prompts for stylization edit types from our dataset are given below:

- **Cartoon:**
  - "A young woman sitting at a small outdoor cafe table in Paris, wearing a red scarf and holding a cup of coffee."
  - "A firefighter standing confidently in front of a red fire truck."
- **Anime:**
  - "An image of a teenage girl standing in a pool area."
  - "A boy walking through a quiet suburban street at dusk."
- **Ghibli:**
  - "A scene of a coffee shop surrounded by people drinking coffee and having fun."
  - "A young girl walking along a countryside path with rolling hills."
- **Photorealism:**
  - "A woman standing beside a vintage car parked on a quiet street."
  - "A cartoon style image of a surfer emerging from the ocean carrying a surfboard."

Overall, this dataset covers a wide range of edit behaviors, from restricted attribute manipulation to scene-wide transformations and appearance transfer. Such diversity is important for showing that the proposed method is not limited to one narrow class of edits, but instead supports continuous steering across multiple qualitatively different editing regimes.

L Hyperparameters Used in Runtime Calculation

For the runtime results reported in Section 4.3 of the main submission, we used a practical implementation setup optimized for fast inference. In the token pooling stage, we use a batch size of 100, which fits within a single H100 GPU in our setup. This stage is performed once per concept and then cached, so its cost does not need to be paid again for every new prompt. Similarly, dataset generation is also performed once per concept and cached.

For the elastic search stage, we use the following hyperparameter configuration for local edit:
`height=512`,
`width=512`,
`target_gap=0.01`,
`maximum_number_of_iterations=25`,
`expand_threshold=0.6`,
`maximum_number_of_points=10`,
`starting_number_of_points=4`,
`lam=1.0`,
`epsilon=0.25`,
`max_dreamsim_distance=0.04`,
`move_fraction=1.0`,
`minimum_value=0.0`,
`maximum_value=100.0`,
`inference_batch_size=20`, and
`embed_batch_size=64`.

For the elastic search stage, we use the following hyperparameter configuration for global/stylization edit:
`height=512`,
`width=512`,
`target_gap=0.075`,
`maximum_number_of_iterations=25`,
`expand_threshold=0.15`,
`maximum_number_of_points=10`,
`starting_number_of_points=4`,
`lam=1.0`,
`epsilon=1.0`,
`max_dreamsim_distance=0.21`,
`move_fraction=1.0`,
`minimum_value=0.0`,
`maximum_value=100.0`,
`inference_batch_size=20`, and
`embed_batch_size=64`.

These hyperparameters are used for the per-prompt elastic search procedure that identifies the valid continuous steering range. We do not introduce additional nontrivial hyperparameters for the remaining stages beyond the standard inference settings of the corresponding backbones.

To further improve runtime efficiency, we utilize SGLang [48] for serving and FlashAttention-3 [37] for accelerated attention computation. These optimizations reduce overhead during generation and representation extraction, and are used throughout the reported runtime measurements. Flux.Dev [17] is used for rendering images

Metric	Training-based				Training-free				
	SAEdit	Flux Slider	SliderEdit	Kontinuous Kontext	FluxSpace	Ours (First Token)	Ours (All Tokens)	Ours (Flux.Dev)	Ours (Qwen)
DreamSim↓	0.08	0.23	0.31	0.21	0.09	0.06	0.34	0.20	0.21

Table 9: **Quantitative dreamsim comparison of editing methods.** We report dreamsim distance of compared methods where lower score means closer perceptual similarity to the original image hence a more content preserving edit. Methods are grouped into *training-based* and *training-free* settings.

