Text-Vision Co-Instructed Image Editing

Chenxi Xie[1,2], **Yuhui Wu**[1,2], **Qiaosi Yi**[1,2], **Lei Zhang**[1,2,†]
[1]The Hong Kong Polytechnic University, [2]OPPO Research Institute
chenxi.xie@connect.polyu.hk cslzhang@comp.polyu.edu.hk
Project Page: https://xiechenxi99.github.io/TVEdit/

Abstract

Existing image editing methods can be generally categorized into textual instruction-based and visual prompt-based ones. Textual instructions are semantically expressive, but are limited by the coarse granularity of spatial control of the editing results. In contrast, visual prompts such as drag and point can provide precise spatial guidance, but are limited by the inherent ambiguity in semantic intent. To unify the strength of textual and visual prompts, we present **Text-Vision Co-Instructed Image Editing**, which jointly models textual instructions as semantic intent and sparse visual instructions as spatial guidance, aiming to achieve precise and intent-faithful image manipulation. To this end, we first construct a textual-visual instruction paired dataset with more than 23K samples derived from dynamic videos, enabling aligned supervision for cross-modal instruction. We then propose **TV-Edit**, a **T**extual-**V**isual instruction unified **Edit**ing framework to contextualize drag or point-based visual instructions with image-text semantics and lift them into semantic-aware control representations for pretrained editing backbones. By integrating semantic intent and spatial constraints, TV-Edit leads to more precise spatial control, less instruction ambiguity, and stronger structural consistency than text-only or drag-based alternatives. Finally, we establish **TV-Edit-Bench**, a deliberately designed benchmark to evaluate semantic faithfulness, spatial alignment, and visual consistency with ground-truth references and controlled textual–visual variations for reliable assessment. Our experiments across multiple editing backbones demonstrate that TV-Edit consistently yields more precise and intent-faithful edits, significantly outperforming state-of-the-art instruction-based and drag-based baselines.

1 Introduction

Benefiting from the rapid development of text-to-image (T2I) generation models [36, 22, 31] and multi-modal large language models (MLLMs) [2, 38, 19, 8], image editing has made significant progress in recent years [16, 45, 15, 4]. In particular, instruction-based image editing [18, 21, 43] has achieved remarkable success, allowing users to specify the intent of editing through expression in natural language. These models are highly effective at manipulating semantic attributes such as color, material, and category with strong visual fidelity. However, natural language remains limited in specifying editing effects that involve spatial control or object actions, such as location, pose, shape, and motion changes. As illustrated in Fig. 1 (b), Qwen-Image-Edit [43] fails to quantify the user's intent of "slightly", leading to edits that deviate from the user's expectation.

On the other hand, drag-based editing [26, 11, 54, 44, 25, 12] offers an effective alternative by allowing precise spatial control. As a representative form of sparse visual prompt-based editing, it allows users to specify target displacements through drag points, achieving fine-grained control over object layout. However, unlike textual instruction-based editors, these methods are predominantly

[†]Corresponding author. This work is supported by the PolyU-OPPO Joint Innovative Research Center.

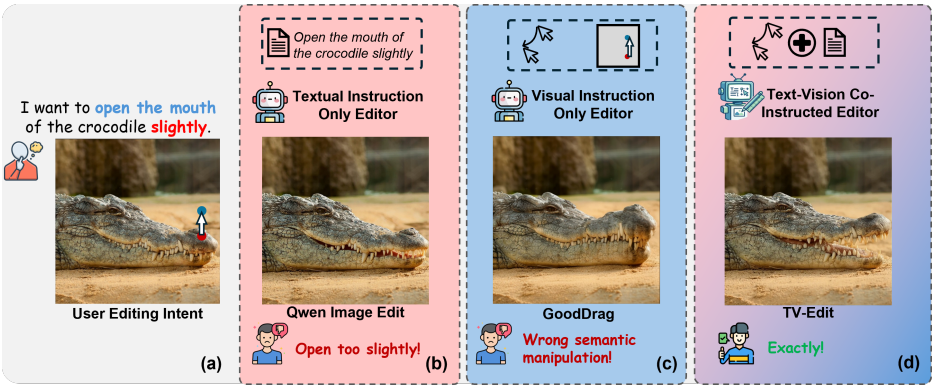

Figure 1: Comparison among different editing paradigms in terms of inputs and results. (a) User editing intent. (b) Textual instruction-based editing. [43] (c) Visual prompt-based editing [54]. (d) Our proposed text-vision co-instructed editing (TV-Editing).

designed for point-only input, which can specify local motion, but provides little semantic grounding. As shown in Fig. 1(c), when a user draws an upward arrow intending to open the crocodile's upper jaw, drag-based methods instead produce an unintended deformation of the jaw region. Although the edit satisfies the geometric constraint, it fails to capture the intended semantic action.

Therefore, these two paradigms share a common limitation: a single modality is insufficient to fully convey user intent. Textual instructions are well suited to specifying *what* a semantic transformation should occur, yet limited in determining *where* and *how* it should be implemented. Sparse visual instructions, in contrast, explicitly constrain local motion and geometry, but limited in specifying the intended semantic transformation. The representational limitation of textual and visual instructions makes it difficult for each of them alone to faithfully realize complex editing goals.

Motivated by the above observations, we propose **Text-Vision Co-instructed Image Editing**, a new task where textual instructions specify semantic intent, and sparse visual prompts impose spatial constraints. Rather than relying exclusively on textual instructions or visual prompts, we treat them as complementary signals to jointly specify the desired editing effects. Our goal is to achieve more precise and intent-faithful image manipulation by reducing the ambiguity inherent in single-modality intent expression. To this end, we introduce **TV-Edit**, a **T**extual-**V**isual instruction unified **Edit**ing framework. We first construct a large-scale textual-visual instruction paired dataset from video data, obtaining more than 23K quadruplets of {*source image*, *target image*, *point trajectory*, *text prompt*}. Leveraging the temporal continuity of videos, we derive sparse geometric cues through point tracking and optical flow, and pair them with content-aware semantic edit descriptions generated by MLLM [2]. This paired dataset explicitly binds textual semantic intent and sparse visual constraints to the same target edit, enabling unified instruction learning. We then present a decoupled Content-Aware Spatial Controller. Rather than directly injecting sparse point trajectories as geometry-only control signals [33, 53], the controller integrates image content, textual conditions, and geometric cues to produce dense control features for pretrained editing backbones. Such a design makes TV-Edit plug-and-play with modern instruction-based foundation editors, equipping them with semantically grounded spatial control. As illustrated in Fig. 1(d), our TV-Editing model produces edits that are semantically faithful, spatially aligned, and ultimately more consistent with user intent.

To systematically evaluate existing methods and our TV-Edit on this new task, we introduce **TV-Edit-Bench**. It contains 120 curated evaluation pairs drawn from multiple sources, including real videos, image-to-video generated videos, and image pairs synthesized by advanced editing models, providing rich and diverse editing intents. Each sample of TV-Edit-Bench provides aligned textual and visual instructions together with ground-truth editing targets, enabling reliable assessment of semantic faithfulness, spatial alignment, and visual consistency. Furthermore, the benchmark features two sub-tasks for geometry disambiguation and semantic disambiguation, explicitly testing a method's ability to follow both types of constraints.

We apply TV-Edit to popular editing foundation models, including Qwen-Image-Edit [43] and FLUX.1 Kontext [18]. Experimental results show that TV-Edit consistently outperforms existing state-of-the-art instruction-based and drag-based competitors, producing edits that are more faithful to the semantic intent and the spatial constraints specified by users.

2 Related Work

Textual Instruction-based Editing has evolved from early prompt-based methods [45, 13, 24] to instruction-based models. Early approaches rely on pre-trained text-to-image (T2I) models via inversion and regeneration, requiring carefully aligned source and target prompts. As a more effective alternative, instruction-based editing replaces cumbersome prompt pairs with direct commands. Pioneering works like InstructP2P [4, 49, 47] train models on paired datasets via diffusion or flow matching, bypassing per-instance inversion. Recently, foundation models such as FLUX.1 Kontext [18] and Qwen-Image-Edit [43] have substantially improved instruction following and visual fidelity. However, these models still struggle with edits involving complex actions and continuous motions. To tackle this, ByteMorph [6] introduces a dataset and baseline for non-rigid motions, while MotionEdit [40] proposes a motion-centric benchmark and post-training framework. Despite these advances, natural language inherently lacks the precision to describe fine-grained dynamics (e.g., exact motion magnitudes or trajectories), often leading to under-specified spatial realizations. This fundamental limitation motivates our formulation of text-vision co-instructed editing.

Visual Prompt-based Editing enables intuitive image manipulation through sparse visual inputs such as points, strokes, and sketches [53, 26]. Among these paradigms, drag-based editing has become one of the most representative forms. Existing diffusion-based drag methods can be broadly divided into two categories. Optimization-based methods [11, 44, 54, 25] rely on inversion and test-time latent optimization, whereas training-based methods [33, 53] learn motion priors from videos and therefore enable more efficient inference. However, both categories primarily enforce geometric constraints without explicitly modeling semantic intent, making edits ambiguous. Recent works [12, 55, 41] have begun to incorporate semantics, but their progress remains limited. CLIP-Drag [12] introduces global text guidance, but its coarse CLIP-based [29] gradients fail to align reliably with fine-grained trajectories. Drag-Flow [55] leverages DiT priors [17] and MLLM assistance, yet requires dedicated mask inputs and only resolves task-level ambiguity. Moreover, these methods still operate within an optimization-based drag-editing framework, making them sensitive to per-instance hyperparameters. More fundamentally, they treat semantics as an auxiliary extension to drag-based editing and are evaluated mainly on drag-oriented benchmarks that emphasize geometric accuracy. In contrast, we formalize text-vision co-instructed editing as a unified task and establish a dedicated framework, dataset, and benchmark for jointly evaluating semantic alignment and spatial controllability, enabling stable, mask-free, and efficient editing.

3 Text-Vision Co-Instructed Editing

3.1 Problem Formulation

Conventional image editing is typically formulated as a conditional mapping problem, i.e., $\hat{\mathcal{I}}_{tgt} = f_\theta(\mathcal{I}_{src}, c)$, where $\mathcal{I}_{src}$ is the source image and condition c is either a textual instruction or a sparse visual prompt. Such a formulation is inherently under-specified for complex edits, where a single modality-based prompt cannot fully express user intent. In this work, we propose **Text-Vision Co-Instructed Image Editing** (TV-Edit), where the desired edit is specified jointly by textual and visual instructions. Given a source image $\mathcal{I}_{src} \in \mathbb{R}^{H\times W\times 3}$, a textual instruction $\mathbf{t}_{\text{txt}}$ describing the intended semantic transformation, and a sparse visual prompt $\mathcal{P}_{\text{vis}} = \{(\mathbf{p}_k^{\text{src}}, \mathbf{p}_k^{\text{tgt}})\}_{k=1}^{K}$, where $\mathbf{p}_k^{\text{src}}, \mathbf{p}_k^{\text{tgt}} \in \mathbb{R}^2$ denote the source and target coordinates of the k-th control point, the goal is to learn a model to generate an edited image as follows:

$$\hat{\mathcal{I}}_{tgt} = f_\theta(\mathcal{I}_{src}, \mathbf{t}_{\text{txt}}, \mathcal{P}_{\text{vis}}). \tag{1}$$

The edited image $\hat{\mathcal{I}}_{tgt}$ is expected to satisfy three requirements: (i) semantic faithfulness to $\mathbf{t}_{\text{txt}}$; (ii) spatial alignment with $\mathcal{P}_{\text{vis}}$ while maintaining locally coherent semantic transformations; and (iii) global coherence with $\mathcal{I}_{src}$ beyond the primary edited regions.

