Drag as Evidence: Motion-Grounded Latent Recomposition for Drag-Based Editing

Xinyu Pu[1,3] **Hongsong Wang**[2*] **Jie Gui**[1,3*] **Pan Zhou**[4]
[1]School of Cyber Science and Engineering, Southeast University, Nanjing, China
[2]School of Computer Science and Engineering, Southeast University, Nanjing, China
[3]Purple Mountain Laboratories, Nanjing, China
[4]Singapore Management University, Singapore
{xinyupu, hongsongwang, guijie}@seu.edu.cn, panzhou3@gmail.com

Abstract

Modern image editors excel at semantic manipulation and visual synthesis, yet remain limited in precise spatial control, motivating the development of drag-based editing. However, existing drag-based methods often struggle to balance drag accuracy with natural, plausible, and intent-aligned generation. We propose *MoRe-Drag*, a motion-grounded drag-based editing method. Our key insight is to treat pixel-space warping as coarse motion evidence, and to inject this evidence into the generative sampling trajectory. Specifically, *MoRe-Drag* performs region-aware latent recomposition over refinement, inpainting, and preservation regions, coupled with stage-adaptive conditioning that progressively shifts from motion-grounded structure formation to semantic refinement. We further support a no-manual-instruction interface by adapting the MLLM-based text encoder for drag-aware instruction inference. Experiments on DRAGBENCH-SR and DRAGBENCH-DR show that *MoRe-Drag* substantially improves drag precision over strong base editors and achieves superior drag accuracy among SOTA drag-based methods, while delivering strong semantic consistency and visually realistic results. Project page: https://xinyu-pu.github.io/projects/more-drag/.

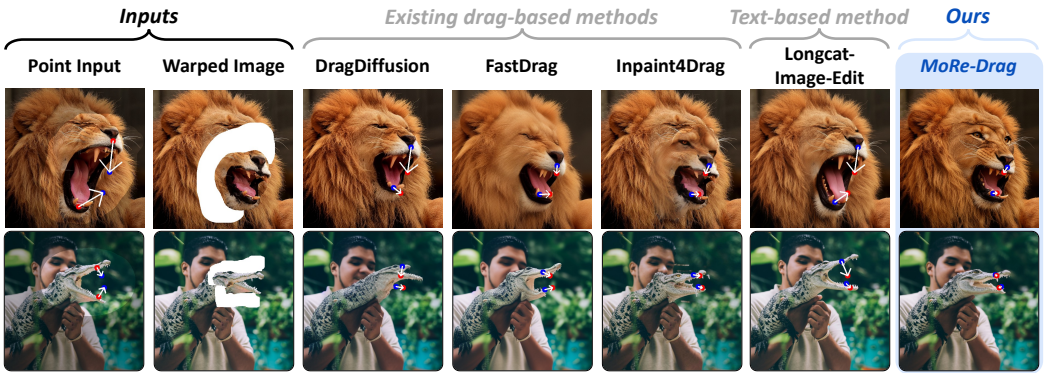

Figure 1: **Editing comparison.** Pixel-space warping provides explicit but imperfect motion evidence. *MoRe-Drag* injects this evidence into the generative trajectory, enabling the editor to follow the specified motion while restoring natural structure and appearance.

*Corresponding authors.

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

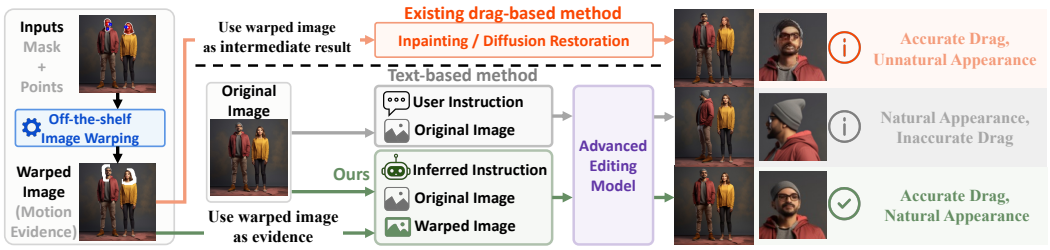

Figure 2: **Conceptual comparison with existing drag-based and text-based editing paradigms.** Existing drag methods use warped image as an intermediate result, leading to accurate but unnatural edits, while text-based editors produce natural results without precise drag control. Our method uses warped image as motion evidence to achieve both accurate dragging and natural appearance.

1 Introduction

Image generative models have advanced rapidly in realism, instruction following, and identity preservation [1–9], driven by progress in data curation [10, 11], model training [12–17], and architectural improvements [18, 19]. These advances have established natural language as an effective interface for semantic image generation and editing. Recent general-purpose image editing models [20–22] further strengthen this paradigm by leveraging MLLM-based text encoders and joint image-text conditioning, enabling image-aware instruction representations and supporting high-fidelity and visually consistent edits. However, as shown in the gray-shaded middle panel of Fig. 2, text-based instructions are primarily semantic: while effective at specifying what should be changed, they often fall short for spatially grounded manipulations, which require users to specify where the content should be adjusted and to what extent. This limitation motivates drag-based editing as a dedicated interface that provides explicit motion control through handle-to-target point pairs [23, 24].

Motivation. Existing drag-based editing methods usually preserve the identity and appearance of the input image via image-specific reconstruction, such as LoRA fine-tuning, inversion, or cached key-value features [24–27]. They then realize drag control either by optimizing latent representations with motion supervision and point tracking [24, 27–32], or by applying explicit deformation in the latent or pixel space [25, 26, 33]. Nevertheless, as shown in the top panel of Fig. 2, accurate dragging and natural synthesis remain difficult to reconcile: strong motion constraints can distort object structure, compromise texture consistency, or introduce unnatural artifacts, as also illustrated by existing drag-based methods in Fig. 1. Conversely, stronger generative regularization can produce realistic images, but often fails to fully satisfy the requested deformation, as seen in the text-based method in Fig. 1. This precision–naturalness trade-off remains a central obstacle to reliable drag-based editing. This limitation motivates us to revisit drag-based editing through modern general-purpose image editors, whose strong generative priors support realistic and visually consistent synthesis. However, drag interaction requires explicit motion control beyond text-based instruction. The key challenge is therefore ***how to couple the strong generative prior of modern editors with explicit motion evidence to obtain natural and motion-consistent edits.***

Contribution. To address this challenge, we make a crucial shift in how pixel-level dragging contributes to the editing process. As illustrated in the bottom green panel of Fig. 2, the warped image is not treated as an intermediate edit, but as motion evidence that coarsely encodes the desired deformation. This evidence complements the generative prior of advanced image editors with explicit drag control. Based on this insight, we propose *MoRe-Drag*, a **Mo**tion-grounded latent **Re**composition framework, which consists of three key components: region-aware recomposition for motion-evidence, disoccluded, and preservation regions; generation-stage adaptive conditioning to balance motion guidance and semantic refinement across timesteps; and instruction interaction to translate visual drag context into editor-friendly textual guidance. Together, they enable accurate dragging with natural synthesis.

We instantiate these components as follows. **(i) Region-aware recomposition.** We partition the warped observation into three non-overlapping regions: motion-evidence regions, where warped pixels provide coarse motion cues; disoccluded regions, which should be semantically completed; and preservation regions, where the content should remain unchanged. During sampling, MoRe-Drag maintains separate latent trajectories and recomposes them according to these regional roles.

(ii) Generation-stage adaptive conditioning. Since early sampling timesteps determine coarse layout and geometry while later refine appearance and texture details, MoRe-Drag first emphasizes the warped observation to stabilize the drag-induced structure, and then shifts toward the original image and editing instruction for semantic refinement. This design mitigates both warp-induced unnaturalness and instruction-only artifacts. **(iii) Instruction interaction.** To reduce the interaction burden, we attach a lightweight adapter to the editing model's MLLM-based text encoder to infer drag-aware instructions from the visual drag annotations. The adapter is activated only for instruction inference and disabled during generation, ensuring that the generative decoder receives embeddings aligned with the original encoder distribution. As illustrated in Fig. 2, our method differs from warp-then-inpaint pipelines such as Inpaint4Drag [34] by using the warped image solely as motion evidence rather than as an intermediate edited result, thereby enabling precise drag control with natural synthesis.

MoRe-Drag significantly boosts controllable drag editing over text-instruction-only baselines. As shown in Fig. 1, *MoRe-Drag* produces natural and accurate edits. By grounding the same editor (LongCat-Image-Edit [21]) with explicit motion evidence, drag accuracy—measured by Mean Distance (MD)—improves from 46.73 to 20.03 on DRAGBENCH-DR [24] and from 53.62 to 18.54 on DRAGBENCH-SR [35].

2 Related Work

Text-guided image editing. Early text-guided image editing methods manipulate pretrained diffusion models through concept learning [36], attention control [3], and inversion [37]. With advances in flow-based generative modeling [14, 15] and diffusion transformers [18], recent editors adopt stronger backbones and multimodal conditioning for open-domain editing [19, 38]. In particular, MLLM-powered editors, such as FLUX.2 [39], Qwen-Image-Edit [20], LongCat-Image-Edit [21], Step1X-Edit [22], and FireRed-Image-Edit [40], jointly interpret the reference image and editing instruction with multimodal encoders, improving instruction following and appearance preservation. However, text alone remains limited for precise spatial motion control.

Drag-based image editing. Drag-based editing addresses this limitation through explicit spatial control. DragGAN [23] introduces drag-based editing with point tracking and motion supervision, and DragDiffusion [24] extends it to diffusion models. Subsequent methods improve robustness, accuracy, and efficiency [25–30, 35, 41–43]. Inpaint4Drag [34] formulates drag editing as pixel-space warp-then-inpaint, whereas MoRe-Drag uses the warped image only as motion evidence to guide a generative editor rather than as an intermediate edit. Recent methods enable drag controllability on MMDiT-based models via explicit correspondence, inversion, region supervision, latent manipulation, or attention alignment [44–46]. In contrast, MoRe-Drag keeps the MMDiT editor unchanged and injects motion externally through conditioning and latent recomposition. Although ContextDrag [46] also builds on a pretrained editing model, it relies on attention-level reference injection, whereas *MoRe-Drag* uses motion evidence to better couple precise motion control with natural synthesis.

3 Methodology

We first formalize the task of interest. Given an image, user-specified point pairs, and deformation masks, our goal is to produce a natural edit while precisely aligning each handle point with its target point. Recent latent- and pixel-warping methods [25, 26, 34] treat the warped outputs as intermediate results, followed by inpainting or restoration to obtain the final drag result (see top of Fig. 2). Text-based methods rely solely on textual instructions to control editing, resulting in limited motion accuracy (see middle of Fig. 2). Moreover, requiring textual instructions further increases the user burden. To address these issues, we introduce *MoRe-Drag*, which combines coarse motion evidence from the warped image with the generative priors of advanced image editors for natural and accurate drag editing (see bottom of Fig. 2).

MoRe-Drag first constructs pixel-space motion evidence. Specifically, following off-the-shelf image warping [34], we interpolate sparse handle displacements $\boldsymbol{d}_i = \boldsymbol{t}_i - \boldsymbol{h}_i$ into a dense mask-aware motion field $\boldsymbol{F}(\boldsymbol{x}) = \sum_i \frac{w_i(\boldsymbol{x})\boldsymbol{d}_i}{\sum_i w_i(\boldsymbol{x})}$, and bidirectionally warp original image $\boldsymbol{X}_o$ to obtain a coarse evidence image $\boldsymbol{X}_w$, with invalid correspondences marked as disocclusions (detailed in Appendix C.4). Then, as illustrated in Fig. 3, this evidence is incorporated into flow sampling via two components: (i)

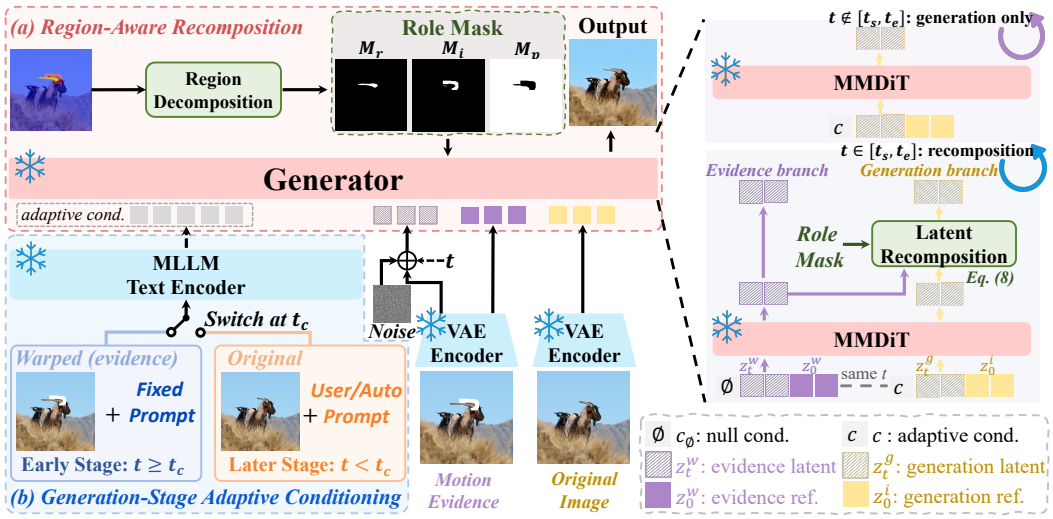

Figure 3: **Overview of *MoRe-Drag*.** The evidence branch and the generation branch are updated by the same MMDiT at the same timestep, and interact only through role-mask recomposition within $[t_s, t_e]$ (right). (a) **Region-aware recomposition**: the evidence and original image are decomposed into role-specific regions, and the resulting role masks guide latent recomposition (Eq. (8)), so motion cues are injected only into the corresponding regions. (b) **Generation-stage adaptive conditioning**: at early timesteps ($t \geq t_c$), the model uses the warped image with a fixed prompt to enforce motion consistency; at later timesteps ($t < t_c$), it switches to the original image with the user/auto prompt for semantic and appearance refinement. The switch time t_c is set independently of $[t_s, t_e]$.

a region-aware recomposition module that injects motion evidence through region decomposition and latent recomposition (Sec. 3.1); (ii) a generation-stage adaptive conditioning strategy that balances motion evidence with semantic refinement across timestep-based condition switch (Sec. 3.2). As shown in Fig. 5, we fine-tune the MLLM, which also serves as the generator's text encoder, to support automatic instruction inference (Sec. 3.3). For clarity, we provide algorithmic pseudocode in Appendix D.