M Pratical Differences of Applying Our Method to Text-to-Image and Image-to-Image Backbones

A practical difference between text-to-image and image-to-image backbones is whether the model already contains an explicit mechanism for preserving the input image structure during the denoising process. In purely text-conditioned generation models such as Wan [42], CogVideoX [46], and FLUX [17], there is no internal image-conditioning that enforces structural alignment with a reference image. As a result, directly applying a large steering coefficient uniformly across all timesteps can cause the steering signal to dominate early denoising stages and alter the global scene layout. To mitigate this, we use a simple linear timestep schedule in which the effective steering strength at timestep t is scaled as $\frac{t}{T}\omega$, where T is the total number of timesteps and ω is the original steering coefficient. This design keeps the steering weak during the earliest denoising stages, allowing the model to first establish the coarse low-frequency structure of the scene, and gradually increases the intervention at later steps where high-frequency details and local attributes are refined. In practice, this is particularly helpful for localized edits, where preserving the original composition is essential. For more global edits, however, we do not use this progressive schedule and instead apply the original steering scale uniformly across all timesteps, since in such cases modifying the full generation trajectory is desirable.

For image-conditioned editing models such as Qwen Image Edit and Flux2 [44, 18], the situation is different. Since these models already receive an input image and are explicitly designed to preserve its structure while performing the requested modification, an additional timestep-based steering schedule is generally unnecessary. The image-conditioning mechanism already acts as a strong structural anchor throughout denoising. Therefore, for image-to-image pipelines we apply the full steering coefficient at all timesteps, without any progressive scaling.

There is also a conceptual difference in how the slider range is applied for the two settings. In text-to-image generation, we begin from the original prompt-conditioned generation and add the steering direction to move the sample away from the original image along the desired semantic axis. That is, increasing the steering coefficient progressively strengthens the target edit. In contrast, for image-to-image editing we first apply the intended edit using the editing model, and then use the negative steering direction, i.e., $-1 \times \omega$, to suppress or roll back the applied edit. This allows us to recover a continuous range between the fully edited result and a version closer to the original image. Hence, while text-to-image steering is used to diverge from the original generation toward the target attribute, image-to-image steering is used in the reverse direction to suppress the already-applied edit and thereby identify the valid slider range.

N Dreamsim Scores of Quantitative Comparison

The quantitative dreamsim scores of the compared methods is provided in Table 9

O Per-Category Quantitative Comparison

The per-category results of the quantitative comparison are reported in Table 10.

P Controlled Same-Backbone Comparison

The main comparison in Table 1 reports each method on the backbone it was designed for, so differences partly reflect the capability of the underlying generator or editor. To separate the effect

Edit Type	Metric	Training-based				Training-free		
		SAEdit	Flux Slider	SliderEdit	Kontinuous Kontext	FluxSpace	Ours (Flux.Dev)	Ours (Qwen)
Stylization	ΔVQA↑	0.06	0.28	0.77	0.53	0.23	0.34	0.63
	ΔDreamSim↓	0.06	0.20	0.33	0.25	0.19	0.14	0.24
Global Edits	ΔVQA↑	0.22	0.36	0.80	0.59	0.06	0.60	0.68
	ΔDreamSim↓	0.09	0.23	0.34	0.25	0.04	0.32	0.24
Local Edits	ΔVQA↑	0.32	0.44	0.68	0.35	0.22	0.43	0.61
	ΔDreamSim↓	0.09	0.26	0.26	0.13	0.07	0.16	0.15

Table 10: **Per-category quantitative comparison.** We report ΔVQA↑, measuring edit compliance, and ΔDreamSim↓, measuring perceptual deviation from the input image (lower indicates better identity preservation). Results are grouped by edit category: *stylization*, *global edits*, and *local edits*.