3.2 Text-Vision Paired Data Construction

Since existing editing datasets [49, 33, 47] are designed mainly for single-modality supervision, we thus construct a textual-visual instruction paired training dataset to provide cooperative supervision to learn our TV-Edit model. Considering that videos naturally contain rich and continuous motion

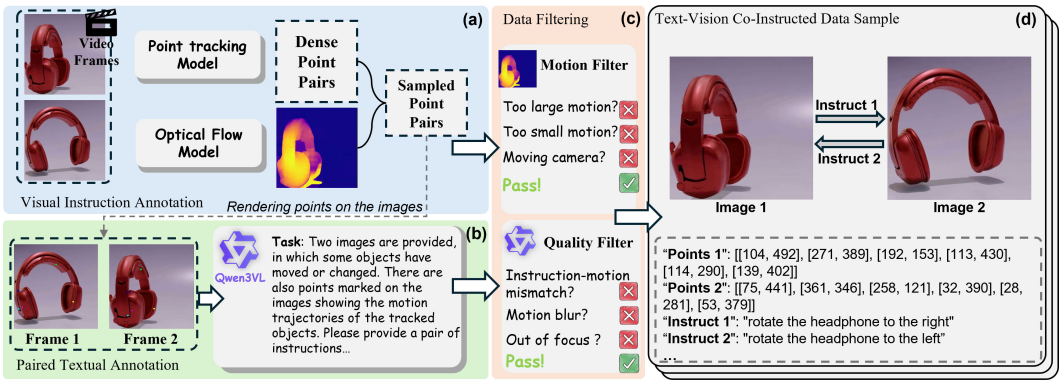

Figure 2: **TV-Edit-23K data construction pipeline.** Given two video frames, we first perform (a) visual instruction annotation to obtain sparse point pairs. We then conduct (b) paired textual annotation on the image pair with rendered points to get motion-focused instructions aligned with the marked points. After (c) data filtering, we obtain (d) the text-vision co-instructed data.

changes and deformations, we collect video data from open source video datasets [46, 30, 1] to build our dataset. Specifically, we segment full videos into clips with varying temporal strides and extract the initial and final frames. From the pairs of initial and final frames, we construct training quadruplets through three stages: visual instruction annotation, paired textual annotation, and data filtering, which are illustrated in Fig. 2 and detailed as follows.

Visual Instruction Annotation. As shown in Fig. 2 (a), we use the SEA-RAFT [42] and the Co-Tracker-V3 [14] to obtain the optical flow magnitude map and the dense grid point trajectories, respectively. As in [33, 53], the normalized flow-magnitude map is utilized as a spatial sampling weight to filter dense points, which retains significant movements while preserving small displacements, yielding sparse point pairs that accurately track cross-frame motion.

Paired Textual Annotation. Directly prompting an MLLM with raw image pairs often yields descriptions misaligned with the intended motion, due to complex video dynamics and inherent MLLM hallucinations. Inspired by recent works [34, 5], we introduce a visual prompting strategy. As shown in Fig. 2 (b), we render the points filtered in the first stage onto the image pairs using different colors and design corresponding prompts to guide the MLLM. This ensures that the MLLM focuses exclusively on the desired dynamics, thereby achieving aligned text-vision annotations.

Data Filtering. To ensure the quality of generated samples, we perform motion, alignment, and visual quality filtering. Because image editing focuses on static scenes, we address the common issue of global video dynamics by thresholding optical flow maps. Specifically, we discard samples where the frame, particularly the boundary regions, shows significant optical flow. This ensures that we only keep images with static backgrounds. To verify text annotations, we introduce a closed-loop generate-then-verify paradigm [48] to minimize MLLM hallucinations. Lastly, we perform a basic screening for motion blur and overall image quality to finalize the dataset.

Ultimately, we collect 23K high-quality sample groups, which build our dataset, namely **TV-Edit-23K**. As illustrated in Fig. 2(d), each raw sample group, denoted as $(I_1, I_2, p_1, p_2, T_1, T_2)$, can be decoupled into two bidirectional editing pairs for supervision. Specifically, they are formulated as the forward mapping $(I_1, p_1, p_2, T_1) \rightarrow I_2$ and the backward mapping $(I_2, p_2, p_1, T_2) \rightarrow I_1$. The constructed TV-Edit-23K dataset encompasses not only diverse scenes but also rich semantic motion transformations and varying motion magnitudes. It provides high-quality, text-vision paired supervision for the training of our TV-Edit model. More details can be found in **Appendix A**.

3.3 TV-Edit Model Training

Model Architecture. Recent textual instruction-based editing methods [43, 18, 21] have demonstrated strong semantic modeling abilities to understand user instructions; however, their editing process lacks fine-grained spatial controllability. Therefore, rather than training a model from scratch or fine-tuning a base text-to-image (T2I) model, we introduce a control branch to inject spatial intent, thereby guiding the editing foundation model to produce geometrically grounded results.

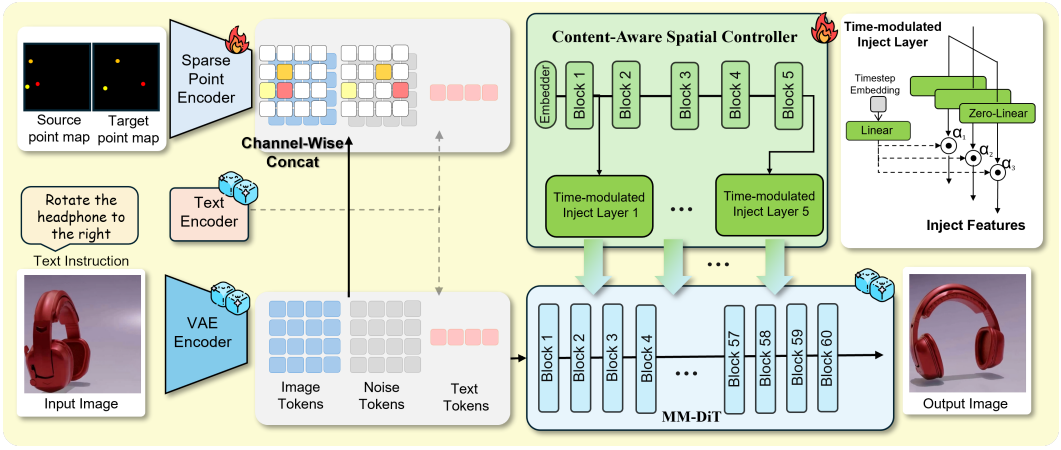

Figure 3: **The architecture of our TV-Edit.** TV-Edit consists of a main branch and a control branch. In the main branch, image, text, and noised latent tokens are processed by the editing backbone. In the control branch, source and target point maps are encoded and concatenated with image and noised latent tokens, grounding sparse trajectories with image content. A lightweight Content-Aware Spatial Controller then performs an early fusion and produces control features for the backbone.

As shown in Fig. 3, our TV-Edit model consists of two branches: a main editing branch, which includes a VAE encoder, a text encoder and a Multi-Modal Diffusion Transformer (MM-DiT), and a control branch, which includes a sparse point encoder and a content-aware controller. Given a source image $\mathcal{I}_{src} \in \mathbb{R}^{H\times W\times 3}$, a textual instruction $\mathbf{t}_{\text{txt}}$, and a sparse visual prompt $\mathcal{P}_{\text{vis}} = \{(\mathbf{p}_k^{\text{src}}, \mathbf{p}_k^{\text{tgt}})\}_{k=1}^{K}$, where $\mathbf{p}_k^{\text{src}}, \mathbf{p}_k^{\text{tgt}} \in \mathbb{R}^2$ denote the source and target coordinates of the k-th control point, the source image is first encoded into image latent tokens $\mathbf{Z}_{\text{img}} \in \mathbb{R}^{\frac{H}{8}\times\frac{W}{8}\times C}$ by the VAE encoder, and the textual instruction is encoded into text tokens $\mathbf{X}_{\text{txt}}$ by the text encoder. In the main editing branch, $\mathbf{Z}_{\text{img}}$, $\mathbf{X}_{\text{txt}}$, and the noised latent tokens $\mathbf{Z}_{\text{noise}}$ are fed into the MM-DiT backbone for denoising.

In the control branch, the sparse visual prompt $\mathcal{P}_{\text{vis}}$ is first rendered as two spatial maps, $\mathbf{M}_{\text{src}}, \mathbf{M}_{\text{tgt}} \in \mathbb{R}^{H\times W\times 1}$. To preserve the correspondence between source and target points, the pixel value at each control point is set to its trajectory index, while all the other locations are set to zero:

$$\mathbf{M}_c[x, y] = \begin{cases} k, & \text{if } (x, y) = \mathbf{p}_k^c, \\ 0, & \text{otherwise,} \end{cases} \qquad c \in \{\text{src}, \text{tgt}\}. \tag{2}$$

The two maps are then processed by a sparse point encoder composed of lightweight convolutional layers. The encoder produces spatially compressed point embeddings $\mathbf{E}_{\text{src}}, \mathbf{E}_{\text{tgt}} \in \mathbb{R}^{\frac{H}{8}\times\frac{W}{8}\times C}$, which have the same spatial resolution as the image latent. The source and target point embeddings are concatenated with the image latent and noised latent, respectively, followed by linear projection:

$$\hat{\mathbf{Z}}_{\text{img}} = \phi\big([\mathbf{Z}_{\text{img}}; \mathbf{E}_{\text{src}}]\big), \qquad \hat{\mathbf{Z}}_{\text{noise}} = \phi\big([\mathbf{Z}_{\text{noise}}; \mathbf{E}_{\text{tgt}}]\big), \tag{3}$$

where $[\cdot;\cdot]$ denotes channel-wise concatenation and $\phi(\cdot)$ is a linear projection for channel compression.

The above operations ground the semantic-agnostic point correspondences in the source image content. As a result, the controller can interpret spatial prompts in a content-dependent manner rather than taking them as geometry-only signals. The controller then produces several residual control features and injects them into the MM-DiT backbone in a ControlNet-like manner to guide the spatial composition of the edited image. The plug-and-play design of our TV-Edit model allows it to be seamlessly integrated into multiple popular editing foundation models. By optimizing only a small number of trainable parameters, TV-Edit can leverage the strong priors of the pretrained backbone while maintaining high training efficiency.

Lightweight Content-aware Spatial Controller. Directly injecting sparse spatial cues into the main diffusion backbone is suboptimal, as the foundation model struggles to align rigid, semantic-agnostic spatial relations with highly semantic visual features. To address this, we design a content-aware spatial controller. By performing early fusion over spatial cues, image content, and textual conditions, it produces content-aware guidance features to accurately guide the editing process.

Inspired by ControlNet [51], we adopt MM-DiT blocks aligned with the backbone as the controller. To keep the controller lightweight, we reduce its complexity in two ways. First, we halve the hidden dimension, which reduces the parameter count by nearly 75%. Second, we restrict the number of network blocks to N, which is significantly less than the number of blocks, denoted by N_{main}, in the main backbone. However, this aggressive compression may reduce the expressiveness of the controller outputs, weakening their ability to regulate the backbone effectively. To mitigate this issue, we introduce a *time-modulated inject layer*. This design adaptively regulates the injection intensity at different semantic depths via block-wise scaling, while dynamically aligning the control influence with the temporal evolution of the diffusion process. For the output $\mathbf{H}_n$ of the n-th controller block, we attach three block-specific linear projection heads $\phi_{n,i}(\cdot)$, where $i \in \{1, 2, 3\}$. In parallel, we predict three timestep-dependent modulation coefficients from the timestep embedding $\mathbf{e}_t$:

$$\alpha_{n,i} = g_{n,i}(\mathbf{e}_t), \qquad i \in \{1, 2, 3\},\ n \in \{1, \ldots, N\}, \tag{4}$$

where $g_{n,i}(\cdot)$ is a learnable mapping from the timestep embedding to a scalar coefficient. Each projected feature is then scaled by its corresponding coefficient:

$$\mathbf{F}_{n,i} = \alpha_{n,i}\, \phi_{n,i}(\mathbf{H}_n), \qquad i \in \{1, 2, 3\},\ n \in \{1, \ldots, N\}, \tag{5}$$

where $\mathbf{F}_{n,i}$ denotes the resulting modulated control feature used for injection into the main backbone.

Finally, there are a total of $3N$ modulated inject features. To match the number of blocks N_{main} in the main backbone, the output features from each controller block are repeated $\frac{N_{\text{main}}}{3N}$ times before being injected into the corresponding layers of MM-DiT. In practice, each backbone block receives a residual control feature, enabling dense guidance throughout the denoising process.

Training Strategy. As shown in Fig. 3, during training we freeze the main branch and optimize only the sparse point encoder and the controller from scratch. Our task prioritizes spatial layout, which is predominantly determined in the high-noise regime. We therefore adopt the $\mathbf{Z}_0$-prediction objective, as it equates to a t^2-weighted velocity loss that intentionally assigns larger weights to large t:

$$\mathcal{L}_{\text{fm}} = \mathbb{E}_{t,\mathbf{Z}_0,\mathbf{Z}_1}\left[\left\|\hat{\mathbf{Z}}_0 - \mathbf{Z}_0\right\|_2^2\right], \qquad \hat{\mathbf{Z}}_0 = \mathbf{Z}_t - t \cdot v_\theta(\mathbf{Z}_t, \mathbf{Z}_{\text{img}}, \mathbf{X}_{\text{txt}}, \mathbf{E}_{\text{src}}, \mathbf{E}_{\text{tgt}}, t), \tag{6}$$

where $\mathbf{Z}_0$ is the clean latent of the target image, $\mathbf{Z}_1$ is the sampled Gaussian noise, and $\mathbf{Z}_t = (1-t)\mathbf{Z}_0 + t\mathbf{Z}_1$. Furthermore, to explicitly reinforce this structural focus, we replace uniform sampling with a Beta distribution $t \sim \text{Beta}(\alpha, \beta)$ biased toward larger timesteps during early training. This bias is gradually relaxed as training progresses. In practice, this strategy improves both convergence speed and controllability. More details are provided in the **Appendix B**.