3.1 Region-Aware Recomposition

The image warping algorithm [34] propagates the sparse handle-to-target displacements across the source region, deforming it according to the indicated motion directions and magnitudes. However, direct warping provides only coarse evidence—it introduces unnatural geometry, local distortions, boundary artifacts, and semantic misalignment, e.g., mouth shapes of the lion and crocodile in Fig. 1. This raises a key requirement: the editing process should retain the geometric faithfulness of the warp while leveraging the generative prior of the editing model to restore realism and semantic coherence. To this end, we first decompose the warped observation into three functional regions for refinement, hole inpainting, and background preservation (see Fig. 3 (a, top)), and then perform latent recomposition to inject the warped image as explicit motion evidence during generation (see Fig. 3 (a, right)). This allows the model to follow the user-specified motion while restoring realistic geometry, plausible semantics, and source-region consistency.

Region decomposition. We obtain the transported target mask $\boldsymbol{M}_t$ by applying the same warp to the source deformation region. As illustrated in the top-left part of Fig. 3, let $\boldsymbol{\Omega}$ be the image domain, direct warping is particularly error-prone near the transported boundary. We therefore augment $\boldsymbol{M}_t$ with a narrow boundary band $\boldsymbol{M}_{\partial t}$ extracted from the contour of the target mask, and define the refinement mask as

$$\boldsymbol{M}_r = \boldsymbol{M}_t \cup \boldsymbol{M}_{\partial t}. \tag{1}$$

To identify the regions that remain uncovered after transport, we take the difference between the source mask and the transported target mask, dilate it to absorb boundary discontinuities, and exclude the refinement mask to avoid overlap:

$$\boldsymbol{M}_i = \mathrm{Dilate}(\boldsymbol{M}_s \setminus \boldsymbol{M}_t) \setminus \boldsymbol{M}_r, \tag{2}$$



Figure 4: Toy editing from different initial timesteps. Top: starting latent timestep; bottom: LPIPS to the original image.

where the dilation kernel size is 5. Finally, all remaining pixels form the preservation mask:

$$\boldsymbol{M}_p = \boldsymbol{\Omega} \setminus (\boldsymbol{M}_r \cup \boldsymbol{M}_i). \tag{3}$$

The three masks provide explicit spatial roles for generation. $\boldsymbol{M}_r$ guides refinement of the dragged content, $\boldsymbol{M}_i$ specifies the area to be inpainted, and $\boldsymbol{M}_p$ constrains the non-edited region to remain consistent.

Latent recomposition. Given the decomposed masks, we perform latent recomposition by assembling latent features within their corresponding mask regions. As illustrated in the right part of Fig. 3, we maintain two generation trajectories. The yellow trajectory on the right corresponds to conditioned generation branch and is initialized from Gaussian noise:

$$\boldsymbol{z}_1^g \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}). \tag{4}$$

We update it by integrating the flow-matching velocity field conditioned on the MLLM text encoder embedding $\boldsymbol{c}$ and reference image $\boldsymbol{z}_0^{image}$, which is obtained by encoding the image with the VAE $\mathcal{E}$:

$$d\boldsymbol{z}_t^g = v_\theta(\boldsymbol{z}_t^g, t, \boldsymbol{c}, \boldsymbol{z}_0^{image})dt, \quad t: 1 \to 0, \quad \text{where} \quad \boldsymbol{z}_0^{image} = \mathcal{E}(\boldsymbol{X}_o) \tag{5}$$

The resulting latent $\boldsymbol{z}_t^g$ serves as the conditional generation branch for subsequent recomposition. The purple trajectory on the left corresponds to the evidence branch. We first encode the warped image $\boldsymbol{X}_w$ into the latent space, and then perturb the resulting latent to a noisy state at timestep t for initialization:

$$\boldsymbol{z}_t^w = (1-t)\boldsymbol{z}_0^w + t\boldsymbol{\epsilon}^w, \qquad \boldsymbol{\epsilon}^w \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}), \quad \boldsymbol{z}_0^w = \mathcal{E}(\boldsymbol{X}_w), \tag{6}$$

The evidence branch follows a null-conditioned reconstruction flow:

$$d\boldsymbol{z}_t^w = v_\theta(\boldsymbol{z}_t^w, t, \boldsymbol{c}_\emptyset, \boldsymbol{z}_0^w)dt, \quad t: 1 \to 0. \tag{7}$$

The evidence branch provides drag-induced spatial cues as well as reconstruction guidance for the non-edited regions. As shown in the right part of Fig. 3, since both $\boldsymbol{z}_t^g$ and $\boldsymbol{z}_t^w$ are defined at the same flow timestep, they can be recomposed in latent space without scale or timestep misalignment. We resize the refinement, inpainting, and preservation masks to the latent resolution, denoted as $\boldsymbol{m}_r$, $\boldsymbol{m}_i$, and $\boldsymbol{m}_p$. During the timestep interval $\mathcal{T} = [t_s, t_e]$, we recompose the sampling latents:

$$\boldsymbol{z}_{t'}^g \leftarrow \boldsymbol{m}_i \odot \boldsymbol{z}_{t'}^g + \boldsymbol{m}_p \odot \boldsymbol{z}_{t'}^w + \boldsymbol{m}_r \odot \left(\lambda(t')\boldsymbol{z}_{t'}^g + (1-\lambda(t'))\boldsymbol{z}_{t'}^w\right), \quad t' \in \mathcal{T}. \tag{8}$$

Here, $\lambda(t')$ is a weight controlling the strength of generative refinement. The inpainting region fully relies on the conditioned generation branch to complete the hole, the preservation region adopts the evidence branch to preserve unchanged content, and the refinement region blends both branches to correct geometric artifacts while retaining the drag-induced spatial structure. This region-aware recomposition injects explicit pixel-space motion evidence into the latent sampling trajectory, while leveraging the model's prior to produce natural deformation, plausible completion, and consistent background preservation.

Weight schedule. We experimented with three timestep-dependent schedules, including constant, linear, and inverse-square decay from $\lambda_{\max}$ to $\lambda_{\min}$. Considering its simplicity, efficiency, and stable empirical performance, we use the constant schedule by default. The quantitative comparison of different schedules is reported in Table 3. The formulations of schedules are provided in Appendix C.3.

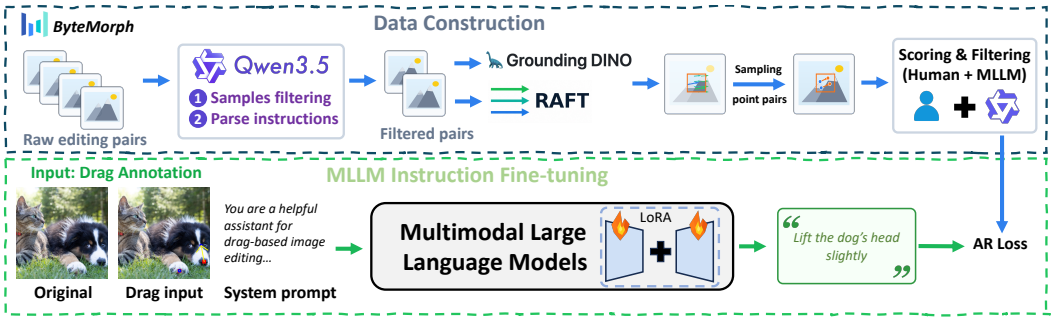

Figure 5: Instruction interaction and data construction. Top: drag annotation construction from BYTEMORPH. Bottom: LoRA-based MLLM fine-tuning with AR loss, enabling automatic instruction inference from drag annotations.

3.2 Generation-Stage Adaptive Conditioning

While region-aware recomposition successfully injects motion evidence into the sampling trajectory, the generator prior still performs semantic refinement based on the source image and editing instruction. As a result, the sampling process may be driven by two competing spatial tendencies: the motion evidence specifies the desired drag target, whereas the generator prior may favor another edit location. When these two tendencies are substantially misaligned, the model attempts to satisfy both, producing duplicated or split edits across different regions, as shown in Fig. 8 (only stage 2). This indicates a competition between the editor's semantic prior and the injected motion evidence.

A common understanding in diffusion/flow-based generation is that early sampling steps tend to establish low-frequency structure and high-level semantics, while later steps increasingly recover high-frequency details [47, 48]. We further conduct a toy editing experiment: given a real image ($t = 0.00$) and an edit instruction, we perturb the image to different timesteps along the forward flow path and start instruction-conditioned generation from each perturbed state. As shown in the Fig. 4, even at an early sampling timestep $t = 0.80$, the major image structures and spatial layout have already been determined, while the remaining steps mainly refine appearance details. This observation motivates us to inject motion evidence in the early sampling stage, when spatial decisions are still being formed, and rely on the instruction for later refinement. Thus, as shown in Fig. 3(b), we introduce adaptive conditioning into conditioned generation branch. Specifically, during the early stage, we define the conditioning embedding $\boldsymbol{c}$ as the joint encoding of the warped image (motion evidence) $\boldsymbol{X}_w$, and a fixed prompt p_{fix}:

$$\boldsymbol{c}^w = \Phi_{\text{mllm}}(\boldsymbol{X}_w, p_{\text{fix}}), \qquad \boldsymbol{z}_0^{\text{img},w} = \mathcal{E}(\boldsymbol{X}_w), \tag{9}$$

where $\boldsymbol{c}^w$ is the early stage condition, and Φ_{mllm} denotes MLLM text-encoder. p_{fix} encourages the model to correct distortions and fill holes in the motion evidence; the full prompt is provided in Box G.5. Then, switching to the original image and the user/auto prompt during later stage:

$$\boldsymbol{c}^o = \Phi_{\text{mllm}}(\boldsymbol{X}_o, p), \qquad \boldsymbol{z}_0^{\text{img},o} = \mathcal{E}(\boldsymbol{X}_o). \tag{10}$$

With this stage-adaptive design, the conditional generation trajectory is sampled as

$$d\boldsymbol{z}_t^g = v_\theta\left(\boldsymbol{z}_t^g, t, \boldsymbol{c}(t), \boldsymbol{z}_0^{\text{img}}(t)\right) dt, \quad \text{where } \left(\boldsymbol{c}(t), \boldsymbol{z}_0^{\text{img}}(t)\right) = \begin{cases} \left(\boldsymbol{c}^w, \boldsymbol{z}_0^{\text{img},w}\right), & t \geq t_c, \\ \left(\boldsymbol{c}^o, \boldsymbol{z}_0^{\text{img},o}\right), & t < t_c. \end{cases} \tag{11}$$

Here, t_c controls the transition from motion-evidence-guided structure formation to source-image-guided semantic refinement. This switching timestep should be properly set to ensure a smooth conditioning transition, balancing motion-evidence injection and semantic refinement. We provide an analysis of its effect in Table 13.

This adaptive conditioning strategy complements the region-aware recomposition in Sec. 3.1 by establishing a dual pathway for motion-evidence injection, while early-stage conditioning guides generation toward the desired motion structure and aligns the model prior with the motion evidence. By first grounding the generation trajectory in the motion evidence and then refining it with the original image and instruction, *MoRe-Drag* improves the alignment between motion evidence and editing instructions while retaining the model's ability to fix artifacts and synthesize plausible content.

3.3 Instruction Interaction

The above components rely on textual instructions, which places an additional burden on users. To alleviate this, we introduce a no-manual-instruction interface that automatically derives drag-aware instructions from drag-annotation visualizations. Specifically, we reuse the MLLM-based text encoder of the base editor for instruction inference, without introducing any additional standalone model. The inferred instruction is then used for subsequent generation.