Regime / backbone	Method	ΔVQA↑	MID (LPIPS)↓	MID (DreamSim)↓	ρ ↑	%mono↑
I2I / FLUX-KONTEXT	Ours	0.51±0.02	**0.45**±0.06	**0.42**±0.09	0.58±0.01	**72.1**±1.8
	Kontinuous Kontext†	0.48±0.01	**0.45**±0.04	0.50±0.06	0.65±0.03	**72.1**±1.5
	SliderEdit†	**0.69**±0.05	0.47±0.03	0.50±0.09	**0.71**±0.01	71.3±1.6
T2I / FLUX.DEV	Ours	0.31±0.03	0.41±0.06	0.46±0.09	0.59±0.02	80.9±0.7
	SAEdit†	0.21±0.04	0.38±0.02	0.44±0.09	0.41±0.01	**87.1**±1.1
	Flux-Sliders†	**0.33**±0.05	0.43±0.07	0.49±0.07	**0.62**±0.04	62.2±1.0
	FluxSpace	0.14±0.03	**0.31**±0.05	**0.41**±0.07	0.52±0.03	82.0±1.8

Table 11: **Controlled same-backbone comparison.** Within each regime, all methods share the backbone, inputs, and five initial noise seeds; values are mean±std over seeds (300 examples per seed). † denotes training-based methods. Best value per regime is shown in **bold**.

of the control method from that of the backbone, we additionally evaluate all methods under two controlled regimes in which methods share the backbone, inputs, and random seeds.

Protocol. We evaluate 300 examples over the 15 concepts of Section 4: 100 local (*smile*, *age*, *frostiness*, *wet texture*, *rusty surface*), 120 global (*winter*, *nighttime*, *rainy weather*, *sitting*, *crowded scene*, *zoom-out camera*), and 80 stylization (*cartoon*, *ghibli style*, *anime*, *photorealistic render style*) examples, using both implicit and explicit prompts.

- **Image-to-image on FLUX-KONTEXT:** our method, Kontinuous Kontext, and SliderEdit share the same FLUX.DEV source images, editing instructions, and seeds.
- **Text-to-image on FLUX.DEV:** our method, SAEdit, FluxSpace, and Flux-Sliders share the same prompts and seeds.

Each example is evaluated under five shared initial noise seeds, and we report mean ± standard deviation over seeds ($n = 300$ per seed). Because this protocol differs from the one used for Table 1 (shared inputs and multiple seeds), absolute values are not directly comparable across the two tables.

Monotonicity statistics. Since MID measures the evenness of semantic change but not its direction (Section 4), we report two explicit monotonicity statistics over the six ordered slider positions. ρ is the mean per-example Spearman correlation between slider position and ΔVQA, set to 0 when the ΔVQA scores of an example are constant; higher ρ means semantic strength increases more consistently with slider position. %mono is the percentage of examples satisfying $\Delta\text{VQA}(I_{\alpha_{k+1}}) \geq \Delta\text{VQA}(I_{\alpha_k}) - 0.01$ at all five adjacent transitions, where the tolerance accounts for scorer noise. %mono is the stricter statistic, since a single local reversal disqualifies an example; conversely, a trajectory that saturates early can still pass, so %mono should be read jointly with ΔVQA and MID.

Perceptual metric. Elastic range search uses DreamSim for calibration, so a DreamSim-based continuity score is not fully independent of the search. We therefore report MID computed with LPIPS, as in the main paper, which is not involved in calibration, alongside MID computed with DreamSim, which uses the same perceptual metric for calibration and evaluation.

Results. Table 11 reports the results. On FLUX-KONTEXT, our training-free method exceeds Kontinuous Kontext in edit strength (ΔVQA 0.51 vs. 0.48) with matching %mono and LPIPS-based