4 Experiments

4.1 Experimental Setup

Implementation Details. We implement TV-Edit on both Qwen-Image-Edit [43] and FLUX.1 Kontext [18], demonstrating its generality across different editing backbones. TV-Edit is trained for 80K iterations with AdamW [23] as the optimizer, using a learning rate of $1e^{-4}$. The effective batch size is set to 64 via gradient accumulation. The Beta noise sampling schedule is annealed from $\text{Beta}(20, 2)$ to $\text{Beta}(5, 2)$ over the first 40K iterations and kept fixed afterwards. We use a number of $N = 5$ controller blocks, which provide a good trade-off between controllability and efficiency. Training is conducted on 16 NVIDIA A800 GPUs.

Compared Methods. TV-Edit is a plug-and-play extension to instruction-based foundation editors, aiming to improve their spatial control capability. Note that TV-Edit does not sacrifice the foundation models' original semantic editing ability, such as adding/deleting objects or changing their attributes. We thus compare TV-Edit against two groups of related baselines: drag-based methods and advanced instruction-based editing models. For drag-based methods, we compare with a few representative and state-of-the-art approaches, such as optimization-based methods DragDiffusion [25] and GoodDrag [54], and training-based method LightningDrag [33]. For instruction-based models, we compare against state-of-the-art editing models, including LongCat-Image-Edit [37], FLUX-Kontext [3] and Qwen-Image-Edit [43]. In addition, we include MotionEdit [40], which is specifically designed for motion scenarios, and the powerful closed-source commercial model NanoBananaPro [8].

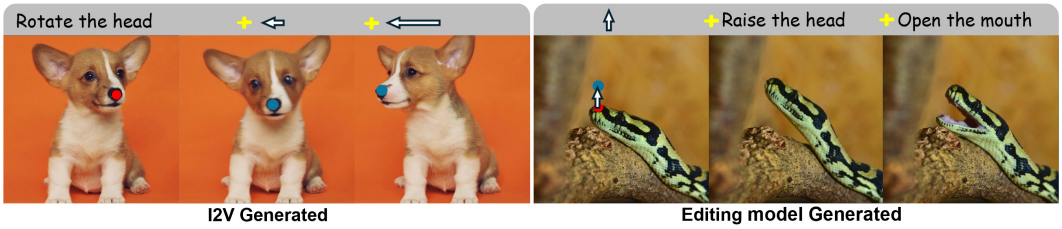

Figure 4: Examples of the two sub-tasks in TV-Edit-Bench. Left: same textual instruction with different motion magnitudes. Right: similar visual prompts with different semantic instructions.

Evaluation Benchmarks. Since there is no existing benchmark available for text-vision co-instructed editing models, we carefully constructed such a benchmark, namely **TV-Edit-Bench**. The details are provided in Sec. 4.2. On the other hand, since TV-Edit is designed for spatially related editing, we further evaluate it on established drag-based benchmarks to verify the generalization of its spatial control ability. Quantitative and qualitative results are provided in the **Appendix F**.

4.2 TV-Edit-Bench

We construct **TV-Edit-Bench**, a benchmark for evaluating text-vision co-instructed editing with paired textual and visual instructions. For the limit of space, here we provide the basic information of its *Data Curation* process and *Evaluation Protocol*. More details are provided in the **Appendix C**.

Data Curation. We collect candidate image pairs from three sources, each serving a different purpose. First, to capture realistic scene dynamics and natural motion patterns, we sample frame pairs from real videos [46]. Then, to explicitly evaluate the model's capability in fine-grained multimodal control, we further design two specific sub-tasks. The first sub-task tests motion magnitude control under fixed semantic intents. We use Wan2.2 [39] to generate examples where the source image undergoes identical actions but with varying spatial extents (see Fig. 4, left). The second sub-task tests semantic disambiguation under similar visual prompts. We use NanoBanana Pro [8] to create instances with similar trajectories but distinct semantic transformations (see Fig. 4, right). Note that these pairs are strictly filtered and manually checked before used as benchmark cases.

With the collected image pairs, we follow a pipeline similar to Fig. 2 to annotate sparse trajectories and textual instructions with a strict manual screening to retain only high-quality samples with consistent images, trajectories, and instructions. In total, TV-Edit-Bench contains 120 carefully curated samples. Each sample provides paired textual and visual prompts together with a ground-truth edited target. We further include auxiliary annotations such as masks and descriptions to facilitate inference and evaluation for different editing methods.

Evaluation Protocol. We evaluate TV-Editing from three complementary perspectives: image fidelity, geometric accuracy, and semantic faithfulness.

 # Image fidelity. We assess image fidelity with LPIPS [52] against the ground-truth target image. To reduce the influence of pixel misalignment, we further introduce two DINOv3-based [35] metrics , namely $\mathrm{DS}^{\mathrm{tgt}}_{\mathrm{global}}$ and $\mathrm{DS}^{\mathrm{tgt}}_{\mathrm{local}}$, which measure global and local visual consistency, respectively.

 # Geometric accuracy. We evaluate geometric accuracy by measuring the displacement error between matched points in the edited image and the target points. Specifically, we report a sparse matching distance MD_s and a dense matching distance MD_d, both normalized by the image size, where matching is performed using high-resolution DINOv3 features within restricted local regions around the handle and target points to avoid false matches.

 # Semantic faithfulness. We use an MLLM-based evaluator to assess whether the edited result is semantically faithful to the intended edit. Following recent MLLM-based evaluation practices [10, 9], we report two scores, concept preservation (CP) and prompt following (PF).

4.3 Main Results

Quantitative Results. In Table 1, we present a quantitative comparison of drag-based methods, instruction-based methods, and our TV-Edit on the TV-Edit-Bench. One can see that Drag-based methods achieve strong geometric accuracy, but their image fidelity is often compromised. Fur-

Table 1: Quantitative comparison on TV-Edit-Bench. M, P, T, I denote mask, prompt, trajectory and instruction respectively. The best and second-best results are highlighted in **bold** and underlined.

Method	Input	Image Fidelity			Geometric Accuracy		MLLM Score	
		$DS^{tgt}_{global}\uparrow$	$DS^{tgt}_{local}\uparrow$	$LPIPS^{tgt}\downarrow$	$MD_s\downarrow$	$MD_d\downarrow$	PF↑	CP↑
DragDiffusion [25]	M+P+T	.8290	.9288	.2490	.1044	.1048	0.72	0.98
GoodDrag [54]	M+P+T	.8724	.9305	.2234	.0630	.0648	0.75	0.96
DragLoRA [44]	M+P+T	.8444	.9207	.2380	.0638	.0671	0.77	0.94
CLIPDrag [12]	M+P+T	.8741	.9301	.2312	.1383	.1363	0.71	0.95
GeoDrag [28]	M+P+T	.8710	.9059	.2904	.0775	.0776	0.79	0.94
LightningDrag [33]	M+T	.8254	.9155	.2633	.0649	.0689	0.79	0.93
FLUX-Kontext [18]	I	.8740	.9302	.2728	.1626	.1613	0.82	0.98
Qwen-Image-Edit [43]	I	.8575	.9191	.3123	.1379	.1380	0.86	0.97
LongCat-Image-Edit [37]	I	.8550	.9185	.3391	.1247	.1229	**0.93**	0.99
MotionEdit [40]	I	.8459	.9166	.2943	.1228	.1173	0.92	0.97
NanoBananaPro [8]	I	.9096	.9432	.2072	.1201	.1195	0.89	**1.00**
TV-Edit-Kontext	I+T	**.9134**	**.9514**	.1696	.0484	.0508	0.86	0.99
TV-Edit-Qwen	I+T	**.9134**	.9490	**.1672**	**.0421**	**.0462**	**0.93**	**1.00**

thermore, despite precise trajectory tracking, their PF scores remain low, indicating poor semantic execution. Specifically, while the leading method GoodDrag[54] achieves an excellent MD_d of 0.0648, its PF score is only 0.75. This reveals that drag-based methods can reliably control *where* to edit, but struggle to determine *what* semantic action to perform.

Instruction-based models exhibit the opposite trend. They achieve strong image fidelity and semantic faithfulness, as reflected by NanoBanana Pro, which reaches 0.9432 on $\mathrm{DS}^{\mathrm{tgt}}_{\mathrm{global}}$, nearly 1.0 on CP, and 0.89 on PF. However, relying entirely on textual instructions without explicit spatial guidance makes their geometric changes highly unpredictable. Consequently, their MD_d consistently exceeds 0.10. In other words, these models can determine *what* semantic edit to perform, but cannot precisely control *where* it is spatially realized.

By contrast, our TV-Edit significantly bridges this gap, achieving high geometric accuracy and semantic faithfulness simultaneously. TV-Edit-Qwen reduces MD_d to 0.0462, achieving a 28.7% improvement over the best drag-based method and demonstrating excellent spatial precision. Beyond spatial control, it elevates the PF score from 0.86 to 0.93 compared to its base model Qwen-Image-Edit, even surpassing the closed-source model NanoBanana Pro. This indicates that the visual input not only provides explicit geometric control, but also works collaboratively with textual guidance to improve semantic execution. Notably, TV-Edit is mask-free but still preserves strong image fidelity, leading to stable and faithful edits.

Qualitative Results. We first present visual comparisons with state-of-the-art methods to highlight TV-Edit's superiority in accurate semantic editing and precise spatial control, then demonstrate its flexibility on fine-grained multimodal control by the two sub-tasks.

In Fig. 5, we present several representative spatially related editing cases, including rotation, non-rigid action, scale change, and translation. Drag-based methods can produce reasonable results when there is little semantic ambiguity of the editing trajectory. For example, in the 1st row, most drag-based baselines correctly rotate the boy's head. However, once the visual prompt admits multiple semantic realizations, these methods often fail to capture the intended action. In the 2nd row, instead of opening the fox's mouth, drag-based methods distort the facial region or produce implausible edits. In addition, the optimization-based point-tracking methods struggle with large motions. In the 4th row, GoodDrag attempts to deform the dog rather than translate it, while LightningDrag moves the dog but leaves the leash behind, resulting in obvious artifacts. On the other hand, instruction-based editing methods Qwen-Image-Edit and Nanobanana Pro correctly capture the semantic intent described by the text. However, they often fail to match the desired motion extent or direction. For instance, NanoBanana Pro rotates the boy's head in the wrong direction in the 1st row and shrinks the flower too much in the 3rd row. In the 4th row, it also alters the dog's pose and layout, deviating from the original intent. By contrast, both versions of our TV-Edit produce edits that are semantically correct, spatially precise, and visually coherent. They can more accurately control the magnitude of the action while preserving the image fidelity. In the 4th row, even without explicit control over the leash, TV-Edit moves it consistently with the dog, yielding faithful results.

Fig. 6 demonstrates TV-Edit's flexibility on the two sub-tasks of fine-grained control. On the left, it accurately controls motion magnitudes under a fixed textual instruction (*e.g.*, rotating the dog's head

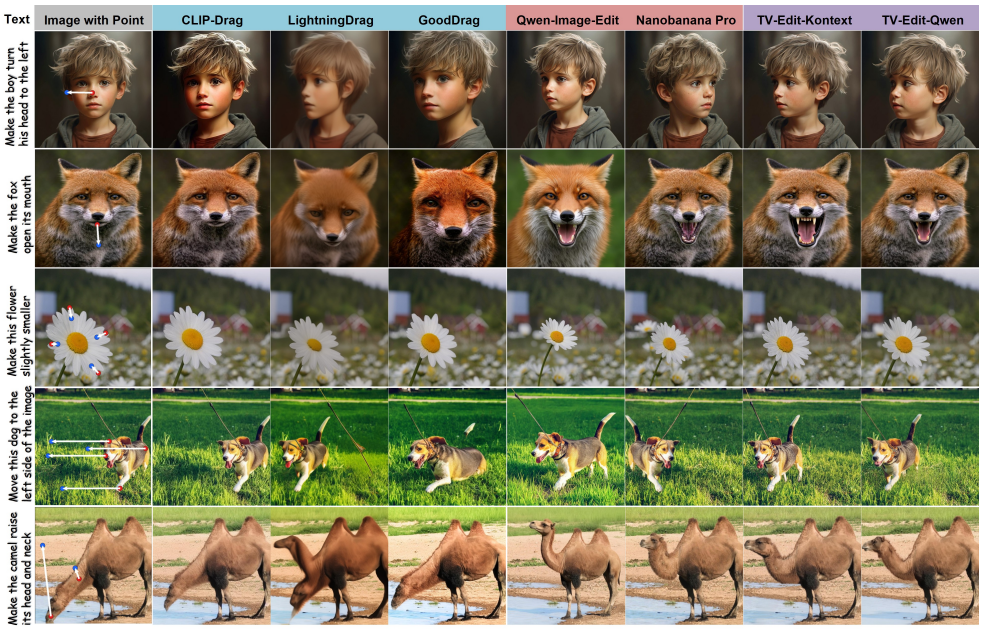

Figure 5: Qualitative results of competing methods on TV-Edit-Bench.