Directly using the original MLLM for instruction inference can yield inaccurate intents, such as misidentifying the edited object or introducing semantic changes unsupported by the drag annotations (see Fig. 9). Therefore, as illustrated in the top part of Fig. 5, we construct a high-quality drag-annotation dataset from the released demo split of BYTEMORPH [49]. Starting from 780K samples, we first remove 560K camera-centric samples using Qwen3.5-9B [50]. We then estimate optical flow with RAFT [51], localize edited subjects with Grounding DINO [52], and cluster regional flow vectors by magnitude and direction to sample handle–target pairs. To further improve annotation quality, we conduct a hybrid human-and-MLLM scoring review of the generated drag annotations, resulting in 68K high-quality samples for subsequent model fine-tuning, detailed in Appendix B.

As shown in the bottom part of Fig. 5, we fine-tune the MLLM using LoRA with the standard AR loss, implemented with an open-source Qwen-VL fine-tuning framework[2]. The adapter is activated only for instruction inference and disabled during generation, so the MMDiT decoder still receives conditioning features from the original encoder distribution. This bridges geometric interaction and semantic conditioning without altering the generative pathway.

3.4 Discussion

We discuss the motivation behind each design choice. (1) *Why motion evidence?* A pretrained editor offers a strong natural-image prior but no pixel-level motion grounding, whereas the warped image specifies the desired displacement but contains artifacts; we therefore use the warp as motion evidence rather than the final result. (2) *Why an evidence branch?* Relying only on the generation branch may let the prior weaken the displacement, while relying only on the warped image keeps its artifacts. The evidence branch is evolved by the same MMDiT, so its state matches the generation branch in flow timestep, although not necessarily in distribution, and can be fused repeatedly during sampling. (3) *Why region-aware recomposition (RAR)?* Warping leaves transported-but-distorted, disoccluded, and unchanged regions, which require different interactions between the two branches; RAR thus decides *where* the evidence can be trusted, which a single global fusion rule cannot provide. (4) *Why generation-stage adaptive conditioning (GAC)?* RAR decides where the evidence is injected, but the condition decides the direction of each update; GAC thus decides *when* the evidence or the prior should dominate, avoiding the failures of using either condition alone (see Fig. 8). (5) *Why instruction interaction?* The drag geometry is fully specified by the points, masks, and warped image, so the instruction only describes *what* content should be restored, and a short instruction inferred by the reused MLLM suffices (Sec. 3.3).

Overall, the evidence branch carries the requested motion, RAR decides *where* this evidence is reliable, GAC decides *when* it should dominate, the generation branch restores natural appearance and missing content, and instruction interaction provides high-level guidance on *what* should be restored.

4 Experiments

4.1 Implementation Details and Experimental Setup

Implementation Details. We use LongCat-Image-Edit [21] as the default base editor for *MoRe-Drag*, and instantiate it on Qwen-Image-Edit [20] as supplementary. Both editors adopt Qwen2.5-VL-7B [53] as the text encoder but use different MMDiT backbones. DragDiffusion [24], DragNoise [28], GoodDrag [30], and FastDrag [25] take drag points with a mask as input, whereas Inpaint4Drag [34] and *MoRe-Drag* use the drag-point-induced warped image. We also include an Inpaint4Drag variant implemented on top of FLUX.1-Fill-dev[3]. Further details are provided in Appendix C.

[2] https://github.com/2U1/Qwen-VL-Series-Finetune

[3] https://huggingface.co/black-forest-labs/FLUX.1-Fill-dev

Table 1: Results on DRAGBENCH-SR/DR. CP and PF are scaled by 100. Metrics of Qwen-Image-Edit and LongCat-Image-Edit are not highlighted, as they are not drag-based methods. "+ FLUX.1-Fill-dev" denotes an Inpaint4Drag variant using a stronger inpainting model. (*) indicates results from the original papers, as the models were not publicly available at the time of evaluation. In the Params column, (-) indicates that *MoRe-Drag* keeps the generative backbone frozen.

Model	Params	DRAGBENCH-SR			DRAGBENCH-DR		
		MD↓	CP↑	PF↑	MD↓	CP↑	PF↑
Test-time Optimization-based Methods							
DragDiffusion	2.1B	$41.08_{\pm 0.43}$	$92.75_{\pm 0.38}$	$77.15_{\pm 1.49}$	$31.61_{\pm 0.23}$	$96.46_{\pm 0.43}$	$78.45_{\pm 0.95}$
DragNoise	2.1B	$47.64_{\pm 1.08}$	$79.00_{\pm 0.25}$	$61.56_{\pm 0.31}$	$30.05_{\pm 0.01}$	$91.46_{\pm 1.34}$	$81.02_{\pm 0.18}$
GoodDrag	2.1B	$28.94_{\pm 0.40}$	$94.50_{\pm 0.13}$	$76.50_{\pm 0.39}$	$23.21_{\pm 0.12}$	$95.36_{\pm 0.24}$	$83.94_{\pm 0.04}$
Optimization-free Methods							
FastDrag	2.1B	$23.17_{\pm 0.19}$	$94.00_{\pm 0.25}$	$79.62_{\pm 4.38}$	$29.50_{\pm 0.12}$	$92.73_{\pm 0.73}$	$75.60_{\pm 1.63}$
ContextDrag*	12B	19.07	92.25	82.00	21.66	90.90	83.78
Inpaint4Drag	2.1B	$19.79_{\pm 0.18}$	$93.00_{\pm 0.13}$	$81.00_{\pm 0.04}$	$21.37_{\pm 0.32}$	$88.53_{\pm 0.25}$	$82.77_{\pm 0.76}$
+ *FLUX.1-Fill-dev*	12B	$18.96_{\pm 0.18}$	$94.69_{\pm 0.53}$	$81.41_{\pm 1.41}$	$21.16_{\pm 0.28}$	$91.70_{\pm 0.49}$	$77.19_{\pm 1.40}$
Qwen-Image-Edit	20B	$50.48_{\pm 0.12}$	$94.57_{\pm 0.63}$	$83.59_{\pm 0.76}$	$47.90_{\pm 0.27}$	$92.43_{\pm 0.41}$	$88.17_{\pm 0.03}$
+ *MoRe-Drag*	-	$23.85_{\pm 0.62}$	$90.15_{\pm 0.12}$	$70.96_{\pm 0.58}$	$26.35_{\pm 0.17}$	$87.31_{\pm 0.00}$	$78.75_{\pm 0.89}$
LongCat-Image-Edit	6B	$53.62_{\pm 0.09}$	$94.00_{\pm 0.50}$	$92.08_{\pm 1.17}$	$46.73_{\pm 0.24}$	$91.28_{\pm 0.55}$	$91.20_{\pm 0.10}$
+ *MoRe-Drag*	-	$\mathbf{18.54_{\pm 0.22}}$	$\mathbf{96.36_{\pm 0.38}}$	$\mathbf{86.36_{\pm 4.06}}$	$\mathbf{20.03_{\pm 0.21}}$	$\mathbf{93.15_{\pm 0.23}}$	$\mathbf{84.82_{\pm 0.98}}$

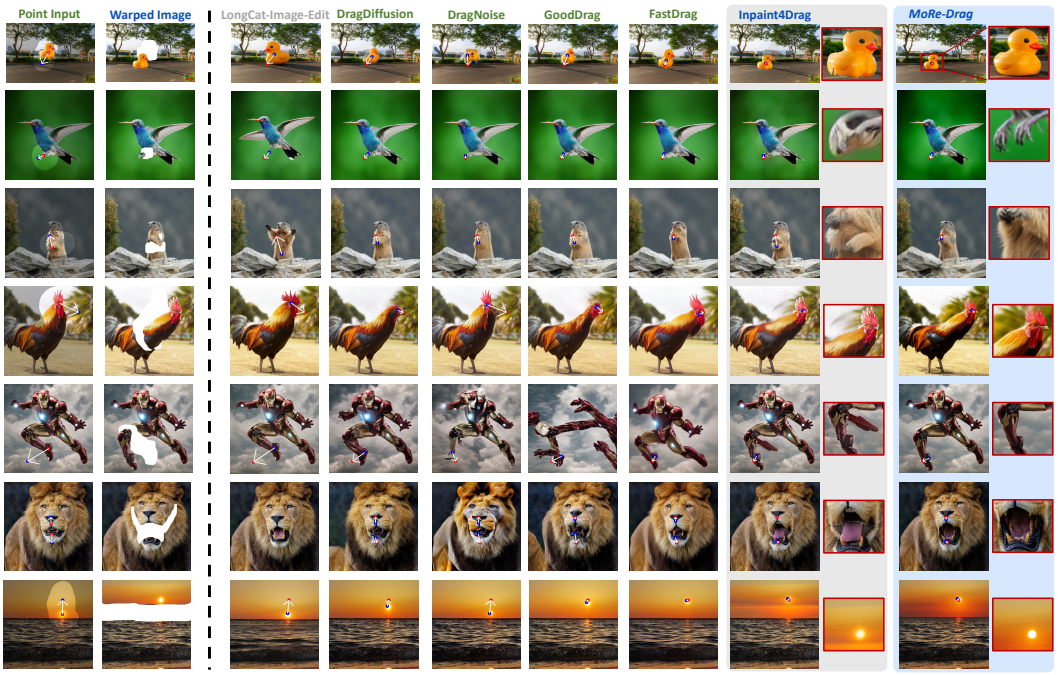

Figure 6: Qualitative comparisons with state-of-the-art drag-based editing models. Blue and red points denote handle and target points, respectively. LongCat-Image-Edit is a text-based editing method; Inpaint4Drag and *MoRe-Drag* take the warped image as input, while the other methods are driven by point inputs. We provide zoomed-in views of the edited regions for the two warped-image-based methods to highlight local structural fidelity. More results are shown in Fig. 11

Experimental Setup. We evaluate *MoRe-Drag* on DRAGBENCH-SR [35] and DRAGBENCH-DR [24]. Since neither benchmark provides natural-language edit instructions, we generate reference instructions with Qwen3.5-9B [50] followed by human correction for evaluation. Unless otherwise specified, *MoRe-Drag* uses inferred instructions produced by our LoRA-adapted MLLM. We report Mean Distance (MD) [24], Concept Preservation (CP) [54], and Prompt Following (PF) [54], each averaged over three runs with standard deviations. Details are provided in Appendices C.5 and C.6.

Table 2: Ablation results on DRAGBENCH-DR.

Variant	MD↓	CP↑	PF↑
MoRe-Drag	**20.03**$_{\pm 0.21}$	93.15$_{\pm 0.23}$	**84.82**$_{\pm 0.98}$
w/o RAR	25.41$_{\pm 0.23}$	92.50$_{\pm 0.18}$	77.14$_{\pm 1.16}$
w/o GAC	29.61$_{\pm 0.33}$	**93.78**$_{\pm 0.24}$	76.52$_{\pm 0.67}$

Table 3: Ablation results on DRAGBENCH-SR.

Variant	MD↓	CP↑	PF↑
Constant	18.54$_{\pm 0.22}$	96.36$_{\pm 0.38}$	**86.36**$_{\pm 4.06}$
Linear	18.41$_{\pm 0.03}$	96.25$_{\pm 0.37}$	82.25$_{\pm 0.00}$
Inv. square	**18.23**$_{\pm 0.10}$	**96.63**$_{\pm 0.13}$	80.14$_{\pm 1.86}$



Figure 7: Qualitative comparison of *MoRe-Drag* with variants without RAR or GAC.

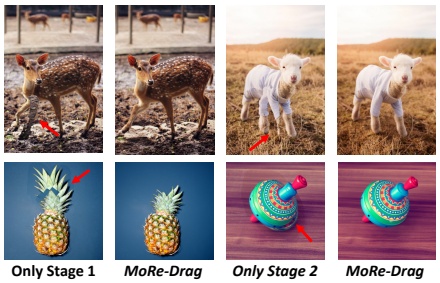

Figure 8: Effect of generation-stage adaptive conditioning.

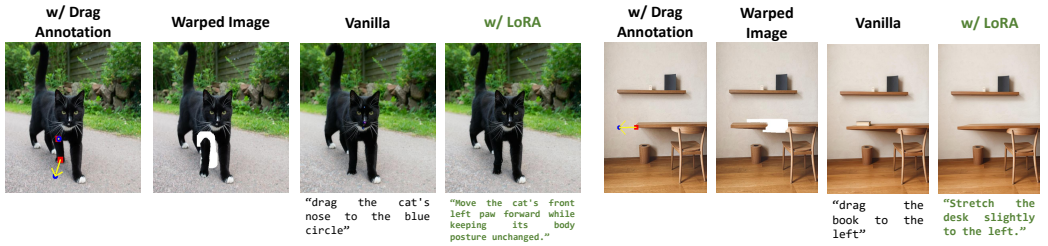

Figure 9: Instruction interaction examples. Compared with vanilla instruction inference, LoRA adaptation produces drag-aware instructions that better match the visual annotations, reducing unintended artifacts and hallucinated content.