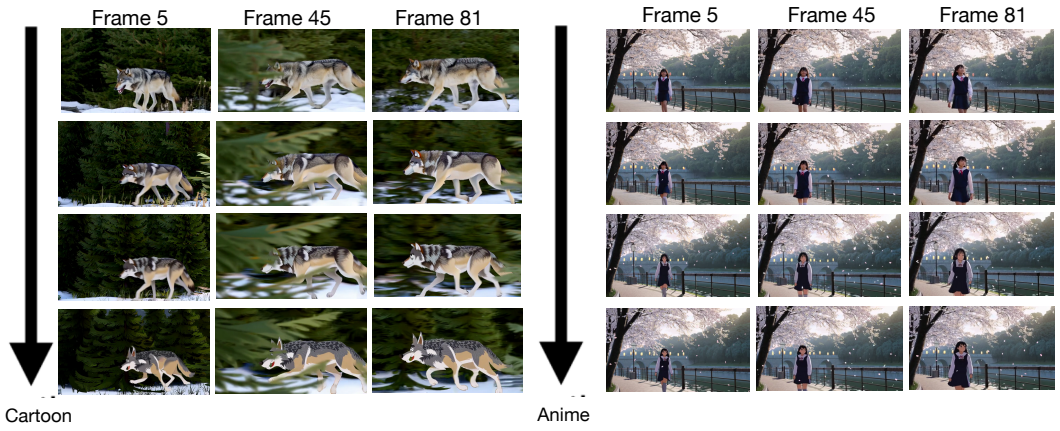

Figure 7: **Early extension to video generation (Wan2.1).** Our text-space pipeline applies to video models without retraining. Results are preliminary, reflecting the current state of video generation backbones.

MID, and the lowest DreamSim-based MID in the regime, while Kontinuous Kontext attains a higher ρ. SliderEdit remains the strongest controller in edit strength and ρ, at slightly higher MID. On FLUX.DEV, our method improves edit strength over the training-free FluxSpace and over SAEdit, and approaches the trained Flux-Sliders (0.31 vs. 0.33) with substantially higher %mono (80.9 vs. 62.2). FluxSpace and SAEdit obtain lower MID and higher %mono, but at markedly weaker edits, consistent with the under-editing behavior discussed in Section 4.1. The choice of perceptual metric does not change these conclusions: in the text-to-image regime the MID ordering is identical under LPIPS and DreamSim, and in the image-to-image regime LPIPS ties our method with Kontinuous Kontext while DreamSim favors ours.

Q Video Extension

Since our method operates entirely in the text representation space, it extends naturally to video generation models without modification. We demonstrate this on Wan2.1 [42] (We also provide .mp4 files in the attached supplementary zip) and observe that slider directions do transfer to the video domain. However, as video generation models are still maturing, the effect quality does not yet match what is achievable with image models — we view this as a promising early signal rather than a conclusive result. Figure 7 shows example generations.

R Broader Impacts

Our work proposes a training-free framework for constructing continuous image-editing sliders from a user-specified concept and prompt. The method is intended to make controllable generation more accessible by reducing the need for concept-specific training, manual dataset construction, hand-selected steering ranges, or architecture-specific interventions. As with many advances in image generation and editing, the broader impacts are mixed: the same capabilities that improve creative control and reduce development cost can also make visual manipulation easier to perform at scale. We discuss both potential benefits and risks below.

Positive societal impacts. A potential positive impact of this work is improved accessibility of controllable visual editing. Existing slider-based methods often require training concept-specific adapters, constructing datasets, tuning coefficients, or modifying internal model components. These requirements can make fine-grained editing tools expensive, technically difficult, or limited to users with substantial computational resources. By using the frozen text-conditioning space of an existing generator, our method lowers the barrier to constructing new continuous controls. This may benefit artists, designers, educators, researchers, and small organizations that want interpretable editing controls without repeatedly training new models.

The proposed approach may also reduce computational cost relative to training-based slider construction. Although our method still incurs test-time calibration cost through elastic range search, it avoids concept-specific fine-tuning and does not require storing or distributing separate adapters for each concept. This can reduce duplicated training runs when users need many lightweight controls over different visual attributes. In settings where editing concepts are exploratory, user-specific, or short-lived, avoiding repeated training may make controllable generation more efficient and practical.