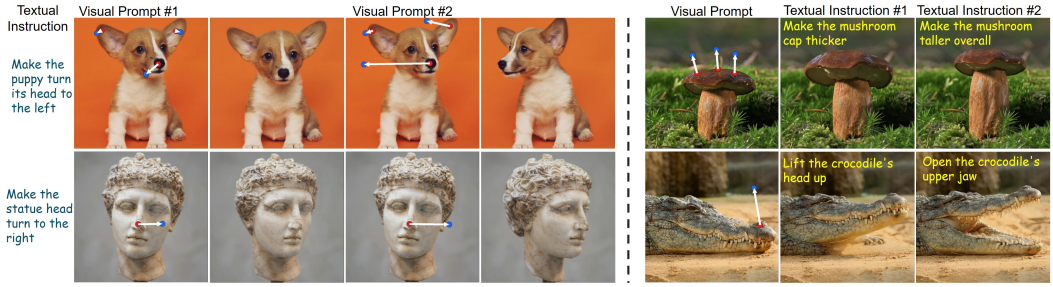

Figure 6: Editing results of TV-Edit on the two sub-tasks of fine-grained control. Left: motion magnitude variation task. Right: semantic variation task.

to different degrees). On the right, when visual prompts are ambiguous, it adapts to different texts to realize distinct actions (*e.g.*, lifting the head vs. opening the mouth for the same trajectory). These results show that TV-Edit can accurately realize user intent by leveraging textual and visual prompts. In **Appendix D**, we provide more qualitative comparisons.

4.4 Ablation Studies on TV-Edit

We conduct ablation studies to validate the architectural design of TV-Edit and our choice of hyper-parameters. All experiments are conducted using the Qwen-Image-Edit-based version of TV-Edit. Additional ablations on multimodal input strategies at inference time, and isolating the contribution of dataset from that of the architecture and training paradigm, are provided in **Appendix E**.

Ablation of Architecture. In Table 2, we present an ablation study on our proposed architecture. We establish a baseline that includes only a sparse point encoder and linear layers that inject features into the editing backbone. As shown in the first row, simply injecting sparse geometric conditions is insufficient for effective spatial control, resulting in a suboptimal geometric accuracy where MD_d only reaches 0.1355. As can be seen in the second row, incorporating the content-aware spatial controller yields substantial improvements in both semantic consistency and geometric accuracy. This indicates that rather than relying on sparse trajectories, processing the fused trajectory and image content through a transformer can yield highly effective control features for backbone injection. Subsequently, in the third row, we attempt to expand the output features of each block using multiple linear injectors before feeding them into the main branch. As observed in the third row, this modification only brings

Table 2: Ablation study on architecture.

Setting	$\text{DS}^{\text{tgt}}_{\text{global}}\uparrow$	$\text{DS}^{\text{tgt}}_{\text{local}}\uparrow$	$\text{MD}_{\text{d}}\downarrow$	PF↑
baseline	.8595	.9185	.1355	0.89
+ content aware spatial controller	.9024	.9430	.0505	0.87
+ multiple linear injectors	.9015	.9443	.0501	0.91
+ time-modulated injectors	**.9134**	**.9490**	**.0462**	**0.93**

Table 3: Ablation on noise sampling strategies.

Noise Dist	$\text{DS}^{\text{tgt}}_{\text{local}}\uparrow$	$\text{DS}^{\text{tgt}}_{\text{global}}\uparrow$	$\text{MD}_{\text{d}}\downarrow$	PF↑
$\text{Beta}(1,1)$	.9324	.8857	.0568	0.90
$\text{Beta}(3,2)$	.9379	.8966	.0510	0.90
$\text{Beta}(5,2)$	**.9490**	**.9134**	**.0462**	**0.93**
$\text{Beta}(10,2)$	.9335	.8861	.0604	0.88

Table 4: Ablation on the number of blocks.

Block Num	$\text{DS}^{\text{tgt}}_{\text{local}}\uparrow$	$\text{MD}_{\text{d}}\downarrow$	PF↑	# Para.
1	.9369	.0481	0.88	167M
3	.9401	.0549	0.91	337M
5	**.9490**	**.0462**	**0.93**	506M
15	.9400	.0468	0.90	1.40B

marginal gains. Despite the addition of extra linear layers intended to increase feature diversity, the network struggles to learn distinct representations without explicit guidance. As indicated in the fourth row, we introduce a time-modulated scaling mechanism, which substantially enhances the model's control capabilities. During the denoising process, it dynamically adjusts the influence of the control branch features on different backbone blocks over timesteps. This dynamic modulation effectively balances spatial control and detail generation, ultimately leading to editing results with high visual consistency.

Ablation of Different Timestep Sampling Strategies. Table 3 shows the effect of different timestep sampling strategies during training. Since TV-Editing mainly involves low-frequency structural changes, placing more emphasis on larger timesteps generally improves geometric accuracy. This trend is reflected by the clear gain under $\text{Beta}(5,2)$. However, when the sampling distribution becomes overly concentrated on high-noise steps, both geometric accuracy and prompt-following performance deteriorate. Although high-noise stages are crucial for structural control, later timesteps also make important contributions to the editing process.

Ablation of the Number of Controller Blocks. Table 4 shows the effect of the number of controller blocks. Even with only one block, the controller already achieves strong performance, reaching an MD_d of 0.0481 with only 167M trainable parameters, which demonstrates the efficiency of our architecture design. As the number of blocks increases, the prompt-following score further improves and reaches 0.93 at $N = 5$, indicating stronger semantic control. However, when N is increased to 15, the parameter count rises to 1.4B and the prompt-following score drops under the same training iteration. Considering both performance and efficiency, we use $N = 5$ in our TV-Edit.

5 Conclusion

We introduced **Text-Vision Co-Instructed Image Editing**, a new image editing task that jointly leverages textual instructions and sparse visual prompts to reduce the ambiguity of single-modality control. To tackle this task, we proposed **TV-Edit**, a plug-and-play framework with a decoupled Content-Aware Spatial Controller, and constructed a text-vision paired training dataset to learn unified semantic and spatial control. In addition, we curated **TV-Edit-Bench**, which contains carefully designed test cases and a comprehensive evaluation protocol. Extensive experiments on TV-Edit-Bench revealed the limitations of both visual prompt-only and instruction-only editing methods. By contrast, TV-Edit achieved much stronger semantic faithfulness, spatial controllability, and visual consistency, demonstrating its superiority over existing methods for robust image editing.

Limitations. Our TV-Edit has some limitations. First, it is built upon large-scale editing foundation models so that its inference speed prevents from real-time interactive editing. Second, it works well for 2D operations but remains limited for complex 3D manipulations such as out-of-plane rotations.

Acknowledgments and Disclosure of Funding

This work is supported by the PolyU-OPPO Joint Innovative Research Center.

References

[1] Qingyan Bai, Qiuyu Wang, Hao Ouyang, Yue Yu, Hanlin Wang, Wen Wang, Ka Leong Cheng, Shuailei Ma, Yanhong Zeng, Zichen Liu, et al. Scaling instruction-based video editing with a high-quality synthetic dataset. *arXiv preprint arXiv:2510.15742*, 2025.

[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] Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, Sumith Kulal, et al. Flux. 1 kontext: Flow matching for in-context image generation and editing in latent space. *arXiv e-prints*, pages arXiv–2506, 2025.

[4] 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.

[5] Mu Cai, Haotian Liu, Siva Karthik Mustikovela, Gregory P Meyer, Yuning Chai, Dennis Park, and Yong Jae Lee. Vip-llava: Making large multimodal models understand arbitrary visual prompts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12914–12923, 2024.

[6] Di Chang, Mingdeng Cao, Yichun Shi, Bo Liu, Shengqu Cai, Shijie Zhou, Weilin Huang, Gordon Wetzstein, Mohammad Soleymani, and Peng Wang. Bytemorph: Benchmarking instruction-guided image editing with non-rigid motions. *arXiv preprint arXiv:2506.03107*, 2025.

[7] DuoSheng Chen, Binghui Chen, Yifeng Geng, and Liefeng Bo. Adaptivedrag: Semantic-driven dragging on diffusion-based image editing. *arXiv preprint arXiv:2410.12696*, 2024.

[8] Google DeepMind. Nano banana pro (gemini 3 pro image), 2025. Google Cloud Blog, November 2025. Accessed: 2026-04-03.

[9] Stephanie Fu, Netanel Tamir, Shobhita Sundaram, Lucy Chai, Richard Zhang, Tali Dekel, and Phillip Isola. Dreamsim: Learning new dimensions of human visual similarity using synthetic data. *Advances in Neural Information Processing Systems*, 36, 2024.

[10] Huiguo He, Pengyu Yan, Ziqi Yi, Weizhi Zhong, Zheng Liu, Yejun Tang, Huan Yang, Kun Gai, Guanbin Li, and Lianwen Jin. Contextdrag: Precise drag-based image editing via context-preserving token injection and position-consistent attention. *arXiv preprint arXiv:2512.08477*, 2025.

[11] Xingzhong Hou, Boxiao Liu, Yi Zhang, Jihao Liu, Yu Liu, and Haihang You. Easydrag: Efficient point-based manipulation on diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8404–8413, 2024.

[12] Ziqi Jiang, Zhen Wang, and Long Chen. Clipdrag: Combining text-based and drag-based instructions for image editing. *arXiv preprint arXiv:2410.03097*, 2024.

[13] Xuan Ju, Ailing Zeng, Yuxuan Bian, Shaoteng Liu, and Qiang Xu. Direct inversion: Boosting diffusion-based editing with 3 lines of code. *arXiv preprint arXiv:2310.01506*, 2023.

[14] Nikita Karaev, Yuri Makarov, Jianyuan Wang, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker3: Simpler and better point tracking by pseudo-labelling real videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6013–6022, 2025.

[15] 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.

[16] Vladimir Kulikov, Matan Kleiner, Inbar Huberman-Spiegelglas, and Tomer Michaeli. Flowedit: Inversion-free text-based editing using pre-trained flow models. *arXiv preprint arXiv:2412.08629*, 2024.

[17] Black Forest Labs. Official weights of FLUX.1 dev. `https://huggingface.co/black-forest-labs/FLUX.1-dev`, 2024. Accessed: 2024-11-14.

[18] 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.

[19] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26296–26306, 2024.

[20] Haofeng Liu, Chenshu Xu, Yifei Yang, Lihua Zeng, and Shengfeng He. Drag your noise: Interactive point-based editing via diffusion semantic propagation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6743–6752, 2024.

[21] Shiyu Liu, Yucheng Han, Peng Xing, Fukun Yin, Rui Wang, Wei Cheng, Jiaqi Liao, Yingming Wang, Honghao Fu, Chunrui Han, et al. Step1x-edit: A practical framework for general image editing. *arXiv preprint arXiv:2504.17761*, 2025.

[22] Xingchao Liu, Chengyue Gong, and qiang liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023.

[23] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[24] Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Null-text inversion for editing real images using guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6038–6047, 2023.

[25] Chong Mou, Xintao Wang, Jiechong Song, Ying Shan, and Jian Zhang. Dragondiffusion: Enabling drag-style manipulation on diffusion models. *arXiv preprint arXiv:2307.02421*, 2023.

[26] Xingang Pan, Ayush Tewari, Thomas Leimkühler, Lingjie Liu, Abhimitra Meka, and Christian Theobalt. Drag your gan: Interactive point-based manipulation on the generative image manifold. In *ACM SIGGRAPH 2023 conference proceedings*, pages 1–11, 2023.

[27] Yuang Peng, Yuxin Cui, Haomiao Tang, Zekun Qi, Runpei Dong, Jing Bai, Chunrui Han, Zheng Ge, Xiangyu Zhang, and Shu-Tao Xia. Dreambench++: A human-aligned benchmark for personalized image generation. *arXiv preprint arXiv:2406.16855*, 2024.

[28] Xinyu Pu, Hongsong Wang, Jie Gui, and Pan Zhou. Dragging with geometry: From pixels to geometry-guided image editing. *arXiv preprint arXiv:2509.25740*, 2025.

[29] 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.

[30] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

[31] 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.

[32] Amadou S Sangare, Adrien Maglo, Mohamed Chaouch, and Bertrand Luvison. Improving controllable generation: Faster training and better performance via x_0-supervision. *arXiv preprint arXiv:2604.05761*, 2026.