4.2 Comparison with SOTAs

Quan titative analysis. Table 1 reports the quantitative results. *MoRe-Drag* consistently improves drag precision over the base editors, demonstrating its generality. The performance gap between Qwen-Image-Edit- and LongCat-Image-Edit-based variants mainly stems from the base editors themselves, especially their null-conditioned reconstruction ability; the reconstruction comparison is reported in Appendix E.9. *MoRe-Drag* based on LongCat-Image-Edit achieves the best drag accuracy on both benchmarks with strong CP and PF scores, showing improved spatial controllability while preserving visual and semantic consistency. Even with the same inputs as ours and a stronger FLUX.1-Fill-dev backbone, Inpaint4Drag still falls short in precise and natural drag editing, highlighting the benefit of using the warped image as motion evidence. Although base editors sometimes obtain slightly higher PF, they are designed for text-guided editing and thus naturally favor this metric; however, they lack precise spatial manipulation ability, leading to much higher MD. Runtime and peak GPU memory comparisons between *MoRe-Drag* and the base editor are reported in Appendix E.10.

Qualitative analysis. Fig. 6 presents qualitative comparisons with representative drag-based editing methods and our base editor (LongCat-Image-Edit). *MoRe-Drag* achieves more accurate drag control while producing visually natural edits with better preservation of object texture, shape, and local structures. Compared with prior drag-based methods, *MoRe-Drag* more effectively maintains boundary continuity, preserves fine structures during motion, and avoids distortions, e.g., the first row of Fig. 6. It also enables semantically coherent edits, such as generating realistic mouth details in the lion example and consistently adjusting the sky color and reflection in the sunset case.

Human preference study. We conduct a human study with 42 participants, each completing 15 randomized ranking tasks. In each task, the participant is shown the source image with drag annotations and three anonymized results from FastDrag [25], Inpaint4Drag [34], and *MoRe-Drag* in

Table 4: Human preference study with 42 participants. We report the average rank (lower is better).

Method	Naturalness↓	Drag Faithfulness↓	Preservation↓
FastDrag [25]	2.52	2.30	2.48
Inpaint4Drag [34]	2.02	2.25	1.88
MoRe-Drag	**1.47**	**1.45**	**1.64**

random order, and ranks them by naturalness, drag faithfulness, and preservation, where rank 1 is the best. As shown in Table 4, *MoRe-Drag* obtains the best average rank on all three criteria.

4.3 Ablation Studies

Region-aware recomposition and adaptive conditioning. Table 2 and Fig. 7 show that both Region-Aware Recomposition (RAR) and Generation-Stage Adaptive Conditioning (GAC) are important: removing RAR weakens local deformation control, while removing GAC produces less plausible structures. Fig. 8 further shows that using only the early-stage condition preserves motion cues but leads to under-refined distortions, whereas using only the later-stage condition can cause competition between semantic editing and injected motion evidence. Since different recomposition weight schedules perform similarly (Table 3), we use the constant schedule by default.

Instruction interaction. Fig. 9 illustrates drag-aware instruction inference. Misaligned instructions can introduce artifacts or unsupported content, whereas LoRA adaptation better captures the drag intent and improves editing faithfulness. Additional comparisons and the inference prompt template are provided in Appendix E.6.

5 Conclusion

In this paper, we presented *MoRe-Drag*, a drag-based editing framework that grounds modern editors with motion evidence. Through region-aware recomposition and stage-adaptive conditioning, *MoRe-Drag* improves spatial control while preserving natural and visually coherent image generation results. We also introduced a no-manual-instruction interface for drag-aware intent inference. Overall, our results suggest that pixel-space warping can serve as coarse motion evidence for advanced generative editors, enabling spatially accurate and visually natural drag-based manipulation.

Limitations and future work. Our method is primarily developed for editing backbones that use an MLLM as the text encoder and an MMDiT-style generative decoder. In our experiments, we find that the effectiveness of motion-evidence injection is related to the model's null-conditioned reconstruction ability, where stronger reconstruction generally leads to more reliable editing consistency and concept preservation. Future work may explore improved reconstruction priors and editor-agnostic calibration strategies to make motion grounding more uniformly effective across diverse generative editing backbones.

Acknowledgments and Disclosure of Funding

This work was supported by the National Natural Science Foundation of China (Grant Nos. 62676086, 62672109, 62302093, and 52441503), Jiangsu Province Natural Science Fund (BK20230833), the Open Research Fund of the State Key Laboratory of Multimodal Artificial Intelligence Systems under Grant E5SP060116, the Singapore Ministry of Education (MOE) Academic Research Fund (AcRF) Tier 1 grant (Proposal ID: 23-SIS-SMU-070), and the Lee Kong Chian Fellowship awarded to ZHOU Pan by Singapore Management University. This work was also supported in part by the CIPS-SMP-Zhipu Large Model Fund and was also partially supported by SEU Kunpeng&Ascend Center of Cultivation. We thank the Big Data Computing Center of Southeast University for providing the facility support on the numerical calculations. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the Ministry of Education, Singapore. Apart from the funding and support disclosed above, the authors report no financial relationships during the 36 months prior to submission that could be perceived as influencing this work.

References

[1] Hansam Cho, Jonghyun Lee, Seoung Bum Kim, Tae-Hyun Oh, and Yonghyun Jeong. Noise map guidance: Inversion with spatial context for real image editing. In *The Twelfth International Conference on Learning Representations, ICLR*, 2024.

[2] Jun Zhou, Jiahao Li, Zunnan Xu, Hanhui Li, Yiji Cheng, Fa-Ting Hong, Qin Lin, Qinglin Lu, and Xiaodan Liang. Fireedit: Fine-grained instruction-based image editing via region-aware vision language model. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, pages 13093–13103, 2025.

[3] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross-attention control. In *The Eleventh International Conference on Learning Representations, ICLR*, 2023.

[4] Ling Yang, Zhaochen Yu, Chenlin Meng, Minkai Xu, Stefano Ermon, and Bin Cui. Mastering text-to-image diffusion: Recaptioning, planning, and generating with multimodal llms. In *Forty-first International Conference on Machine Learning, ICML*, pages 56704–56721, 2024.

[5] Zechuan Zhang, Ji Xie, Yu Lu, Zongxin Yang, and Yi Yang. In-context edit: Enabling instructional image editing with in-context generation in large scale diffusion transformer. *arXiv preprint arXiv:2504.20690*, 2025.

[6] Chenyuan Wu, Pengfei Zheng, Ruiran Yan, Shitao Xiao, Xin Luo, Yueze Wang, Wanli Li, Xiyan Jiang, Yexin Liu, Junjie Zhou, Ze Liu, Ziyi Xia, Chaofan Li, Haoge Deng, Jiahao Wang, Kun Luo, Bo Zhang, Defu Lian, Xinlong Wang, Zhongyuan Wang, Tiejun Huang, and Zheng Liu. Omnigen2: Exploration to advanced multimodal generation. *arXiv preprint arXiv:2506.18871*, 2025.

[7] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *Mach. Intell. Res.*, 22(4):730–751, 2025.

[8] Xulu Zhang, Xiaoyong Wei, Wentao Hu, Jinlin Wu, Jiaxin Wu, Wengyu Zhang, Zhaoxiang Zhang, Zhen Lei, and Qing Li. A survey on personalized content synthesis with diffusion models. *Mach. Intell. Res.*, 22(5):817–848, 2025.

[9] Hailong Yan, Shice Liu, Tao Wang, Xiangtao Zhang, Yijie Zhong, Jinwei Chen, Le Zhang, and Bo Li. Animeagent: Is the multi-agent via image-to-video models a good disney storytelling artist? *arXiv preprint arXiv:2602.20664*, 2026.

[10] 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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, pages 26125–26135, 2025.

[11] Cong Wei, Zheyang Xiong, Weiming Ren, Xeron Du, Ge Zhang, and Wenhu Chen. Omniedit: Building image editing generalist models through specialist supervision. In *The Thirteenth International Conference on Learning Representations, ICLR*, 2025.

[12] Zeyue Xue, Jie Wu, Yu Gao, Fangyuan Kong, Lingting Zhu, Mengzhao Chen, Zhiheng Liu, Wei Liu, Qiushan Guo, Weilin Huang, and Ping Luo. Dancegrpo: Unleashing GRPO on visual generation. *arXiv preprint arXiv:2505.07818*, 2025.

[13] Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-grpo: Training flow matching models via online rl. *arXiv preprint arXiv:2505.05470*, 2025.

[14] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations, ICLR*, 2023.

[15] 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, ICLR*, 2023.

[16] Xiaofeng Tan, Jun Liu, Bin-Bin Gao, Yuanting Fan, Xi Jiang, Chengjie Wang, Hongsong Wang, and Feng Zheng. When policy entropy constraint fails: Preserving diversity in flow-based RLHF via perceptual entropy. *arXiv preprint arXiv:2605.12112*, 2026.

[17] Xiaofeng Tan, Jun Liu, Yuanting Fan, Bin-Bin Gao, Xi Jiang, Xiaochen Chen, Jinlong Peng, Chengjie Wang, Hongsong Wang, and Feng Zheng. Consistentrft: Reducing visual hallucinations in flow-based reinforcement fine-tuning. *arXiv preprint arXiv:2602.03425*, 2026.

[18] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *IEEE/CVF International Conference on Computer Vision, ICCV*, pages 4172–4182, 2023.

[19] Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, Sumith Kulal, Kyle Lacey, Yam Levi, Cheng Li, Dominik Lorenz, Jonas Müller, Dustin Podell, Robin Rombach, Harry Saini, Axel Sauer, and Luke Smith. FLUX.1 kontext: Flow matching for in-context image generation and editing in latent space. *arXiv preprint arXiv:2506.15742*, 2025.

[20] Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Shengming Yin, Shuai Bai, Xiao Xu, Yilei Chen, Yuxiang Chen, Zecheng Tang, Zekai Zhang, Zhengyi Wang, An Yang, Bowen Yu, Chen Cheng, Dayiheng Liu, Deqing Li, Hang Zhang, Hao Meng, Hu Wei, Jingyuan Ni, Kai Chen, Kuan Cao, Liang Peng, Lin Qu, Minggang Wu, Peng Wang, Shuting Yu, Tingkun Wen, Wensen Feng, Xiaoxiao Xu, Yi Wang, Yichang Zhang, Yongqiang Zhu, Yujia Wu, Yuxuan Cai, and Zenan Liu. Qwen-image technical report. *arXiv preprint arXiv:2508.02324*, 2025.

[21] Hanghang Ma, Haoxian Tan, Jiale Huang, Junqiang Wu, Jun-Yan He, Lishuai Gao, Songlin Xiao, Xiaoming Wei, Xiaoqi Ma, Xunliang Cai, Yayong Guan, and Jie Hu. Longcat-image technical report. *arXiv preprint arXiv:2512.07584*, 2025.

[22] Shiyu Liu, Yucheng Han, Peng Xing, Fukun Yin, Rui Wang, Wei Cheng, Jiaqi Liao, Yingming Wang, Honghao Fu, Chunrui Han, Guopeng Li, Yuang Peng, Quan Sun, Jingwei Wu, Yan Cai, Zheng Ge, Ranchen Ming, Lei Xia, Xianfang Zeng, Yibo Zhu, Binxing Jiao, Xiangyu Zhang, Gang Yu, and Daxin Jiang. Step1x-edit: A practical framework for general image editing. *arXiv preprint arXiv:2504.17761*, 2025.

[23] 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, SIGGRAPH*, pages 78:1–78:11, 2023.

[24] Yujun Shi, Chuhui Xue, Jun Hao Liew, Jiachun Pan, Hanshu Yan, Wenqing Zhang, Vincent Y. F. Tan, and Song Bai. Dragdiffusion: Harnessing diffusion models for interactive point-based image editing. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, pages 8839–8849, 2024.

[25] Xuanjia Zhao, Jian Guan, Congyi Fan, Dongli Xu, Youtian Lin, Haiwei Pan, and Pengming Feng. Fastdrag: Manipulate anything in one step. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS*, 2024.

[26] Jingyi Lu, Xinghui Li, and Kai Han. Regiondrag: Fast region-based image editing with diffusion models. In *The 18th European Conference on Computer Vision, ECCV*, pages 231–246, 2024.

[27] Siwei Xia, Li Sun, Tiantian Sun, and Qingli Li. Draglora: Online optimization of lora adapters for drag-based image editing in diffusion model. In *Forty-second International Conference on Machine Learning, ICML*, 2025.

[28] Haofeng Liu, Chenshu Xu, Yifei Yang, Lihua Zeng, and Shengfeng He. Drag your noise: Interactive point-based editing via diffusion semantic propagation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, pages 6743–6752, 2024.

[29] Pengyang Ling, Lin Chen, Pan Zhang, Huaian Chen, Yi Jin, and Jinjin Zheng. Freedrag: Feature dragging for reliable point-based image editing. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, pages 6860–6870, 2024.