Potential negative societal impacts. The main negative impact is that easier controllable image editing can facilitate misuse. A system that allows users to smoothly alter attributes, expressions, styles, settings, or scene properties may be used to create misleading or deceptive visual content. Compared with one-shot editing, continuous control can make manipulations more subtle, allowing malicious users to tune an edit until it appears plausible while still changing the intended meaning of the image. This could contribute to disinformation, harassment, impersonation, or manipulation of public perception.

The method may also amplify biases already present in the underlying generative model. Our approach assumes that the base generator can represent both endpoints of a concept and that the text-conditioning space encodes a meaningful direction between them. If the model has biased or stereotyped associations, the estimated direction may inherit those associations. For example, a concept direction for age, professionalism, beauty, or emotion could unintentionally correlate with gender, skin tone, clothing, or cultural markers if the generated contrastive prompts or the base model representations are biased. Although our method uses balanced contrastive pairs and concept-token pooling to reduce entanglement, these mechanisms do not guarantee fairness or remove all bias inherited from the LLM, the text encoder, or the image generator.

Scope. The risks discussed above arise primarily from the broader class of image-generation and image-editing systems rather than from our method alone. Our contribution does not train a new generative model, collect personal data, or introduce a new capability to synthesize images from scratch. However, it makes continuous control easier to construct for existing text-conditioned generators, and therefore may lower the effort required to perform both beneficial and harmful edits. We believe that acknowledging these impacts is important, particularly because controllability is a central goal of the work. Future research should pair improved controllable generation with stronger provenance, bias measurement, consent-aware editing policies, and misuse-resistant deployment practices.

S Compute Resources

Experiments were run on a cluster equipped with 8 NVIDIA H100 GPUs and 32 Intel CPU cores. Across the full experimental workflow, approximately 400GB of storage was used for generated images, cached embeddings, intermediate outputs, and evaluation artifacts.

We did not track total end-to-end compute usage across all exploratory, preliminary, and failed runs. Instead, we report the runtime of the main reproducible stages of the proposed pipeline in Section 4.3, including contrastive dataset generation, token pooling, token selection, and elastic range search. Dataset generation and token pooling are performed once per edit concept and cached, while token selection and elastic range search are performed per prompt.

T Limitations

Our approach assumes that the generative model can produce both ends of a concept spectrum, allowing interpolation through steering. However, this assumption does not always hold in practice due to inherent model biases. For example, if the underlying model consistently favors anatomically common structures such as five-fingered hands, it may fail to generate valid six-fingered hands even when explicitly prompted (see Fig. 8). In such cases, the steering direction corresponding to that change cannot be effectively realized. Additionally, while token selection can be handled by a relatively small reasoning model, contrastive dataset generation still benefits from a stronger LLM to avoid repetitive examples and maintain diversity in the debiased pairs. As LLMs continue to improve, we expect this dependency to weaken further, which would make our pipeline even more effective and easier to deploy. Finally, according to our knowledge, video models does not have a sufficient

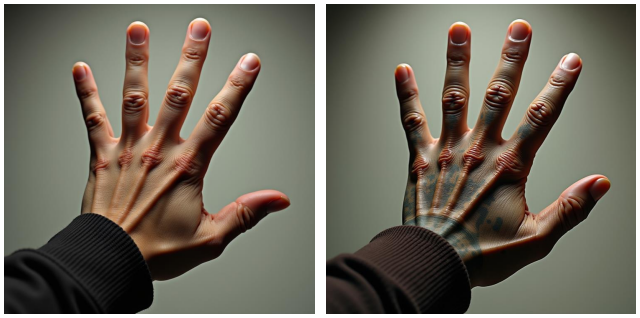

Figure 8: **Limitations.** Even when prompted with conceptually distinct descriptions (e.g., "An image of a 5 fingered hand" vs. "An image of a 6 fingered hand"), the base generative model generates only five-fingered outputs. Such inherent biases prevent us from steering between these two concepts.

enough performing perceptual similarity metric like DreamSim yet. As a result, this prohibits us right now from doing an elastic band search for video generation. It is also important to note that while we have seen early signs of our proposed methods applicability to video domain, the state of current video models are not there yet. As we claim, we argue that the gap will close compared to training based video sliders as better video models will be made by the community

Failure taxonomy. We summarize the main ways in which slider construction can become unreliable, together with mitigations.