[33] Yujun Shi, Jun Hao Liew, Hanshu Yan, Vincent YF Tan, and Jiashi Feng. Lightningdrag: Lightning fast and accurate drag-based image editing emerging from videos. *arXiv preprint arXiv:2405.13722*, 2024.

[34] Aleksandar Shtedritski, Christian Rupprecht, and Andrea Vedaldi. What does clip know about a red circle? visual prompt engineering for vlms. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11987–11997, 2023.

[35] Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

[36] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

[37] Meituan LongCat Team, Hanghang Ma, Haoxian Tan, Jiale Huang, Junqiang Wu, Jun-Yan He, Lishuai Gao, Songlin Xiao, Xiaoming Wei, Xiaoqi Ma, et al. Longcat-image technical report. *arXiv preprint arXiv:2512.07584*, 2025.

[38] OpenAI Team. Gpt-4o system card, 2024.

[39] Team Wan. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

[40] Yixin Wan, Lei Ke, Wenhao Yu, Kai-Wei Chang, and Dong Yu. Motionedit: Benchmarking and learning motion-centric image editing. *arXiv preprint arXiv:2512.10284*, 2025.

[41] Qihang Wang, Yaxiong Wang, Lechao Cheng, and Zhun Zhong. Tdedit: A unified diffusion framework for text-drag guided image manipulation. *arXiv preprint arXiv:2509.21905*, 2025.

[42] Yihan Wang, Lahav Lipson, and Jia Deng. Sea-raft: Simple, efficient, accurate raft for optical flow. In *European Conference on Computer Vision*, pages 36–54. Springer, 2024.

[43] 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.

[44] Siwei Xia, Li Sun, Tiantian Sun, and Qingli Li. Draglora: Online optimization of lora adapters for drag-based image editing in diffusion model. *arXiv preprint arXiv:2505.12427*, 2025.

[45] Chenxi Xie, Minghan Li, Shuai Li, Yuhui Wu, Qiaosi Yi, and Lei Zhang. Dnaedit: Direct noise alignment for text-guided rectified flow editing. *arXiv preprint arXiv:2506.01430*, 2025.

[46] Zhucun Xue, Jiangning Zhang, Teng Hu, Haoyang He, Yinan Chen, Yuxuan Cai, Yabiao Wang, Chengjie Wang, Yong Liu, Xiangtai Li, et al. Ultravideo: High-quality uhd video dataset with comprehensive captions. *arXiv preprint arXiv:2506.13691*, 2025.

[47] Yang Ye, Xianyi He, Zongjian Li, Bin Lin, Shenghai Yuan, Zhiyuan Yan, Bohan Hou, and Li Yuan. Imgedit: A unified image editing dataset and benchmark. *arXiv preprint arXiv:2505.20275*, 2025.

[48] Shukang Yin, Chaoyou Fu, Sirui Zhao, Tong Xu, Hao Wang, Dianbo Sui, Yunhang Shen, Ke Li, Xing Sun, and Enhong Chen. Woodpecker: Hallucination correction for multimodal large language models. *Science China Information Sciences*, 67(12):220105, 2024.

[49] Qifan Yu, Wei Chow, Zhongqi Yue, Kaihang Pan, Yang Wu, Xiaoyang Wan, Juncheng Li, Siliang Tang, Hanwang Zhang, and Yueting Zhuang. Anyedit: Mastering unified high-quality image editing for any idea. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 26125–26135, 2025.

[50] Ahmad Zafarani, Zahra Dehghanian, Mohammadreza Davoodi, Mohsen Shadroo, MohammadAmin Fazli, and Hamid R Rabiee. Realdrag: The first dragging benchmark with real target image. *arXiv preprint arXiv:2512.12287*, 2025.

[51] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3836–3847, 2023.

[52] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

[53] Yabo Zhang, Xinpeng Zhou, Yihan Zeng, Hang Xu, Hui Li, and Wangmeng Zuo. Framepainter: Endowing interactive image editing with video diffusion priors. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 18121–18131, 2025.

[54] Zewei Zhang, Huan Liu, Jun Chen, and Xiangyu Xu. Gooddrag: Towards good practices for drag editing with diffusion models. *arXiv preprint arXiv:2404.07206*, 2024.

[55] Zihan Zhou, Shilin Lu, Shuli Leng, Shaocong Zhang, Zhuming Lian, Xinlei Yu, and Adams Wai-Kin Kong. Dragflow: Unleashing dit priors with region based supervision for drag editing. *arXiv preprint arXiv:2510.02253*, 2025.

Appendix

In this appendix, we provide the following materials:

A More details of the TV-Edit-23K dataset (referring to Sec. 3.2 in the main paper).

B More analysis of training strategy (referring to Sec. 3.3 in the main paper).

C More details of TV-Edit-Bench, including the dataset construction and evaluation protocol (referring to Sec. 4.2 in the main paper).

D More editing results of TV-Edit and more visual comparisons on TV-Edit-Bench (referring to Sec. 4.3 in the main paper).

E More ablation studies on TV-Edit (referring to Sec. 4.4 in the main paper).

F Quantitative and qualitative comparison with drag-based methods on Drag-Bench.

G User study.

H Failure cases and analysis.

I Potential social impact.

A More Details of TV-Edit-23K Dataset

A.1 Detailed Prompts for Paired Textual Annotation

In the paired textual annotation stage, we instruct Qwen-3-VL [2] to provide the action that can transform one image to the other. Detailed prompts for this transformation are shown in Fig. A1.

#Task:
Two images are provided, in which some objects have moved or changed. Please provide a pair of instructions — for example, "make the dog move to the left" and "make the dog move to the right" — that simply describe the actions needed to transform one image into the other.

#Guidelines:
- Two images are provided, in which some objects have moved or changed.
- There are also points marked on the images showing the motion trajectories of the tracked objects.
- Note: These points are only used to help identify what object should be focused, **DO NOT** describe the points.
- **Do not include any description about the marked color points we provide.**
- Format your output as a strings without using the newline character.

#Example output:
```
{
  "instruct1": "make the dog's mouth more open",
  "instruct2": "make the dog's mouth more closed"
}
```

Figure A1: Detailed prompts for paired textual annotation in TV-Edit-23K data construction pipeline.

A.2 Samples in TV-Edit-23K Dataset

In Fig. A2, we show some training samples in TV-Edit-23K. We see that our data construction pipeline generates training data with dense and accurate point pairs, enabling the model to learn the corresponding geometric relationships. Furthermore, it can be seen that MLLM provides accurate semantic transformation instructions which match the motion of labeled point pairs.

B More Analysis of TV-Edit Training Strategy

Since TV-Edit aims to achieve precise spatial control and geometric accuracy guided by sparse points, prioritizing the optimization of the global spatial layout is crucial. In flow matching models, the generative process at large timesteps (the high-noise regime, where $t \to 1$) is primarily responsible for establishing this global spatial layout and low-frequency structures. Conversely, small timesteps (the low-noise regime, where $t \to 0$) focus on high-frequency texture details. Therefore, the fundamental motivation behind our training strategy is to force the network to focus its learning capacity on

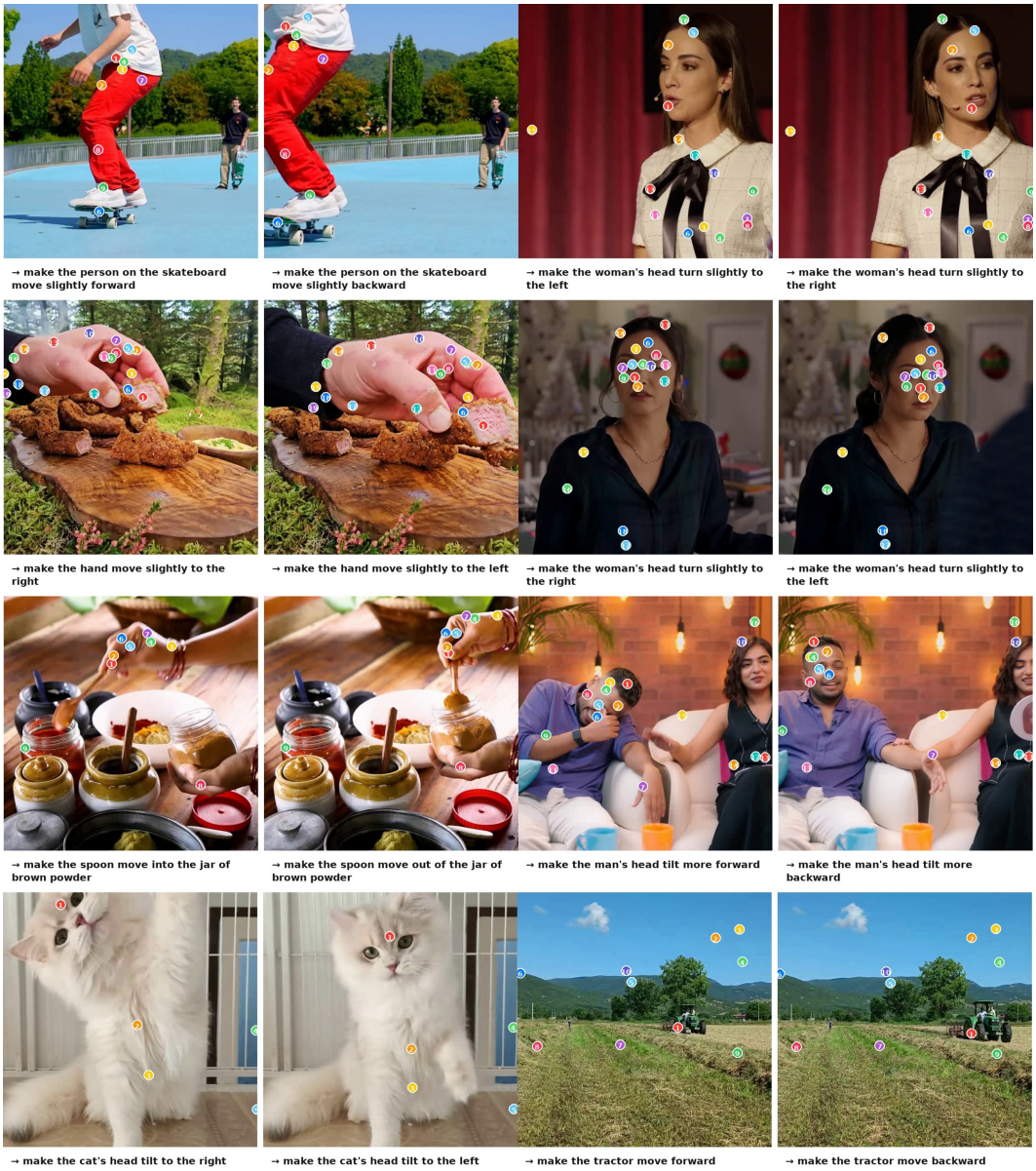

Figure A2: Samples in TV-Edit-23K Dataset. The text below the image represents the instructions to transform the image into another image.

the high-noise regime. We achieve this through two complementary approaches: an implicit loss weighting mechanism and an explicit time-step sampling strategy.

V-supervision flow matching loss. In the standard Rectified Flow, the forward process constructs a linear interpolation between the clean data latent $\mathbf{Z}_0$ and the sampled Gaussian noise $\mathbf{Z}_1$. The intermediate state $\mathbf{Z}_t$ at timestep $t \in [0, 1]$ is defined as:

$$\mathbf{Z}_t = (1 - t)\mathbf{Z}_0 + t\mathbf{Z}_1. \tag{A1}$$

The ground-truth velocity field that drives $\mathbf{Z}_0$ to $\mathbf{Z}_1$ is simply the derivative of $\mathbf{Z}_t$ with respect to t:

$$v_t = \frac{\mathrm{d}\mathbf{Z}_t}{\mathrm{d}t} = \mathbf{Z}_1 - \mathbf{Z}_0. \tag{A2}$$

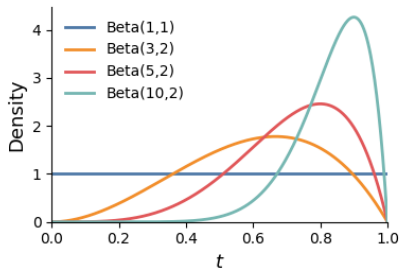

Figure A3: Illustration of different Beta distribution.