[30] Zewei Zhang, Huan Liu, Jun Chen, and Xiangyu Xu. Gooddrag: Towards good practices for drag editing with diffusion models. In *The Thirteenth International Conference on Learning Representations, ICLR*, 2025.

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

[32] Ziqi Jiang, Zhen Wang, and Long Chen. Clipdrag: Combining text-based and drag-based instructions for image editing. In *The Thirteenth International Conference on Learning Representations, ICLR*, 2025.

[33] Yujun Shi, Jun Hao Liew, Hanshu Yan, Vincent Y. F. Tan, and Jiashi Feng. Lightningdrag: Lightning fast and accurate drag-based image editing emerging from videos. In *Forty-second International Conference on Machine Learning, ICML*, 2025.

[34] Jingyi Lu and Kai Han. Inpaint4drag: Repurposing inpainting models for drag-based image editing via bidirectional warping. In *International Conference on Computer Vision, ICCV*, 2025.

[35] Shen Nie, Hanzhong Allan Guo, Cheng Lu, Yuhao Zhou, Chenyu Zheng, and Chongxuan Li. The blessing of randomness: SDE beats ODE in general diffusion-based image editing. In *The Twelfth International Conference on Learning Representations, ICLR*, 2024.

[36] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit Haim Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion. In *The Eleventh International Conference on Learning Representations, ICLR*, 2023.

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

[38] Kunyu Feng, Yue Ma, Bingyuan Wang, Chenyang Qi, Haozhe Chen, Qifeng Chen, and Zeyu Wang. Dit4edit: Diffusion transformer for image editing. In *Thirty-Ninth AAAI Conference on Artificial Intelligence, Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence, Fifteenth Symposium on Educational Advances in Artificial Intelligence, AAAI*, pages 2969–2977, 2025.

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

[40] Changhao Qiao, Chao Hui, Chen Li, Cunzheng Wang, Dejia Song, Jiale Zhang, Jing Li, Qiang Xiang, Runqi Wang, Shuang Sun, Wei Zhu, Xu Tang, Yao Hu, Yibo Chen, Yuhao Huang, Yuxuan Duan, Zhiyi Chen, and Ziyuan Guo. Firered-image-edit-1.0 technical report. *arXiv preprint arXiv:2602.13344*, 2026.

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

[42] Yutao Cui, Xiaotong Zhao, Guozhen Zhang, Shengming Cao, Kai Ma, and Limin Wang. Stabledrag: Stable dragging for point-based image editing. In *The 18th European Conference on Computer Vision, ECCV*, pages 340–356, 2024.

[43] Gayoon Choi, Taejin Jeong, Sujung Hong, and Seong Jae Hwang. Dragtext: Rethinking text embedding in point-based image editing. In *IEEE/CVF Winter Conference on Applications of Computer Vision, WACV*, pages 441–450, 2025.

[44] Zixin Yin, Xili Dai, Duomin Wang, Xianfang Zeng, Lionel M. Ni, Gang Yu, and Heung-Yeung Shum. Lazydrag: Enabling stable drag-based editing on multi-modal diffusion transformers via explicit correspondence. *arXiv preprint arXiv:2509.12203*, 2025.

[45] 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.

[46] Huiguo He, Pengyu Yan, Ziqi Yi, Weizhi Zhong, Zheng Liu, Yejun Tang, Huan Yang, Guanbin Li, and Lianwen Jin. Contextdrag: Precise drag-based image editing via context-preserving token injection and position-aligned attention. *arXiv preprint arXiv:2512.08477*, 2026.

[47] Haeil Lee, Hansang Lee, Seoyeon Gye, and Junmo Kim. Beta sampling is all you need: Efficient image generation strategy for diffusion models using stepwise spectral analysis. In *IEEE/CVF Winter Conference on Applications of Computer Vision, WACV*, pages 4215–4224, 2025.

[48] Jooyoung Choi, Jungbeom Lee, Chaehun Shin, Sungwon Kim, Hyunwoo Kim, and Sungroh Yoon. Perception prioritized training of diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, pages 11462–11471, 2022.

[49] 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.

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

[51] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *The 16th European Conference on Computer Vision, ECCV*, 2020.

[52] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023.

[53] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Ming-Hsuan Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

[54] 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. In *The Thirteenth International Conference on Learning Representations, ICLR*, 2025.

[55] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloé Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross B. Girshick. Segment anything. In *IEEE/CVF International Conference on Computer Vision, ICCV*, pages 3992–4003, 2023.

A Broader Impacts

MoRe-Drag is designed to improve the controllability and usability of interactive image editing. Such capability can benefit creative workflows, visual design, image retouching, education, and AR/VR content creation. Our method reduces the need for manual trial-and-error tuning, enabling more accurate and natural manipulation. However, more controllable and realistic editing tools may also increase the risk of misuse. Precise drag-based manipulation raises potential misuse concerns, including the alteration of visual evidence, the creation of misleading media, and non-consensual edits of personal images. These risks are not unique to our method; however, increased spatial controllability may reduce the effort needed to produce convincing deceptive content. Responsible deployment should therefore incorporate safeguards such as explicit user consent, provenance tracking, watermarking, and authentication of edited images.

B Drag-Instruction Dataset Construction

Below, we describe in detail the dataset construction pipeline mentioned in Sec. 3.3. BYTEMORPH[4] [49] covers several editing types, including camera zoom, camera motion, object motion, human motion, and human-object interaction. Since our method focuses on drag-based editing, we design an automated data processing pipeline to extract samples with reliable local motion, mine dense drag correspondences, filter noisy annotations, and construct spatial candidate regions. The overall data construction pipeline is summarized in Table 5. Starting from BYTEMORPH, the motion filtering stage retains approximately 220K samples, and the final score-based filtering produces approximately 68K high-quality drag-instruction samples.

(i) Motion-edit filtering. We first filter BYTEMORPH to keep only samples involving local motion edits. Specifically, we use Qwen3.5-9B [50] as a structured instruction parser. Given an editing instruction, the model determines whether the instruction is motion-related, whether it involves global camera changes, the motion type, and the entities being edited. We retain a sample only if it describes human or object motion, and discard samples involving global camera changes. The extracted edited entities are further used in the subsequent step to obtain masks for the editing subjects. This stage retains approximately 220K motion-related samples from the original dataset.

(ii) Drag annotation mining. For each retained sample, we automatically extract drag annotations from the source–target image pair. We first localize the edited entities obtained from the instruction parser. Grounding DINO [52] is used to detect entity boxes in the source image, and SAM [55] converts the detected boxes into pixel-level masks. We then compute bidirectional optical flow between the source and target images using RAFT [51]. A pixel is considered a valid motion candidate if the magnitudes of both its forward flow and the corresponding backward flow exceed 1.5 pixels.

The candidate motion mask is defined as the intersection of the entity mask and the valid motion mask. We remove a 4-pixel image border and discard connected components with fewer than 64 pixels. Large components are further partitioned by applying k-means clustering to their flow vectors, which separates regions with different motion directions. We retain at most 8 motion regions per sample. Within each region, handle points are sampled on a stratified grid, at a density of approximately one point per 16×16 pixels. The number of handle points is capped at 8 per region and 64 per sample. For each handle point (x_h, y_h), the corresponding target point is obtained by adding the forward optical-flow vector:

$$(x_t, y_t) = (x_h + \Delta x, y_h + \Delta y).$$

We discard target points outside the image boundary and remove samples that contain no valid drag points.

(iii) Drag quality scoring and filtering. The automatically mined drag annotations may still contain noisy correspondences or motion cues that are not semantically aligned with the intended edit. We therefore score the annotations with a VLM, using human inspection to calibrate the scoring prompt and spot-check the retained samples. For each sample, Qwen3.5-9B [50] is given three images: the source image, the source image overlaid with drag guidance, and the target image. The drag visualization marks handle points in red, target points in blue, and drag directions with arrows.

The VLM scores each sample along three dimensions: camera motion, instruction alignment, and target alignment. The camera-motion score penalizes global camera changes in the source–target pair; the instruction-alignment score measures whether the drag guidance matches the editing instruction; and the target-alignment score measures whether the drag guidance explains the visible transformation from the source image to the target image. Each score is an integer in $\{0, 1, 2, 3\}$, with higher values indicating better quality. The final score is the normalized average:

$$S_{\text{avg}} = \frac{1}{3}\left(\frac{S_{\text{cam}}}{3} + \frac{S_{\text{ins}}}{3} + \frac{S_{\text{tgt}}}{3}\right).$$

We discard samples with $S_{\text{avg}} < 0.6$ and remove obvious failure cases through human spot-checking. This score-based filtering step yields approximately 68K high-quality drag-annotated samples. The prompt used for VLM-based drag quality scoring is shown in Box G.2.

[4]We use the released demo version, ByteMorph-6M-Demo: https://huggingface.co/datasets/ByteMorph/BM-6M-Demo.

Table 5: Summary of the drag-instruction dataset construction pipeline.

Stage	Main models	Key settings	Size
Raw dataset	-	-	780K
Motion-edit filtering	Qwen3.5-9B	local human/object motion; no global camera change	220K
Drag annotation mining	Grounding DINO, SAM, RAFT	flow threshold 1.5	220K
Quality scoring and filtering	Qwen3.5-9B + Human	alignment scores in $\{0, 1, 2, 3\}$; keep $S_{\text{avg}} \geq 0.6$	68K

C Implementation, Evaluation, and Training Details

C.1 Implementation Details

We implement *MoRe-Drag* on LongCat-Image-Edit [21] and Qwen-Image-Edit [20]. Both models adopt Qwen2.5-VL-7B [53] as the MLLM-based text encoder, while using MMDiT generators with 6B and 20B parameters, respectively. All evaluation experiments are conducted on NVIDIA RTX 4090 GPUs. Unless otherwise specified, we use 30 sampling steps. For DRAGBENCH-SR [35], we use a CFG scale of 3, apply region-aware latent recomposition from step 1 to step 16, and switch generation-stage adaptive conditioning at step 12. For DRAGBENCH-DR, we use a CFG scale of 2, apply region-aware latent recomposition from step 1 to step 14, and switch generation-stage adaptive conditioning at step 11. For the recomposition weight, we set $\lambda_{\max} = 0.5$ and $\lambda_{\min} = 0.3$.

C.2 Training Details

For drag-aware instruction inference, we fine-tune the Qwen2.5-VL-7B text encoder on our filtered 68K drag-instruction dataset. The fine-tuning is performed with LoRA on two NVIDIA RTX PRO 6000 GPUs for 15 GPU-hours. We freeze the vision tower, LLM backbone, and multimodal merger, and train LoRA adapters with rank 32, alpha 64, and dropout 0.05. The model is trained for 3 epochs using a global batch size of 80, with a per-device batch size of 40 and no gradient accumulation. We use bfloat16 precision and DeepSpeed ZeRO-2 setting. The learning rate is set to 2×10^{-4} with weight decay 0.1, a warmup ratio of 0.03, and a cosine learning-rate schedule. The input image resolution is controlled by setting the minimum and maximum visual tokens to $256 \times 28 \times 28$ and $1280 \times 28 \times 28$, respectively.

C.3 Formulation of Weight Schedule

We provide the formulations of the weight schedules used in latent recomposition (Sec. 3.1). Unless otherwise specified, we adopt the constant schedule as the default setting.

$$\begin{aligned} \textit{Constant}: \ \lambda(t_i) = \lambda_{\max}, \quad \textit{Linear}: \ \lambda(t_i) = \lambda_{\max} + (\lambda_{\min} - \lambda_{\max})\rho_i, \\ \textit{Inverse square}: \ \lambda(t_i) = \lambda_{\max} + (\lambda_{\min} - \lambda_{\max})\rho_i^2. \end{aligned} \tag{12}$$

C.4 Details of Bidirectional Warping

We adopt the bidirectional pixel-space warping strategy from Inpaint4Drag [34] to construct the warped observation used as motion evidence. Given a source image $\boldsymbol{X}$, a manipulation mask $\boldsymbol{M}_s$, and handle–target pairs $\{(\boldsymbol{h}_i, \boldsymbol{t}_i)\}_{i=1}^{K}$, the goal is to propagate sparse user-specified displacements to dense pixel motions within the editable region. For each handle point, the displacement vector is defined as

$$\boldsymbol{d}_i = \boldsymbol{t}_i - \boldsymbol{h}_i. \tag{13}$$

A dense motion field $\boldsymbol{F}(\boldsymbol{x})$ is then estimated by interpolating the sparse displacements over pixels $\boldsymbol{x} \in \boldsymbol{M}_s$, with nearby handles contributing more strongly:

$$\boldsymbol{F}(\boldsymbol{x}) = \sum_{i=1}^{K} \frac{w_i(\boldsymbol{x})\boldsymbol{d}_i}{\sum_{i=1}^{K} w_i(\boldsymbol{x})}, \qquad w_i(\boldsymbol{x}) = \frac{1}{\|\boldsymbol{x} - \boldsymbol{h}_i\| + \epsilon}. \tag{14}$$

Pixels outside the manipulation mask are kept unchanged.