- **Under-steering from a conservative range.** Elastic range search bounds the range with DreamSim thresholds (Section C). If the maximum threshold is too conservative for a given concept, the search stops before the edit is fully realized, yielding a smooth but weak slider. Raising the threshold, or using the global/stylization configuration for edits that require larger perceptual change, mitigates this.
- **Over-steering or unrelated changes from a permissive range.** Because the search enforces perceptual rather than semantic constraints, a permissive maximum threshold can admit endpoints where the target attribute saturates, or where perceptual change is driven by unrelated content (e.g., layout or background shifts) rather than the target concept. The search cannot by itself detect such semantic reversal or saturation; these cases are captured only by post-hoc evaluation such as ΔVQA, ρ, and %mono (Section P).
- **Token-selection variability.** For some prompts, several tokens are plausible steering targets, or the edit can reasonably be assigned to more than one category (e.g., local vs. global). Different LLMs or runs may then select different tokens (Section B), changing the edit scope. Because selections are explicit, users can inspect and adjust them when needed.
- **Unrealizable endpoints.** As discussed above, if the generator cannot produce one endpoint of the concept (Fig. 8), no steering range realizes the edit.

User Study Details

We conducted a user study with a total of 20 participants to assess perceptual edit quality across methods (ours Flux, FluxSpace, Flux-Sliders, SAEdit). The study consisted of 15 questions, each presenting four anonymized edited images corresponding to the four compared methods. For every question, participants were asked to select the image that best reflected the intended edit while preserving overall visual quality. An example question interface is shown in Fig. 9. After all responses were collected, the winner for each question was determined as the method receiving the majority of votes among the 20 participants. These per-question winners were then aggregated to compute the overall rates in Table 12.