Typically, the model v_θ is trained to predict this constant velocity using the standard v-prediction objective, which uniformly weights all timesteps:

$$\mathcal{L}_v = \mathbb{E}_{t,\mathbf{Z}_0,\mathbf{Z}_1}\left[\|v_\theta(\mathbf{Z}_t) - (\mathbf{Z}_1 - \mathbf{Z}_0)\|_2^2\right]. \tag{A3}$$

Derivation of the $\mathbf{Z}_0$-supervision flow matching loss. Instead of directly supervise the velocity, TV-Edit adopts a $\mathbf{Z}_0$-supervision loss. Based on the forward process definition, we can express the ground-truth $\mathbf{Z}_0$ in terms of $\mathbf{Z}_t$ and the ground-truth velocity v_t. By substituting $\mathbf{Z}_1 = \mathbf{Z}_0 + v_t$ into the $\mathbf{Z}_t$ equation, we get:

$$\mathbf{Z}_t = (1-t)\mathbf{Z}_0 + t(\mathbf{Z}_0 + v_t) = \mathbf{Z}_0 + tv_t. \tag{A4}$$

Thus, the ground-truth data latent can be written as:

$$\mathbf{Z}_0 = \mathbf{Z}_t - tv_t. \tag{A5}$$

During training, our model predicts the velocity $v_\theta(\mathbf{Z}_t, \dots)$, which is then used to estimate the clean latent $\hat{\mathbf{Z}}_0$:

$$\hat{\mathbf{Z}}_0 = \mathbf{Z}_t - t \cdot v_\theta(\mathbf{Z}_t) \tag{A6}$$

Our training objective $\mathcal{L}_{\text{fm}}$ minimizes the Mean Squared Error (MSE) between the estimated $\hat{\mathbf{Z}}_0$ and the ground-truth $\mathbf{Z}_0$. By substituting the expressions for $\hat{\mathbf{Z}}_0$ and $\mathbf{Z}_0$, we can reveal its relationship with the standard v-supervision flow matching loss:

$$\begin{aligned}
\mathcal{L}_{\text{fm}} &= \mathbb{E}_{t,\mathbf{Z}_0,\mathbf{Z}_1}\left[\left\|\hat{\mathbf{Z}}_0 - \mathbf{Z}_0\right\|_2^2\right] \\
&= \mathbb{E}_{t,\mathbf{Z}_0,\mathbf{Z}_1}\left[\|(\mathbf{Z}_t - t \cdot v_\theta(\mathbf{Z}_t)) - (\mathbf{Z}_t - t \cdot v_t)\|_2^2\right] \\
&= \mathbb{E}_{t,\mathbf{Z}_0,\mathbf{Z}_1}\left[\|-t \cdot v_\theta(\mathbf{Z}_t) + t \cdot v_t\|_2^2\right] \\
&= \mathbb{E}_{t,\mathbf{Z}_0,\mathbf{Z}_1}\left[t^2 \|v_\theta(\mathbf{Z}_t) - (\mathbf{Z}_1 - \mathbf{Z}_0)\|_2^2\right].
\end{aligned} \tag{A7}$$

This demonstrates that our $\mathbf{Z}_0$-supervision is mathematically equivalent to the v-supervision loss scaled by a coefficient t^2. This implicit t^2 mechanism intentionally assigns significantly larger penalty weights to errors made at large t. The effectiveness of this loss weighting strategy has also been corroborated by recent studies in controllable generation [32].

Timestep sampling strategy. While the $\mathbf{Z}_0$-supervision implicitly weights the loss, we further explicitly bias the training distribution to sample more large timesteps. As illustrated in Fig. A3, we plot the probability density functions of the Beta distribution under various α and β hyperparameters. For Beta$(10, 2)$, the distribution is heavily concentrated around 0.9, which corresponds to extremely high noise levels where spatial layout is determined. Conversely, the Beta$(3, 2)$ distribution is only slightly skewed towards larger timesteps, and empirical training under this setting is inferior to Beta$(5, 2)$. As evidenced by the ablation study on sampling distributions presented in this appendix, the Beta$(5, 2)$ strategy yields the optimal performance for TV-Edit, as it primarily focuses on the high-noise regime while maintaining adequate coverage of low-noise timesteps.



Figure A4: More samples in TV-Edit-Bench.

Table A1: Comparison of drag-based editing datasets.

Dataset	#Train/ #Test	Target Image	Instruction	Description	Dense Annotation	Controlled Variation
DragBench [25]	0 / 205	✗	✗	✓	✗	✗
ReDBench [55]	0/120	✗	✗	✓	✗	✗
LightningDrag [33]	220k / 0	✓	✗	✓	✗	✗
FramePainter [53]	22k / 200	✓	✗	✓	✗	✗
RealDrag [50]	0 / 415	✓	✓	✓	✗	✗
Ours	23k / 120	✓	✓	✓	✓	✓

C More Details of TV-Edit-Bench

C.1 Samples in TV-Edit-Bench

As illustrated in Fig. A4, we present additional samples from our TV-Edit-Bench dataset. Each pair consists of a source image (left) and a reference target image (right), which maintain high visual consistency owing to our rigorous generation pipeline and manual filtering. The annotated points indicate spatial correspondences between the two images. Specifically, the larger points serve as visual prompts for the editing process and are utilized to compute the sparse MD ($\mathrm{MD_s}$) during evaluation. Recognizing that a natural spatial edit typically involves the holistic transformation of a local region, we sample additional surrounding points to track and evaluate the dense MD ($\mathrm{MD_d}$) between the edited results and the targets.

Furthermore, Table A1 compares our established dataset with existing drag-based datasets, highlighting our unique advantages. Our dataset provides comprehensive splits for both training and evaluation, with each image pair accompanied by a specific motion-centric instruction. Notably, TV-Edit-Bench is the first to introduce dense annotations, which are crucial for evaluating the consistency of regional transformations. Additionally, we incorporate test pairs with controlled variations, including both magnitude and semantic variation cases, to comprehensively assess the spatial accuracy and semantic flexibility of various editing methods.

C.2 Evaluation Protocol

Details of DINOv3-based evaluation metrics. We evaluate editing quality using features from DINOv3 [35]. We denote by $f_{\text{cls}}(I)$ and $\{f_i^{\text{patch}}(I)\}_{i=1}^{N}$ the CLS token and the set of N patch tokens extracted from image I, respectively.

- *Global DINO Score.* The overall semantic consistency between the target image I_{tgt} and the edited image I_{edit} is measured via the cosine similarity of their CLS tokens:

$$DS_{\text{global}} = \cos\big(f_{\text{cls}}(I_{\text{tgt}}),\ f_{\text{cls}}(I_{\text{edit}})\big). \tag{A8}$$

- *Local DINO Score.* To assess the fine-grained fidelity within edited regions, we employ a nearest-neighbour patch matching scheme. Given a reference image I_{ref} (either source or target) and a corresponding binary mask M, we first downsample M to the patch grid via average pooling to obtain the set of masked patch indices $\mathcal{M}$. For each masked reference patch $i \in \mathcal{M}$, we find its nearest neighbour in the edited image:

$$j^*(i) = \underset{j\in\{1,\dots,N\}}{\arg\max}\ \cos\big(\hat{f}_i^{\text{patch}}(I_{\text{ref}}),\ \hat{f}_j^{\text{patch}}(I_{\text{edit}})\big), \tag{A9}$$

where $\hat{f} = f/\|f\|$ means ℓ_2-normalization. The local score is the average similarity over all masked patches:

$$DS_{\text{local}} = \frac{1}{|\mathcal{M}|}\sum_{i\in\mathcal{M}} \cos\big(\hat{f}_i^{\text{patch}}(I_{\text{ref}}),\ \hat{f}_{j^*(i)}^{\text{patch}}(I_{\text{edit}})\big). \tag{A10}$$

By computing similarity at the patch level, this approach effectively decouples content fidelity from spatial positioning. Consequently, it mitigates the evaluation inaccuracies that arise when using strictly pixel-aligned metrics such as LPIPS [52].

Details of MLLM-based evaluation metrics. To comprehensively evaluate our model, we adopt the automatic evaluation protocol introduced in ContextDrag [10] based on DreamBench++ [27]. Specifically, we leverage an MLLM (Qwen-3-VL [2]) as the evaluator to assess the generated images across two dimensions: prompt following (PF) and concept preservation (CP). We modify the original prompts from ContextDrag by incorporating reference ground truth from our benchmark dataset as an evaluation criterion. This addition provides clearer guidance, making it easier for the MLLM to understand the evaluation task.

- *Concept Preservation.* CP employs an MLLM for a comprehensive evaluation of image fidelity, surpassing traditional pixel- or feature-level metrics by assessing the semantic consistency of unedited regions. Fig. A5 shows the evaluation instructions for MLLM. Prompted with the **source image**, **reference GT**, **edited result**, and **textual instruction**, the MLLM utilizes self-aligned reasoning to understand and plan the evaluation process. Finally, it assigns a score from 0 to 4 for each case. The final score for a method is obtained by averaging and normalizing these case-level ratings.

- *Prompt Following.* PF evaluates whether the edited result adheres to the user-provided textual instruction and successfully achieves the intended semantic transformation. Compared to the evaluation prompts used in ContextDrag [10], our TV-Edit-Bench introduces a key improvement: rather than requiring the MLLM to infer semantic changes from visual prompts, we directly supply the explicit semantic transformations. This modification yields significantly more reliable evaluation results. As illustrated in Fig. A6, we provide the MLLM with the source image, reference ground truth, edited result, and textual instruction, prompting it to rate the editing correctness ranging from 0 to 4. Specifically, the MLLM is instructed to focus on four key aspects, including operation type, target object, spatial accuracy, and magnitude. After self-aligned reasoning, the model formulates a 6-step evaluation plan to assign the score. Finally, similar to the previous metric, the case-level scores are averaged and normalized to produce the final performance score.

D More Editing Results of TV-Edit

D.1 Visual Comparisons on TV-Edit-Bench

As shown in Fig. A7, we provide more visual comparisons on TV-Edit-Bench. TV-Edit is capable of performing a variety of spatial transformations, including rotation and translation, demonstrating

Evaluation Instructions

[Task description] As an experienced image editing evaluator, your task is to assess the "Concept Preservation (CP)" between three given images. Concept Preservation measures whether the edited image faithfully retains the semantic objects, structural relationships, colors, textures, and overall style of the reference image.

[CP Scoring Criteria] Compare the Result Image against the Source Image and Reference GT across the following four dimensions:
1. Global Visual Quality: Assess the overall structural integrity of the entire image. Are there any noticeable artifacts, blurriness, noise, or unnatural distortions introduced during the generation process?
2. Semantics: Check if all elements (both edited and unedited) maintain their original semantic meaning. The structural relationships between objects and the background should remain logical and consistent.
3. Identity: Verify the specific identity of ALL subjects. If an object was edited or moved, does it remain the EXACT same entity (e.g., the same specific person or animal)? Do the unedited objects retain their original identities?
4. Color & Texture: Examine the lighting, hues, and surface details across the whole image. Are the original color schemes, shadows, and fine textures (like fabric, fur, skin, or background materials) perfectly maintained?

[[Scoring Range] Assign an integer score from 0 to 4 based on the following scale:
0 - Very Poor: Completely deviated from the Source and Reference GT. The image suffers from severe artifacts, structural collapse, or total loss of original concepts across both the subject and background.
1 - Poor: Noticeable artifacts are present. The overall image structure is roughly similar, but there is a massive loss of original content, semantics, or identity (e.g., the edited object is severely distorted, or the background is ruined).
2 - Fair: General semantics are roughly maintained, but the specific identity and fine details are not preserved (e.g., the edited object looks like a different instance of the same category, or background details are lost).
3 - Good: The identity of subjects/objects (both edited and unedited) is recognizable and preserved, but minor details (such as fine textures, slight color shifts, or subtle lighting changes) are not perfectly maintained.
4 - Excellent: Flawless preservation. The semantics, color, texture, identity, and global visual quality of the ENTIRE image are virtually identical to the Source and Reference GT.

[Input Format] Each time, you will receive 3 images and an Instruction:
- Image 1 (Source Image): The original image before editing.
- Image 2 (Reference GT): The ground truth image, serving as the perfect reference for concept preservation.
- Image 3 (Result Image): The AI-edited image to be evaluated.
- Instruction: The text prompt describing the intended operation on the Source Image.

[Output Format]
[internal thinking] (Briefly describe how you compared the three images step-by-step across the 4 dimensions: Visual Quality, Semantics, Identity, and Color/Texture for the ENTIRE image.)
[Score]Concept Preservation (CP): <an integer from 0 to 4>

Summary / Plan

[Summary]
Concept Preservation (CP) measures how faithfully an edited image retains the original semantic content, visual integrity, and stylistic consistency of the source image, as compared to a ground truth reference. It focuses on whether the edited object remains the same entity (identity), whether the relationships between objects and the background are preserved (semantics), and whether the overall look—colors, textures, lighting, and structural quality—remains unchanged. Unlike "Prompt Following," which evaluates whether the edit matches the user's instruction (e.g., "move the cat to the left"), CP evaluates whether the resulting image looks like a natural continuation of the original, with no unintended distortions or losses in detail, even if the edit was performed correctly.