Following Inpaint4Drag [34], we apply bidirectional warping to better preserve transported content while identifying disoccluded regions. Forward warping maps each source pixel to its displaced

location:

$$\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{F}(\boldsymbol{x}), \qquad \boldsymbol{x} \in \boldsymbol{M}_s. \tag{15}$$

Since forward warping may create holes or overlaps, we further construct a backward mapping for pixels in the transformed target region. For each target pixel $\boldsymbol{x}'$, we find its N nearest pixels $\{\boldsymbol{x}'_i\}_{i=1}^{N}$ that have valid correspondences from the forward warping step, where $\boldsymbol{x}_i$ denotes the corresponding source position of $\boldsymbol{x}'_i$. The source position of $\boldsymbol{x}'$ is then estimated by locally interpolating the inverse displacement:

$$\boldsymbol{x}_s(\boldsymbol{x}') = \boldsymbol{x}' + \sum_{i=1}^{N} w_i(\boldsymbol{x}') \left(\boldsymbol{x}_i - \boldsymbol{x}'_i\right), \tag{16}$$

where $w_i(\boldsymbol{x}')$ are normalized inverse-distance weights computed from the distance between $\boldsymbol{x}'$ and $\boldsymbol{x}'_i$. We only keep valid mapping pairs whose source and target coordinates lie inside the image domain,

$$\boldsymbol{x}_s(\boldsymbol{x}'), \boldsymbol{x}' \in [0, W) \times [0, H), \tag{17}$$

where W and H denote the image width and height. The warped image is then obtained by

$$\boldsymbol{X}_w(\boldsymbol{x}') = \boldsymbol{X}(\boldsymbol{x}_s(\boldsymbol{x}')). \tag{18}$$

The resulting warped image $\boldsymbol{X}_w$ provides a coarse observation of the desired drag motion.

Fig. 10 visualizes the intermediate representation derived from drag annotations. The user-specified drag input is converted into a warped observation that serves as pixel-space motion evidence. Based on this observation, we construct three functional masks for region-aware latent recomposition: a preservation mask for unchanged content, a refinement mask for transported content, and an inpainting mask for invalid regions.

This warping result is not used as the intermediate edited image. Instead, *MoRe-Drag* treats $\boldsymbol{X}_w$ as coarse motion evidence that provides drag-relevant cues about where and how the content should move. The subsequent region-aware recomposition and adaptive conditioning stages repair artifacts introduced by warping and synthesize visually plausible content. For additional implementation details of the bidirectional warping procedure, we refer readers to Inpaint4Drag [34].

C.5 Benchmark Instruction Generation

DragBench-SR [35] and DragBench-DR [24] provide source images, manipulation masks, and handle–target point pairs, but do not include the natural-language editing instructions needed for our instruction-based evaluation metrics. We therefore generate one edit prompt for each sample with a vision-language model. For each sample, we pair the original source image with an annotated version in which handle points are marked as red squares, target points as blue circles, and handle-to-target displacements as yellow arrows. We feed this two-image input to Qwen3.5-9B [50], together with a structured intent-parsing prompt (Box G.3), to obtain a concise instruction that captures the semantic intent of the drag operation. The generated instructions are manually checked and corrected when necessary. The final instruction is written back to the sample metadata for instruction-based evaluation, while the original drag points and masks are kept for drag-based editing and metric computation.

C.6 Evaluation Metrics

Following DragDiffusion [24], we use Mean Distance (MD) to evaluate drag precision. MD measures the average Euclidean distance between the tracked handle points in the edited image and the user-specified target points; lower values indicate more accurate drag control. Specifically, for each handle–target pair, we track the handle location after editing, compute its distance to the target point, and average the distances over all point pairs and samples.

We do not report Image Fidelity (IF), as prior work has noted that it is not well suited for evaluating identity preservation in drag-based editing [34, 44]. For drag-based editing, valid geometric transformations may inevitably increase LPIPS despite preserving the intended identity, making IF an unreliable proxy for fidelity in this setting. We therefore adopt the vision-language-model-based metrics from DreamBench++ [54], namely Concept Preservation (CP) and Prompt Following (PF), which are designed to better capture semantic consistency and instruction alignment.

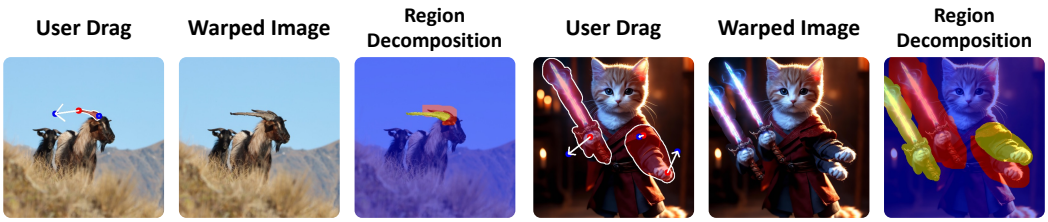

Figure 10: Illustration of bidirectional warping and mask decomposition. Given handle–target drag annotations, bidirectional warping produces a warped observation that provides pixel-space motion evidence. We decompose the result into preservation, refinement, and inpainting regions for region-aware latent recomposition: blue denotes the anchor preservation $\boldsymbol{M}_p$, yellow denotes the refinement mask $\boldsymbol{M}_r$, and red denotes the inpainting mask $\boldsymbol{M}_i$.

Algorithm 1 *MoRe-Drag*

Require: Source image $\boldsymbol{X}_o$, warped image $\boldsymbol{X}_w$, drag masks $(\boldsymbol{m}_r, \boldsymbol{m}_i, \boldsymbol{m}_p)$, instruction p, fixed restoration prompt p_{fix}, recomposition interval $\mathcal{T}$, switching timestep t_c.

Encode source and warped images: $\boldsymbol{z}_0^o = \mathcal{E}(\boldsymbol{X}_o)$, $\boldsymbol{z}_0^w = E(\boldsymbol{X}_w)$.
Initialize generation latent $\boldsymbol{z}_1^g \sim \mathcal{N}(0, \boldsymbol{I})$ and evidence latent $\boldsymbol{z}_t^w$ from the flow perturbation of $\boldsymbol{z}_0^w$.
for each timestep $t : 1 \to 0$ **do**
  if $t \geq t_c$ **then**
    Use warped-image condition $(\boldsymbol{c}(t), \boldsymbol{z}_0^{\text{img}}(t)) = (\Phi_{\text{mllm}}(\boldsymbol{X}_w, p_{\text{fix}}), \mathcal{E}(\boldsymbol{X}_w))$.
  else
    Use original-image condition $(\boldsymbol{c}(t), \boldsymbol{z}_0^{\text{img}}(t)) = (\Phi_{\text{mllm}}(\boldsymbol{X}_o, p), \mathcal{E}(\boldsymbol{X}_o))$.
  end if
  Update the generation trajectory with one MMDiT forward pass:
  $\boldsymbol{z}_{t-\Delta t}^g \leftarrow \text{Step}(\boldsymbol{z}_t^g, \boldsymbol{v}_\theta(\boldsymbol{z}_t^g, t, \boldsymbol{c}(t), \boldsymbol{z}_0^{\text{img}}(t)))$.
  if $t \in \mathcal{T}$ **then**
    Update the evidence trajectory with a null-condition MMDiT forward pass:
    $\boldsymbol{z}_{t-\Delta t}^w \leftarrow \text{Step}(\boldsymbol{z}_t^w, \boldsymbol{v}_\theta(\boldsymbol{z}_t^w, t, \boldsymbol{c}_\emptyset, \boldsymbol{z}_0^w))$.
    Recompose the generation latent:
    $\boldsymbol{z}_{t-\Delta t}^g \leftarrow \boldsymbol{m}_i \odot \boldsymbol{z}_{t-\Delta t}^g + \boldsymbol{m}_p \odot \boldsymbol{z}_{t-\Delta t}^w + \boldsymbol{m}_r \odot (\lambda \boldsymbol{z}_{t-\Delta t}^g + (1-\lambda)\boldsymbol{z}_{t-\Delta t}^w)$.
  end if
end for
Decode $\boldsymbol{z}_0^g$ to obtain the edited image.

Following DreamBench++, we use GPT-4o as the evaluator for both CP and PF. CP is computed under the original DreamBench++ setting to assess whether the edited image preserves the main visual concept and identity of the source content. For PF, since our task is image editing rather than text-to-image generation, we adapt the original prompt-following protocol by providing the evaluator with a triplet consisting of the source image, the edited image, and the corresponding edit instruction. The evaluator is asked to determine whether the edited image follows the instruction while preserving visual content that should remain unchanged. The PF score ranges from 0 to 4, with higher values indicating better instruction alignment and content preservation. The prompt used for PF evaluation is shown in Box G.4.

D Algorithmic Details of MoRe-Drag

Algorithm 1 summarizes the inference procedure of *MoRe-Drag*. The input instruction p can be either provided by the user or automatically inferred from drag annotations using the fine-tuned MLLM text encoder of the editor (Sec. 3.3). The method maintains two latent trajectories during sampling: an instruction-conditioned generation trajectory and a null-condition evidence reconstruction trajectory. The generation trajectory is updated at every sampling step using the adaptive image-text condition (Sec. 3.2). The evidence trajectory is updated only within the recomposition interval $\mathcal{T} = [t_s, t_e]$, where the warped observation is injected as motion evidence (Sec. 3.1).

E Extra Results

E.1 Quantitative Results

Table 6: Comparison with RegionDrag and DragFlow on DRAGBENCH-SR/DR. CP and PF are scaled by 100. *MoRe-Drag* uses LongCat-Image-Edit as the base editor.

Model	DRAGBENCH-SR			DRAGBENCH-DR		
	MD↓	CP↑	PF↑	MD↓	CP↑	PF↑
RegionDrag [26]	22.73	77.25	65.75	26.36	76.10	63.05
DragFlow [45]	36.87	76.75	60.00	40.59†	75.49	65.98
MoRe-Drag	**18.54**	**96.36**	**86.36**	**20.03**	**93.15**	**84.82**

† *The original DragFlow paper reports an MD of 31.59 on* DRAGBENCH-DR.

We further compare *MoRe-Drag* with RegionDrag [26] and DragFlow [45] using their released implementations, where the target-region centroids required by DragFlow are derived from the benchmark annotations. As shown in Table 6, *MoRe-Drag* outperforms both methods on all metrics, and its MD remains lower than that reported in the DragFlow paper.

E.2 Qualitative Results

Fig. 11 presents additional qualitative comparisons across diverse drag-based editing scenarios, including object translation, pose adjustment, and human/object motion. Existing drag-based methods often suffer from a precision–naturalness trade-off. Optimization-based methods, such as DragDiffusion [24], DragNoise [28], and GoodDrag [30], lack a sufficiently strong mechanism for motion injection, which limits their ability to perform high-precision edits. Optimization-free methods also exhibit clear limitations: FastDrag [25] performs warping in the latent space followed by generative refinement, which can lead to detail loss and inaccurate edits, as shown in Rows 3 and 5 of Fig. 11. Inpaint4Drag [34] uses the warped image as an intermediate result and can achieve relatively accurate motion in some cases, but it cannot avoid distortions and unnatural artifacts introduced by warping, as observed in Rows 1, 3, 5, and 8 of Fig. 11. In contrast, *MoRe-Drag* uses the warped image as motion evidence to provide deformation constraints for the editing model, enabling edits that are both accurate and natural. For example, in Row 3 of Fig. 11, where the crocodile's mouth is opened, *MoRe-Drag* satisfies the drag constraints while generating realistic inner-mouth content. In Row 8, *MoRe-Drag* successfully adjusts Iron Man's pose while preserving a more natural and plausible arm structure, whereas Inpaint4Drag produces visible arm distortions. These examples further demonstrate the effectiveness of our method.

E.3 Comparison with Design Alternatives

Table 8 compares *MoRe-Drag* with simpler alternatives. To check whether RAR is needed, we replace it with global blending, which keeps the same two branches and blending weight but mixes the whole latent without role masks. This variant is even worse than removing RAR, showing that the benefit of RAR comes from handling each region separately.

We then examine the evidence branch. One alternative re-noises the warped latent to the current timestep, $\bar{z}_t^w = (1-t)\,\mathcal{E}(\boldsymbol{X}_w) + t\,\boldsymbol{\epsilon}$, so it matches the timestep but is never updated by the editor. Another drops the evidence branch and lets the base editor restore the warped image directly. Re-noising degrades all metrics, and direct restoration keeps drag accuracy but changes the source identity and appearance, while the model-evolved evidence branch balances both.

E.4 Robustness to Input Masks

MoRe-Drag takes handle–target points and a coarse user mask as input, and does not require a pixel-accurate mask. To test its sensitivity to the mask, we perturb each deformation mask of DRAGBENCH-SR before warping by erosion, dilation, and removing a contiguous part, each at a mild and a strong level. The warped image and all role masks are then recomputed from the perturbed mask. We also test automatic masks predicted by SAM [55] with the handle points as prompts. As

Table 7: Hyperparameter analysis on DRAGBENCH-SR.