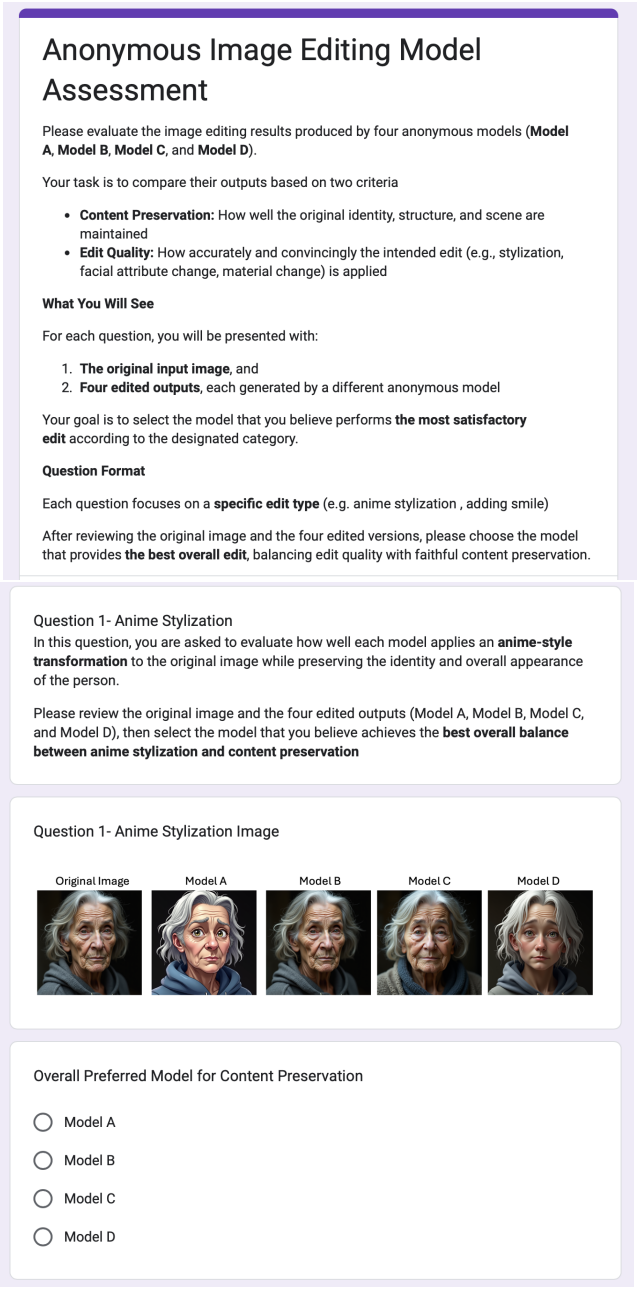

Figure 9: **Example question interface from our user study.** Participants are shown the original input image alongside four edited outputs generated by anonymous models (Model A–D). For each question, participants select the model that provides the most satisfactory edit according to the specified category, balancing edit quality with faithful content preservation. Across the full study, 20 participants answered 15 such questions, and the winning model for each question was determined by majority vote.

SAEdit	Flux-Sliders	FluxSpace	Ours (Flux)
20%	7%	13%	**60%**

Table 12: **User study.** Fraction of participants preferring each method. Best in **bold**, second in underline.

NeurIPS Paper Checklist

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

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

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

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

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

IMPORTANT, please:

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

1. **Claims**

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

   Answer: [Yes]

   Justification: Main claims are explained both in abstract and the last paragraph of the introduction

   Guidelines:

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

- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

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

   Answer: [Yes]

   Justification: Explain in Section 5

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [N/A]

   Justification: This paper does not propose a theoretical finding hence not applicable.

   Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: In Section 3 the main algorithms of the paper is given and in the appendix the psuedocode for the elastic band search and the system prompts for the LLM to be able to construct the dataset and the token finding is given.

Guidelines:

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

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

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

   Answer: [Yes]

   Justification: Code is publicly available at `https://github.com/YigitEkin/diffusion-sliders`. Implementation details, prompts, and hyperparameters are additionally provided in the appendix.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https:`

//neurips.cc/public/guides/CodeSubmissionPolicy) for more details.

- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes].

   Justification: The paper specifies the main experimental setup, including the evaluated edit categories, compared methods, backbone choices, metrics, dataset size, ablations, and runtime protocol. It also states which components are training-free and which LLMs are used for token selection and contrastive dataset generation. Additional implementation details, including elastic-range search hyperparameters, competing-method settings, inference details, dataset composition, and runtime hyperparameters, are provided in the appendix/supplementary material.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [No].

   Justification: The paper reports aggregate quantitative results for the main evaluation metrics, including ΔVQA, MID, and DreamSim. Runtime variability is reported separately as mean and standard deviation over 100 examples.

   Guidelines:

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

- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Explained in Section S

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The proposed method conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: Broader impacts are discussed in Section R

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [N/A]

    Justification: This paper does not add any additional editing capability to its built image editing pipelines as a result it is not applicable. However, the utilized backbones contain a safety filter within which has been taken account by the authors.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: The dataset used for evaluation is created by the authors, the backbone neural networks built upon are cited and properly mentioned accordingly.

    Guidelines:

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

13. **New assets**

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

Answer: [N/A]

Justification: This paper does not release any new assets

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

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

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

Answer: [N/A]

Justification: This paper does not include any human subject in its method.

Guidelines:

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

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

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

Answer: [N/A]

Justification: This paper does not include any human subject in its method.

Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [Yes]

Justification: This paper in its main core utilizes LLMs for token selection and dataset creation and this is explicitly mentioned in Section 3

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.