[Plan]
1. **Global Visual Quality**: First, assess the overall clarity, sharpness, and presence of artifacts (e.g., blurring, noise, warping, or unnatural edges) across the entire Result Image. Compare it to the Source and Reference GT to check for structural integrity and realism.
2. **Semantics**: Next, examine the logical relationships between objects and the background. Are the scene elements coherent? For example, if a person was moved, does the background still make sense in context? Are lighting and perspective consistent?
3. **Identity**: Then, verify the specific identity of all subjects—both edited and unedited. Does the edited object look exactly like the original (same person, animal, or object)? Are unedited objects unchanged in appearance?
4. **Color & Texture**: Finally, analyze color fidelity, lighting consistency, and surface details (e.g., skin, fabric, fur, background materials). Are the hues, shadows, and textures preserved across the entire image?

[Restated Scoring Criteria]
- 0 (**Very Poor**): Severe deviations; major structural collapse, identity loss, or artifacts. The image is unrecognizable compared to the reference.
- 1 (**Poor**): Noticeable artifacts; significant loss of content or semantics. The image is roughly similar but flawed in key areas (e.g., distorted object, ruined background).
- 2 (**Fair**): General semantics preserved, but identity and fine details are lost (e.g., edited object looks like a different instance, textures are degraded).
- 3 (**Good**): Identity and semantics are preserved, but minor texture, color, or lighting inconsistencies exist.
- 4 (**Excellent**): Flawless preservation—identical semantics, identity, color, texture, and visual quality across the entire image.

Figure A5: Illustration of the Concept Preservation (CP) Evaluation Instruction and the corresponding Summary / Planning returned by Qwen 3-VL.

superior image fidelity and geometric accuracy compared to other methods. Furthermore, it is evident that TV-Edit is highly robust to edits across a wide range of magnitudes and affected areas. For instance, as shown in the last row, moving the person to the right side of the image involves a substantially large transformation magnitude and spatial extent. Despite this, TV-Edit still yields high-quality editing results, whereas other baseline methods suffer from severe artifacts.

Evaluation Instructions

[Task description] As an experienced image editing evaluator, your task is to assess the "Prompt Following (PF)" between three given images and an editing instruction. Prompt Following measures whether the AI-edited image correctly and accurately executes the editing operation described in the text instruction.

[PF Scoring Criteria] Compare the Result Image against the Source Image, Reference GT, and the Instruction across the following four dimensions:
1. Operation Type: Was the correct type of editing operation performed? For example, if the instruction says "drag the cat to the right", was a spatial translation performed (rather than deletion, color change, or no edit at all)?
2. Target Object: Was the correct object or region identified and edited? If the instruction specifies editing a particular element (e.g., "move the red ball"), did the model operate on exactly that element, and not a different one?
3. Spatial Accuracy: Was the edit applied in the correct direction and to the correct location? Compare the edited object's position, pose, or shape in the Result Image against the Reference GT. Is the spatial transformation consistent with what the instruction describes?
4. Magnitude & Completeness: Is the degree/extent of the edit appropriate? Was the operation completed fully, or was it only partially applied? Compare the magnitude of change in the Result Image to that in the Reference GT.

[Scoring Range] Assign an integer score from 0 to 4 based on the following scale:
0 - Very Poor: The instruction was completely ignored. No visible editing was performed, or the edit is entirely wrong (e.g., wrong object, wrong operation type, or the image is unchanged/corrupted).
1 - Poor: Some attempt at editing is visible, but it fundamentally fails to follow the instruction. For example: the wrong object was edited, the edit goes in the completely wrong direction, or the type of operation is incorrect.
2 - Fair: The general type of edit is correct and the right object was targeted, but the specifics are significantly off. For example: the object was moved but to a noticeably wrong position, or the magnitude of change is far from the Reference GT.
3 - Good: The instruction is largely followed correctly. The right object was edited with the right operation in roughly the right direction and magnitude, but there are minor deviations in precision compared to the Reference GT (e.g., slightly off-target position, slightly under/over-applied).
4 - Excellent: The instruction is perfectly followed. The edit matches the Reference GT closely in terms of which object was changed, what operation was applied, the direction, position, and magnitude of the edit. The result is virtually indistinguishable from the Reference GT in terms of the intended editing operation.

[Input Format] Each time, you will receive 3 images and an Instruction:
- Image 1 (Source Image): The original image before editing.
- Image 2 (Reference GT): The ground truth image, serving as the perfect reference for what the correct edit should look like.
- Image 3 (Result Image): The AI-edited image to be evaluated.
- Instruction: The text prompt describing the intended operation on the Source Image.

[Output Format]
[internal thinking] (Briefly describe how you evaluated the edit step-by-step across the 4 dimensions: Operation Type, Target Object, Spatial Accuracy, and Magnitude & Completeness.)
[Score] Prompt Following (PF): <an integer from 0 to 4>

Summary / Plan

[Summary]
Prompt Following (PF) measures how accurately an AI-generated image executes a specific editing instruction, focusing solely on the correctness of the **intended edit** rather than the preservation of unedited content. It evaluates whether the AI correctly identified the target object, applied the right type of operation (e.g., dragging, resizing), moved it in the correct direction and to the correct location, and applied the edit with the appropriate magnitude. Unlike Concept Preservation, which assesses whether the original scene's semantics remain intact, PF is concerned only with the fidelity of the **action** described in the instruction—regardless of how well other parts of the image are preserved.

[Plan]
1. **Analyze the Instruction** – Identify the key elements: what object is to be edited, what operation is requested (e.g., drag, move, resize), and the direction/magnitude of change (e.g., "to the right," "by 10 pixels").
2. **Compare Source and Reference GT** – Determine exactly what changed from the source to the reference. This defines the correct edit: which object moved, how far, in what direction, and whether the operation type matches.
3. **Evaluate Operation Type** – Check if the Result Image shows the same kind of edit (e.g., spatial translation) as the Reference GT, or if it shows a different operation (e.g., deletion, color change, or no edit).
4. **Assess Target Object** – Confirm that the same object was edited in the Result Image as in the Reference GT. Is it the correct object, or was a different one altered?
5. **Evaluate Spatial Accuracy** – Compare the position, orientation, or shape of the edited object in the Result Image to the Reference GT. Is it in the right place, facing the right way, and aligned properly?
6. **Assess Magnitude & Completeness** – Determine whether the edit is fully applied and at the correct scale. Is it slightly off, or nearly complete? Does it match the Reference GT in extent?
7. **Assign Score** – Use the four dimensions to determine the final PF score based on the provided criteria.

[Restated Scoring Criteria]
- 0 (**Very Poor**): No edit performed or completely wrong (wrong object, wrong operation, or image unchanged/corrupted).
- 1 (**Poor**): Edit attempted but fundamentally incorrect—wrong object, wrong direction, or wrong operation type.
- 2 (**Fair**): Correct operation type and object, but significant errors in position, direction, or magnitude (e.g., moved too far, in wrong direction).
- 3 (**Good**): Mostly correct—right object, right operation, right direction and magnitude—but minor inaccuracies (e.g., slight misalignment, slight under/over-application).
- 4 (**Excellent**): Perfect execution—matches Reference GT exactly in object, operation, direction, and magnitude; indistinguishable from correct edit.

Figure A6: Illustration of the Prompt Following (PF) Evaluation Instruction and the corresponding Summary / Planning returned by Qwen 3-VL.

D.2 Results on Simultaneous Spatial Control and Semantic Editing

Our method is capable of achieving precise spatial control while simultaneously accommodating additional semantic edits. As illustrated in Figure A8, by providing the instruction "change it to a tiger" alongside the motion constraints, our model successfully rotates the dog's head while transforming its identity into a tiger. Similarly, as shown in the second and third rows, our approach allows for



Figure A7: Visual comparison on TV-Edit-Bench. Best viewed by zoom-in.

complex semantic modifications—such as adding objects or altering colors—while maintaining precise control over the magnitude of spatial transformations.

E More Ablation Studies on TV-Edit

Ablation of textual and visual inputs at inference time. Table A2 presents the effect of differ-

Table A2: Inference-time ablation of textual and visual inputs.

Exp.	Setting	$MD_d \downarrow$	PF ↑
1	Qwen baseline	.1380	.86
2	Trajectory + text (w/o controller text)	.0502	.91
3	Trajectory only (empty text)	.0627	.83
4	Text only (w/o trajectory)	.1410	.87
5	Trajectory + text (full model)	**.0462**	**.93**

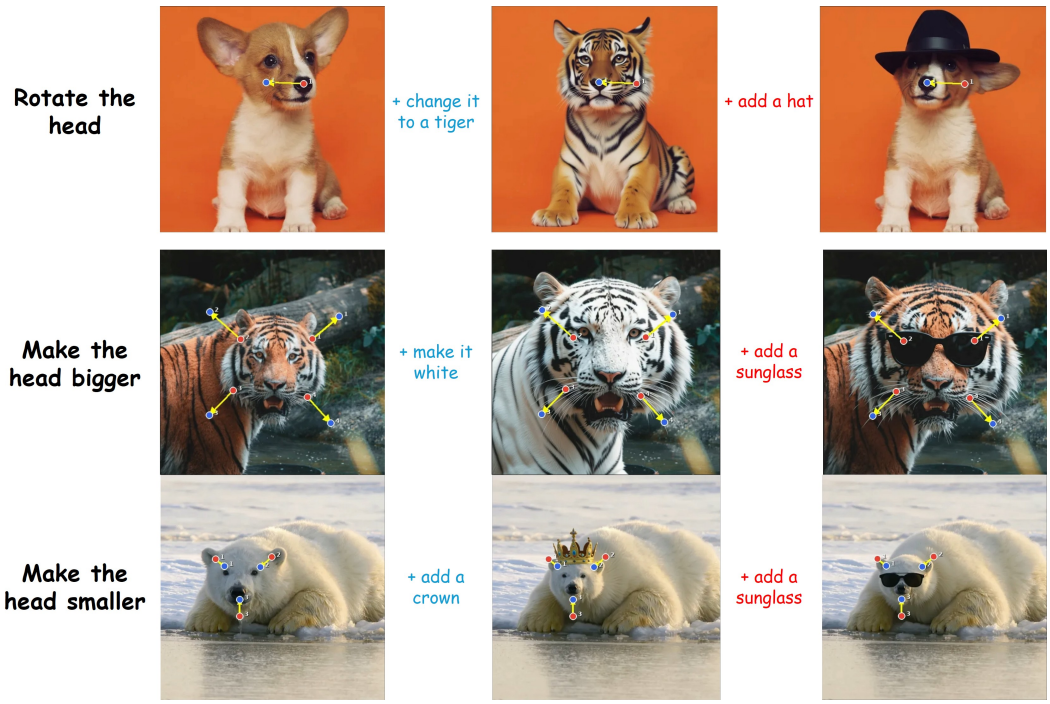

Figure A8: Results on simultaneous spatial control and semantic editing.

ent combinations of textual and visual inputs on semantic instruction following. Specifically, for trajectory-only inference, we remove the text input either from the control branch alone or from the entire model. Comparing Exp. 1 with Exp. 4, we see that text-only TV-Edit performs similarly to Qwen-Image-Edit baseline, indicating that spatial control is unavailable without trajectories. Comparing Exp. 2 with Exp. 5, we see that removing text from the control branch causes only minor degradation, demonstrating the robustness of the spatial branch, while the PF drop also shows that controller-side text benefits semantic execution. Comparing Exp. 3 with Exp. 5, we see that with completely empty text, the model retains reasonable geometric accuracy but suffers from a clear PF drop from 0.93 to 0.83, as it cannot distinguish actions with similar trajectories such as opening the mouth and raising the head. Finally, comparing Exp. 4 with Exp. 5, we see that adding visual prompts significantly reduces $\mathrm{MD_d}$ from 0.1410 to 0.0462 and PF from 0.87 to 0.93, showing that visual input provides spatial constraints and facilitates semantic execution through collaboration with text. Overall, text resolves semantic ambiguity, visual prompts provide spatial control, and their collaboration yields the best semantic and spatial accuracy.

Table A3: Results of text-only LoRA fine-tuning on TV-Edit-23K.

Method	$\mathrm{DS}^{\mathrm{tar}}_{\mathrm{global}} \uparrow$	$\mathrm{DS}^{\mathrm{tar}}_{\mathrm{local}} \uparrow$	$\mathrm{MD}_d \downarrow$	PF ↑
Qwen-Image-Edit	.8575	.9191	.1379	.86
+ text-only LoRA	.8806	.9230	.1254	.87

Text-only LoRA fine-tuning on TV-Edit-23K. Table A3 shows the comparison of LoRA fine-tuning on Qwen-Image-Edit using instruction–image pairs from TV-Edit-23K, in order to disentangle the contributions of data and model architecture. Training on a large collection of motion-related samples better adapts the model to such editing patterns, leading to a moderate improvement on the benchmark. However, the gain remains limited, and the relatively high MD_d indicates that text alone cannot reliably control motion magnitude. This suggests that text-only supervision is insufficient for modeling motion magnitude, and multimodal architecture and joint training paradigm are the keys to effectively improving performance on the TV-Edit task.