Setting (RAR steps, GAC switch)	DRAGBENCH-SR		
	MD↓	CP↑	PF↑
(6, 2)	$37.00_{\pm 0.48}$	$92.75_{\pm 0.00}$	$69.72_{\pm 0.28}$
(11, 9)	$21.09_{\pm 0.20}$	$95.75_{\pm 0.25}$	$80.63_{\pm 0.38}$
(16, 12)	$18.54_{\pm 0.22}$	$96.36_{\pm 0.38}$	$86.36_{\pm 4.06}$
(21, 14)	$18.43_{\pm 0.31}$	$95.87_{\pm 0.88}$	$83.50_{\pm 2.36}$
(30, 20)	$17.98_{\pm 0.37}$	$94.25_{\pm 1.00}$	$80.75_{\pm 1.61}$

Table 8: Comparison with design alternatives.

Benchmark	**Setting**	**MD↓**	**CP↑**	**PF↑**
DRAGBENCH-DR	w/o RAR	25.41	92.50	77.14
	Global blending	33.50	83.92	77.55
	RAR (ours)	**20.03**	**93.15**	**84.82**
DRAGBENCH-SR	Direct re-noising	19.01	94.68	82.91
	Direct restoration	**18.06**	92.28	81.06
	MoRe-Drag	18.54	**96.36**	**86.36**

shown in Table 9, mild perturbations only cause small drops. Erosion and missing regions mainly hurt MD, since part of the motion evidence is lost. Strong dilation keeps MD but lowers CP and PF, since unrelated content is also moved. SAM masks are often coarse or only cover a local part of the object, which leads to a clear drop. We therefore use the coarse user mask by default and treat automatic masks as an optional mode.

E.5 Effect of Disocclusion Severity

Large disoccluded regions make the warped image less informative. We measure this with the disocclusion ratio $r_{\text{dis}} = |\boldsymbol{M}_i|/|\boldsymbol{M}_r \cup \boldsymbol{M}_i|$, and split DRAGBENCH-SR into three groups of similar size. As shown in Table 10, the high-disocclusion group has a clearly worse MD, while CP and PF remain stable. The low and medium groups are not strictly ordered, since r_{dis} only partly reflects the difficulty and the edit type also matters. Thus, severe disocclusion mainly hurts spatial accuracy, which is consistent with the failure cases in Appendix E.8.

E.6 Effect of Instruction Interaction

We provide additional examples of drag-aware instruction inference in Fig. 12. Each column shows one drag-annotated input together with the instructions inferred by the vanilla MLLM and the LoRA-adapted MLLM. Compared with the vanilla model, the LoRA-adapted model more accurately identifies the edited subject and converts the drag geometry into an appropriate editing intent. Table 11 quantifies the effect of drag-aware instruction inference on DRAGBENCH-SR. Compared with the vanilla MLLM, the LoRA-adapted model improves drag accuracy, reducing MD from 19.94 to 18.54, while also increasing CP and PF. These results indicate that LoRA adaptation produces instructions that are better aligned with the drag annotations and more beneficial for downstream editing.

The prompt template used for instruction inference is shown in Box G.1. During inference, the model receives the source image and its drag-annotated version, and outputs a natural-language instruction describing the intended edit. The same prompt is used for both the vanilla MLLM and the LoRA-adapted MLLM.

Effect of instruction detail. We investigate whether more detailed text improves editing performance. On 25 samples, we compare the instruction inferred by our LoRA-adapted MLLM with a detailed instruction generated by GPT-5 and a fixed restoration instruction. As shown in Table 12, detailed instructions do not improve drag accuracy over the inferred instructions, and a fixed restoration instruction also retains comparable spatial accuracy. The inferred instructions better preserve the source concept than the fixed restoration instruction, with stronger prompt following. These

Table 9: Robustness to perturbed and automatically predicted masks on DRAGBENCH-SR. IoU is computed with respect to the original mask.

Input mask	Level	IoU	MD↓	CP↑	PF↑
Original	–	1.000	18.54	96.36	86.36
Erosion	Mild	0.818	21.11	95.80	84.64
	Strong	0.683	24.17	92.73	85.00
Dilation	Mild	0.802	18.30	94.68	85.25
	Strong	0.600	18.45	92.45	72.92
Missing region	Mild	0.804	21.67	92.45	85.25
	Strong	0.620	30.60	91.33	82.07
SAM (point prompt)	–	0.361	24.91	83.51	70.62

Table 10: Results under different disocclusion severities on DRAGBENCH-SR.

Severity	r_{dis}	# Samples	MD↓	CP↑	PF↑
Low	$[0, 0.38]$	33	17.56	96.21	86.36
Medium	$[0.38, 0.53]$	34	15.26	96.97	84.38
High	$[0.53, 0.75]$	33	23.77	96.88	89.84

results support using a short inferred instruction to guide semantic restoration, while motion evidence provides spatial control.

E.7 Hyperparameter Analysis

We analyze two related hyperparameters on DRAGBENCH-SR: the number of Region-Aware Recomposition (RAR) steps and the switching step of Generation-Stage Adaptive Conditioning (GAC). Although they are defined separately, they play a similar role in practice: increasing the RAR steps injects motion evidence for a longer interval, while delaying the GAC switch keeps the generation branch conditioned on the warped observation for more steps. Both therefore strengthen the influence of motion evidence during sampling. For this reason, we vary them synchronously and evaluate five paired settings, as shown in Table 7.

As the RAR steps and GAC switching step increase, MD consistently decreases, indicating that stronger motion-evidence grounding improves drag precision. However, after the $(16, 12)$ setting, the reduction in MD becomes marginal, while CP and PF show slight declines. This suggests that the injected motion evidence is already sufficient at $(16, 12)$, and further increasing its influence brings limited gains in drag accuracy while slightly weakening semantic consistency. We therefore use $(16, 12)$ as the default setting.

We further isolate the effect of the GAC switching step by fixing the number of RAR steps to 16. The quantitative results are reported in Table 13. When the switch occurs too early, motion evidence is removed before the spatial layout is sufficiently anchored, leading to degraded drag accuracy. Conversely, keeping the warped-observation conditioning for too long leaves insufficient sampling steps for instruction-guided refinement, often producing less natural results and prompt following. This reveals a trade-off between motion grounding and semantic refinement, and supports our choice of an intermediate switching step.

E.8 Failure Cases

We present two representative failure cases in Fig. 13, where inaccurate warped observations weaken the usefulness of motion evidence. Our method is designed to incorporate motion evidence into the editing generation process, and it works reliably when the warped observation provides a reasonably consistent indication of the desired motion. However, in these cases, the motion evidence becomes ambiguous and misleading. For example, in the cat case, the drag point is insufficiently constrained for the intended rotation, which would typically require an additional anchor point; in the toilet case, the warped image does not provide a clear geometry for closing the lid. As a result, the editor tends to treat the corrupted and ambiguous regions as content to be restored rather than reliable evidence

Table 11: Quantitative analysis of instruction interaction on DRAGBENCH-SR.

	MD↓	CP↑	PF↑
Vanilla	$19.94_{\pm 0.27}$	$95.12_{\pm 0.13}$	$78.63_{\pm 1.38}$
w/ LoRA	$18.54_{\pm 0.22}$	$96.36_{\pm 0.38}$	$86.36_{\pm 4.06}$

Table 12: Effect of instruction detail on 25 samples from DRAGBENCH-SR.

Text condition	**MD↓**	**CP↑**	**PF↑**
Auto-inferred drag-aware instruction	17.30	98.00	85.72
Detailed GPT-5-generated instruction	17.67	95.00	86.00
Fixed restoration instruction	16.97	91.00	82.00

Table 13: Effect of switching timestep t_c on DRAGBENCH-SR.

Step k	4	8	16	20	24	28
t_c	0.87	0.73	0.47	0.33	0.20	0.07
MD ↓	$21.01_{\pm 0.1}$	$19.27_{\pm 0.25}$	$18.59_{\pm 0.32}$	$17.98_{\pm 0.09}$	$17.52_{\pm 0.24}$	$17.36_{\pm 0.16}$
CP ↑	$96.09_{\pm 0.63}$	$97.10_{\pm 0.13}$	$97.47_{\pm 0.25}$	$97.10_{\pm 0.13}$	$95.96_{\pm 0.50}$	$96.46_{\pm 0.26}$
PF ↑	$83.33_{\pm 2.68}$	$85.10_{\pm 1.99}$	$80.08_{\pm 1.29}$	$81.06_{\pm 0.51}$	$79.31_{\pm 0.74}$	$80.94_{\pm 0.13}$

of the intended motion, leading to edits that remain visually plausible but only partially follow the target articulation. More generally, the warping in *MoRe-Drag* is purely 2D. It does not know which part is in front, which part becomes hidden, or what a newly exposed surface should look like. Edits with large disoccluded regions or strong 3D motion, such as rotating an object, are therefore harder. Future work may build the motion evidence with depth and visibility information.

Another potential issue is that nearby handles with opposite displacements can partially cancel during motion interpolation, weakening the intended deformation. Region decomposition remains well defined, but cannot recover motion information lost in this upstream step.

Fig. 14 shows failure cases of drag-aware instruction inference. The LoRA-adapted MLLM may infer imperfect instructions when drag annotations are spatially too close, particularly for fine-grained shape changes, since the annotations may occlude local image content and provide incomplete visual evidence. As a result, subtle shape deformation may be misinterpreted as object motion or orientation change. Nevertheless, the injected motion evidence can partially compensate for suboptimal textual guidance by directly constraining the edited region. As shown in the tram example, *MoRe-Drag* still produces a motion-faithful result despite the imperfect inferred instruction. Future work may investigate alternative representations of drag information for MLLMs to further improve robustness.

E.9 Comparison of Base Editor Reconstruction Ability

MoRe-Drag relies on the base editor's null-conditioned reconstruction ability, since the injected motion evidence should guide the intended deformation while the editor is expected to faithfully preserve content outside the edited regions. We therefore compare the reconstruction behavior of different base editors under null conditions. Fig. 15 shows qualitative examples, and Table 14 reports reconstruction LPIPS and DINO similarity together with downstream editing performance on DRAGBENCH-SR. LongCat-Image-Edit [21] reconstructs the source images more faithfully than Qwen-Image-Edit [20] and achieves stronger editing results despite its smaller generator. This suggests that reconstruction stability contributes to the performance difference between the two variants.

We further examine this relationship within a single editor by grouping 205 samples from DRAGBENCH-DR according to the DINO similarity between each source image and its null-conditioned reconstruction by LongCat-Image-Edit. Table 15 reports the downstream editing results for each group. The low-quality group has the worst drag accuracy and concept preservation, but the high and medium groups are not consistently ordered. The low-quality group also contains few samples. Severe reconstruction errors can therefore indicate a risk of editing failure, while reconstruction similarity alone does not determine editing quality. These observations support the use

Table 14: Null-conditioned reconstruction quality of the two base editors and downstream editing results of their *MoRe-Drag* variants on DRAGBENCH-SR.

Base editor	Recon. LPIPS↓	DINO Sim.↑	MD↓	CP↑	PF↑
Qwen-Image-Edit	0.1533	0.9151	23.85	90.15	70.96
LongCat-Image-Edit	0.1373	0.9466	18.54	96.36	86.36

Table 15: Downstream editing results on 205 DRAGBENCH-DR samples grouped by LongCat-Image-Edit's null-conditioned reconstruction DINO similarity.

Quality	DINO similarity	# Samples	MD↓	CP↑	PF↑
High	$s \geq 0.90$	176	20.31	94.17	84.96
Medium	$0.80 \leq s < 0.90$	23	16.68	89.22	83.11
Low	$s < 0.80$	6	24.76	78.17	87.28

of a base editor with stable reconstruction, without assuming that reconstruction quality fully predicts downstream performance.

E.10 Runtime and Memory Analysis

We further report the computational cost of *MoRe-Drag*. All measurements are conducted on an NVIDIA RTX 4090 GPU with CPU offloading enabled. We randomly select 10 samples and report the average generation time per image and the peak GPU memory during inference. The default setting uses 30 sampling steps, applies region-aware latent recomposition from step 1 to step 16, and therefore introduces an additional evidence-branch forward pass for approximately half of the sampling trajectory.

As shown in Table 16, *MoRe-Drag* increases the average generation time from 30.74s to 41.31s. This overhead is expected, since the evidence reconstruction branch requires extra MMDiT forward passes during the RAR interval. Under our default setting, RAR is applied for 16 out of 30 sampling steps, and the observed runtime increase is roughly consistent with the additional forward computation. In contrast, the peak GPU memory only increases from 16.05 GB to 17.18 GB. These results indicate that *MoRe-Drag* improves drag editing performance with a moderate runtime overhead and limited additional memory usage.