Table A4: Quantitative comparison of drag-based editing methods on DragBench.

Method	IF (↑)	MD (↓)
DragDiffusion [25]	0.88	32.13
DragNoise [20]	0.89	35.17
AdaptiveDrag [7]	0.86	35.70
CLIPDrag [12]	0.88	32.30
GoodDrag [54]	0.87	24.26
LightningDrag [33]	**0.89**	29.10
DragLora [44]	0.87	23.77
GeoDrag [28]	0.85	29.24
TV-Edit-Qwen (ours)	0.86	**17.31**

Table A5: User study on overall editing quality.

Metric	**TV-Qwen**	**TV-Kontext**	**NanoBanana**	**GoodDrag**	**LongCat**	**LightningDrag**
Avg. Rank ↓	**2.4905**	2.5048	3.5439	3.7476	4.0429	4.6714
Top-1 (%) ↑	27.62	**30.95**	14.76	8.10	13.33	5.24
Top-2 (%) ↑	**61.43**	56.19	34.29	18.57	20.00	9.52

F Comparison on Drag-Bench

To demonstrate the generalization ability of our approach in terms of geometric accuracy, we follow prior work [44] and conduct experiments on the drag-based benchmark DragBench [25], comparing our method both quantitatively and qualitatively against several state-of-the-art baselines.

F.1 Quantitative Comparison

For inference on DragBench, we first employ Qwen-3-VL [2] to analyze the images alongside their visual prompts, then utilize the inferred semantic changes as textual instructions for TV-Edit-Qwen. Following prior work [44, 25], we evaluate our method using MD and IF, which are computed as the mean point displacement distance and $1-$LPIPS, respectively. As shown in Table A4, TV-Edit-Qwen achieves an excellent balance between geometric accuracy and image fidelity. Notably, it yields an MD of 17.31, which is significantly lower than existing methods. Such a low MD indicates a highly successful geometric edit, which inevitably introduces some deviation from the original image. Nevertheless, even without requiring pre-defined edit region masks, TV-Edit-Qwen achieves an IF score of 0.86, surpassing GeoDrag and comparable to most state-of-the-art approaches.

F.2 Qualitative Comparison

Figure A9 compares TV-Edit-Qwen with state-of-the-art drag-based methods, including DragDiffusion [25], DragLora [44], GeoDrag [28] and GoodDrag [54]. From the visual comparisons, one can see that our TV-Edit-Qwen achieves much better geometric accuracy and fidelity. Specifically, in the second row where the visual prompt aims to open the lion's mouth, only TV-Edit-Qwen successfully executes this semantic transformation. GoodDrag attempts to follow the prompt but introduces artifacts, while the other methods show no noticeable changes. This highlights the inherent difficulty of previous visual-prompt-only methods in handling complex semantic actions. Furthermore, the sixth row demonstrates our model's robust spatial editing capabilities: TV-Edit-Qwen correctly rotates the car head with high visual quality. In contrast, among the baselines, only GoodDrag exhibits a visible rotation, whereas DragDiffusion fails to achieve the edit and generates artifacts.

G User Study

To further validate the effectiveness of TV-Edit, we conduct a user study comparing it against several representative methods. We select two variants of TV-Edit along with other competitive baselines that

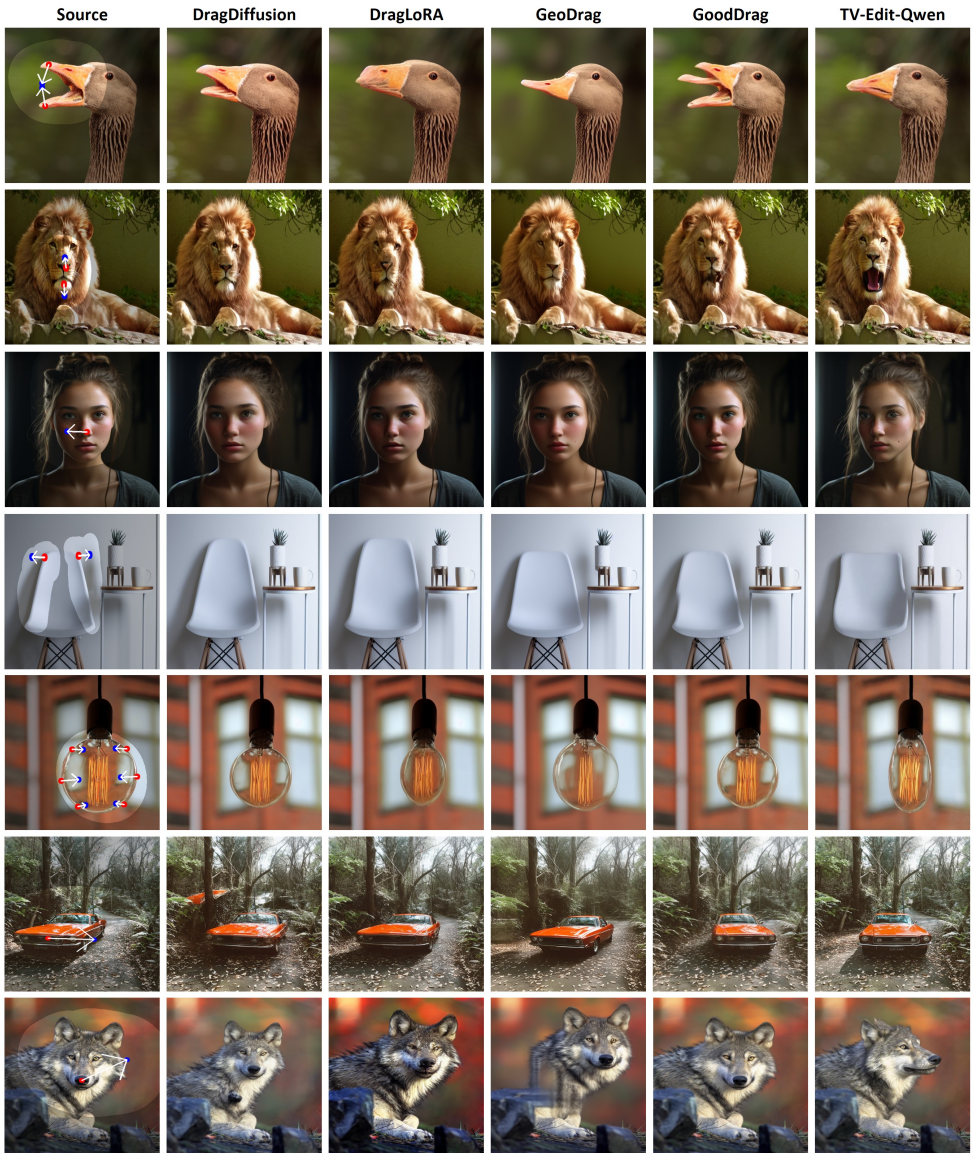

Figure A9: Visual comparisons among TV-Edit and drag-based methods on DragBench.

rank high in quantitative results, including NanoBanana Pro [8], GoodDrag [54], LongCat-Image-Edit [37], and LightningDrag [33]. For the experimental setup, we randomly sample 30 images from TV-Edit-Bench and anonymously present the editing results of all methods to participants via a web interface, asking them to rank the methods by overall editing quality, considering both instruction following and geometric accuracy. The user study interface is shown in Fig. A10. A total of 15 volunteers participate, and we report the average rank of each method in Table A5. As shown in the table, TV-Edit-Kontext achieves the highest Top-1 rate, while TV-Edit-Qwen achieves the best average rank and Top-2 rate. Both are preferred substantially more often than instruction- and drag-based baselines, supporting our automated results.

H Failure Cases

We show several failure cases of TV-Edit in Fig. A11. In the left case, when multiple objects are moved simultaneously in a complex scene, the editing of some objects may fail. A possible solution is to perform multi-round editing on individual objects, which may yield better results. In the right

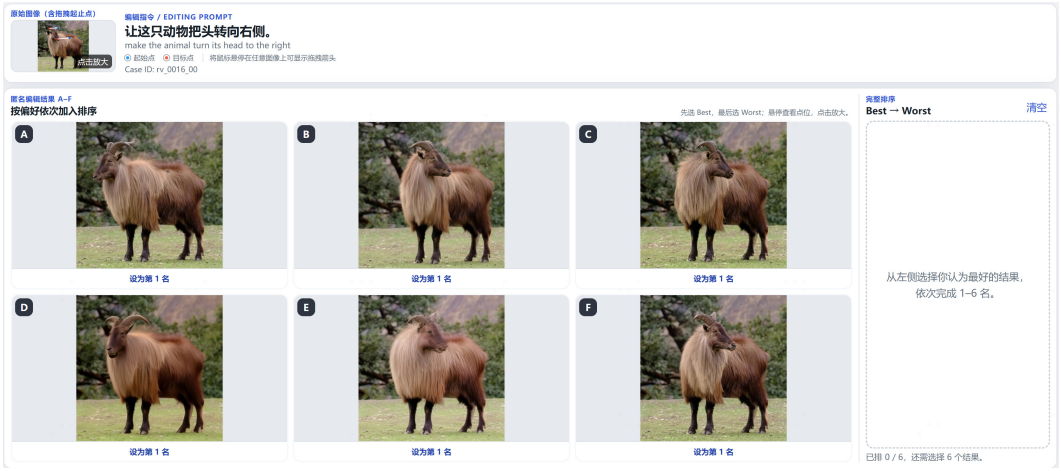

Figure A10: User study web interface.

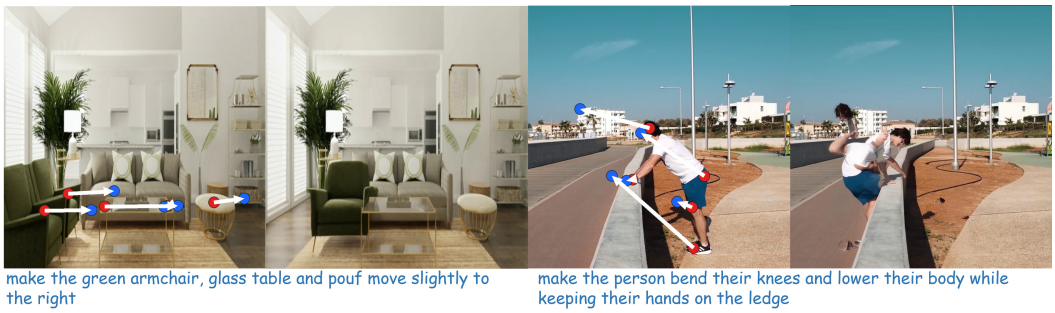

Figure A11: Failure cases.

case, the instruction mentions changes in the knees and hands but does not mention the jumping motion, which conflicts with the trajectory condition to some extent. Under such complex motion cases, the editing collapses and generates artifacts. For controlling complex motions, more accurate instruction descriptions and consistent trajectories can help improve the editing quality.

I Broader Impacts

Our proposed image editing method has several potential societal impacts, both positive and negative. On the positive side, this method can enhance creative industries by providing artists and designers with powerful tools for content creation and modification. However, there are potential negative impacts to consider. The technology could be misused for creating misleading or harmful content, which could have significant implications for privacy and security. To mitigate these risks, we suggest implementing mechanisms for monitoring and controlling the use of the technology, such as gated releases and developing tools to detect and counteract malicious uses.

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: Justification: Yes, the main claims in the abstract and introduction accurately reflect the paper's contributions and scope.

   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: We discuss limitations in the final section (after the conclusion).

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: [Yes]

   Justification: Yes, the paper provides a full set of assumptions and a proof for each theoretical result.

   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: Yes, the paper fully discloses all necessary information to reproduce the main experimental results, supporting the main claims and conclusions.

   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: All data are publicly available. We have released the codes and new data.

   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: Yes, the paper provides all necessary details, including hyperparameters, evaluation metrics and *etc.* to fully understand the results.

   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: We do not report error bars.

   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: We detail the type of compute resources in experiment section.

   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: Yes, the research in the paper fully conforms to the NeurIPS Code of Ethics in every respect.

   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: Yes, we analyze the potential social impact in appendix.

    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: The paper poses no such risks.

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: Yes, we clearly indicate the baseline methods and testing data used in the paper. Their licenses permit use with academic scope.

    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: [Yes]

    Justification: Code and dataset have been released.

    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: The paper does not involve crowd sourcing nor research with human subjects.

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: The paper does not involve crowd sourcing nor research with human subjects.

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: We describe the use of Qwen-3-VL for data annotation in Section 3.2.

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.