F Declaration of LLM Usage

We use LLMs/MLLMs in data construction and evaluation. For data construction, we use Qwen3.5-9B [50] to filter camera-centric samples, parse motion-centric editing instructions, and generate natural-language instructions for DRAGBENCH-SR and DRAGBENCH-DR. For semantic evaluation, we follow the LLM-based evaluation protocol of DreamBench++ [54] and use GPT-4o to compute the PF score and CP score. We also use GPT-5 to generate detailed instructions for the ablation on instruction detail.

G Prompts Used in Experiments

Box G.1: Prompt template used for drag-aware instruction inference

System Prompt. You are a helpful assistant for drag-based image editing. Given (1) a source image and (2) an image overlaid with drag annotations, infer the intended edit and output a natural-language editing instruction.
Annotation Meaning. A red square denotes the handle point, a blue circle denotes the target point, and a yellow arrow denotes the drag direction from the handle point to the target point.
Output Requirement. Output only the editing instruction.

Table 16: Runtime and memory comparison. We report the average generation time per image and peak GPU memory over 10 samples.

Method	Sampling steps	Avg. time / image (s)	Peak GPU memory (GB)
Baseline editor	30	30.74	16.05
MoRe-Drag	30	41.31	17.18

Box G.2: Prompt used for VLM-based drag-annotation quality scoring

Task Definition. You are a strict judge for mined drag-annotation quality in an image-editing dataset. You will be given three images in this exact order:

1. Source image before editing.
2. Source image with drag guidance drawn on it.
3. Target image after editing.

Use the images as primary evidence. Use the instruction as supporting evidence. Be conservative: if the evidence is weak, ambiguous, or the drag signal is tiny and unclear, choose the lower score.

Drag Annotation Legend. In image 2, the drag guidance uses this legend:

- red circle: handle/source point
- blue circle: target point
- white arrow: drag direction from handle to target

Output Format. Return exactly one valid JSON object and nothing else. Do not use markdown. Do not wrap the JSON in code fences. The JSON must be on a single line. Always output all required fields. Never output null, missing fields, placeholders, or non-integer scores. If uncertain, still choose one valid integer score for every score field.

Output Schema Example.
`{"camera_motion_score":3,"camera_motion_reason":"The framing and background stay almost identical between source and target.","instruction_alignment_score":2,"instruction_alignment_reason":"The drag arrows roughly follow the requested motion but only partially cover the edited part.","target_alignment_score":3,"target_alignment_reason":"The drag guidance clearly matches the visible local change in the target image.","confidence":"high","summary":"Reliable local drag annotation with negligible camera change."}`

Field Definitions.

- `camera_motion_score`: integer in $\{0, 1, 2, 3\}$
- `instruction_alignment_score`: integer in $\{0, 1, 2, 3\}$
- `target_alignment_score`: integer in $\{0, 1, 2, 3\}$
- `confidence`: one of `high`, `medium`, or `low`
- all reason and summary fields should be short and concrete

Scoring Rubric.

1. Camera motion score. The `camera_motion_score` compares image 1 and image 3 only.

- **3:** no noticeable camera motion; framing, viewpoint, and background layout are essentially unchanged
- **2:** very slight camera change or tiny crop/scale shift, but it is barely noticeable
- **1:** noticeable camera/view/framing shift that is not huge but should still be penalized
- **0:** large camera pan, zoom, tilt, rotation, viewpoint change, or global reframing

2. Instruction alignment score. The `instruction_alignment_score` compares the instruction and image 2.

- **3:** the drag guidance clearly and directly expresses the requested edit intent
- **2:** mostly aligned, but somewhat incomplete, partially ambiguous, or not fully precise
- **1:** weak alignment; some relevance exists, but the intent is hard to read, incomplete, noisy, or the drag magnitude is too small to clearly communicate the edit
- **0:** clearly mismatched with the instruction, or the drag guidance is so unclear that the intended edit cannot be identified

Important rule for `instruction_alignment_score`: tiny drags, nearly overlapping handle/target points, or arrows that barely indicate movement should receive a low score even if they are not obviously wrong.

3. Target alignment score. The `target_alignment_score` compares image 2 and image 3.

- **3:** the drag guidance clearly predicts the target edit; following the arrows would plausibly transform the source toward the target

- **2:** mostly predictive of the target, with minor mismatch or incompleteness
- **1:** weak relation; some target change is explained but important parts are missing or inaccurate
- **0:** the drag guidance does not explain the target edit

Additional Rules.

- Favor local geometric motion or deformation evidence over vague semantic guesses.
- If the target change is dominated by global camera change, `target_alignment_score` should usually be low.
- Do not hallucinate hidden edits that are not visible in the images.

Box G.3: Prompt used to generate natural-language editing instructions for DRAGBENCH-SR and DRAGBENCH-DR

You are an intent parser for drag-based image editing.
You are given two images of the same scene:

- **Source image**: the original image.
- **Annotated image**: the same image with drag annotations.

Your task is to infer the single most appropriate editing instruction implied by the drag annotations.

Image roles.

1. Use Source image to identify what objects, parts, people, animals, and structures truly exist in the scene.
2. Use Annotated image only to read the drag annotations and motion geometry.
3. Do not infer any editable subject that is not grounded in Source image.

Annotation meaning in Annotated image.

- red square: handle point
- blue circle: target point
- yellow arrow: intended drag direction from handle to target

Task objective. Infer what existing content should be edited, how it should be edited, and to what extent, based on the drag guidance. Return exactly one concise natural-language editing instruction.

Edits that fit drag-based editing include:

- moving an object or a coherent semantic part
- rotating, turning, tilting, or reorienting an object or part
- changing pose of a person, animal, or articulated object
- opening or closing an articulated structure
- extending, shortening, widening, narrowing, enlarging, shrinking, bending, stretching, compressing, or reshaping an existing part or structure

Edits that do not fit drag-based editing include:

- adding, removing, replacing, or duplicating objects
- changing identity
- changing background or unrelated scene content
- changing style, texture, material, lighting, weather, or rendering

Inference procedure.

1. Locate the visual content nearest each handle point in Source image.
2. Decide the correct semantic subject of editing: a whole object, a coherent part, an articulated part, or a local structure.
3. Prefer the most natural semantic level. If a handle lies on a small attached feature, first consider whether the drag is better explained by editing the larger connected structure rather than moving that small feature by itself.
4. Use the handle-to-target geometry to determine the edit type. Consider whether the drag indicates translation, rotation, reorientation, pose change, articulation, scaling, or local shape deformation.
5. Use the target location and drag distance to determine the intended final state or magnitude of the edit.
6. If multiple drag pairs are present, combine them into one coherent edit.
7. Output one instruction describing only the intended edit, while keeping all unrelated regions unchanged.

Guidelines for choosing the semantic subject.

- Prefer a realistic, coherent edit over isolated motion of a tiny local feature.
- If the handle lies on a rigidly attached facial feature such as the nose, nostril, cheek, eye corner, or ear, consider whether the intended edit is actually the head, face orientation, expression, or mouth/jaw articulation.

- If the handle lies on a tip, contour, boundary, or elongated protruding part such as hair, cloth edge, leaf, tail, horn, ear, branch, handle, or limb, consider whether the intended edit is local extent or shape change of that part.
- If the handle lies on a naturally articulated structure such as a jaw, mouth, beak, eyelid, lid, claw, arm, or leg, consider whether the intended edit is opening, closing, bending, raising, lowering, or pose change.
- Prefer the interpretation that best matches visible structure, drag direction, target position, and natural editing semantics.

Guidelines for determining edit type.

- Use translation when the drag suggests relocating the subject without changing its shape or orientation.
- Use rotation or reorientation when the drag suggests turning the subject around a coherent base or axis.
- Use pose change when the drag changes the configuration of an articulated body or part.
- Use opening or closing when connected parts are being separated or brought together around a hinge-like structure.
- Use scaling when the drag changes the overall size of the subject.
- Use local deformation when the drag changes the length, width, height, curvature, span, or shape of an existing part.

Guidelines for determining edit extent.

- Infer the final intended position, angle, pose, opening state, size, or shape from the target point rather than describing only the drag direction.
- Choose the resulting state that best matches the drag geometry and the visible structure.
- Be specific about the resulting orientation or structural change when it is visually implied.

Preservation constraints.

- When appropriate, include a brief constraint stating which closely related parts should remain unchanged.
- Preserve all unrelated objects, background regions, lighting, style, and scene content.
- Keep the constraint realistic and local to the edit. Mention unchanged parts only when they help specify the intended edit more precisely.
- Prefer preserving the surrounding structure that naturally anchors the edited subject, such as the rest of the body, the rest of the face, nearby connected parts, or the object's main base.
- Do not add unnecessary preservation details. Only mention the most relevant unchanged parts.
- The instruction may include a short clause such as while keeping the rest of the subject unchanged, when that makes the intent clearer.

Output requirements.

1. Return exactly one concise edit instruction.
2. Output only the instruction and nothing else.
3. Do not output explanations, reasoning steps, JSON, or alternatives.
4. Do not mention annotations, colors, points, arrows, coordinates, Image 1, or Image 2.
5. The instruction may include a brief preservation constraint about what should remain unchanged.
6. Keep unrelated content unchanged.

Examples of good outputs.

- Turn the dog's head toward the camera while keeping the body pose unchanged
- Open the lion's mouth while keeping the head position unchanged
- Raise the man's arm while keeping the rest of his pose unchanged
- Pull the hat brim down while keeping the hat crown in place
- Lengthen the table while keeping its position unchanged
- Widen the cup while keeping its height unchanged
- Extend the leaf into the gap while keeping the rest of the plant unchanged
- Move the fire hydrant to the left while keeping the background unchanged

Examples of bad outputs.

- Move the nose to the left
- Move one eye upward
- Add another leaf
- Make the image brighter
- Change the background

Box G.4: Prompt used for adapted PF evaluation in the editing setting.

Task Definition. You will be provided with a source image, an edited image, and an edit instruction. As an experienced evaluator, your task is to evaluate whether the edited image follows the edit instruction while preserving the unchanged visual content from the source image, according to the scoring criteria.
Scoring Criteria. When assessing the consistency of an edited image with an edit instruction, compare the source image and the edited image carefully. This evaluation can be based on several key aspects:

1. **Instruction alignment:** Determine whether the visible change in the edited image directly follows the requested edit instruction.
2. **Edit locality and preservation:** Evaluate whether areas, subjects, identities, background, style, and visual details that are not requested to change remain consistent with the source image.
3. **Accuracy:** Examine whether the edited target area, motion, deformation, position, appearance, or interaction is depicted correctly according to the instruction.
4. **Completeness:** Evaluate whether all critical parts of the instruction are reflected in the edited image without omitting important requested changes.

Scoring Range. Based on these criteria, a specific integer score from 0 to 4 can be assigned to determine the level of semantic consistency:

- **Very Poor (0):** No meaningful alignment. The edited image does not follow the instruction, or the edit severely damages unrelated content.
- **Poor (1):** Weak alignment. The edited image only vaguely relates to the instruction and has major errors, missing changes, or unnecessary alterations.
- **Fair (2):** Moderate alignment. The edited image partially follows the instruction but has noticeable inaccuracies, incomplete edits, or preservation issues.
- **Good (3):** Strong alignment. The edited image follows most of the instruction with only minor inaccuracies or minor unintended changes.
- **Excellent (4):** Near-perfect alignment. The edited image faithfully executes the instruction and preserves all unrelated visual content.

Input format. Every time you will receive an edit instruction and two images. The first image is the source image before editing, and the second image is the edited image after applying the instruction.
Please carefully review the source image, edited image, and edit instruction. Before giving a score, please provide a brief analysis of the above evaluation criteria, which should be very concise and accurate.
Output Format.
Analysis: [Your analysis]
Score: [Your Score]

Box G.5: Fixed prompt used in Generation-stage adaptive conditioning.

Repair this roughly warped edited image by filling all white blank holes, missing regions, and transparent-looking gaps with semantically appropriate content inferred from the surrounding image.
Do not simply fill the holes with background; reconstruct the correct object or scene content based on context, such as animal body parts, fur, grass, or other relevant structures.
Refine and correct any unnatural, distorted, warped, stretched, blurry, duplicated, or structurally inconsistent areas introduced by the coarse warp editing, especially inside and around the edited regions.
Preserve the original identity, appearance, composition, pose, background, colors, lighting, texture, perspective, and fine details as much as possible outside the damaged or edited areas.
Ensure the final image is realistic, seamless, and visually coherent, with natural transitions, clean boundaries, and no visible artifacts or inconsistencies.



Figure 11: Additional qualitative comparisons on diverse drag-based editing examples. *MoRe-Drag* faithfully follows the specified drag motion while preserving object structure, texture details, and background consistency.



Figure 12: Additional comparisons of drag-aware instruction inference. Each column shows a drag-annotated input, the instruction inferred by the vanilla MLLM, and the instruction inferred by the LoRA-adapted MLLM. LoRA adaptation improves grounding to the visual drag annotations and produces instructions that better reflect the intended object motion or deformation.

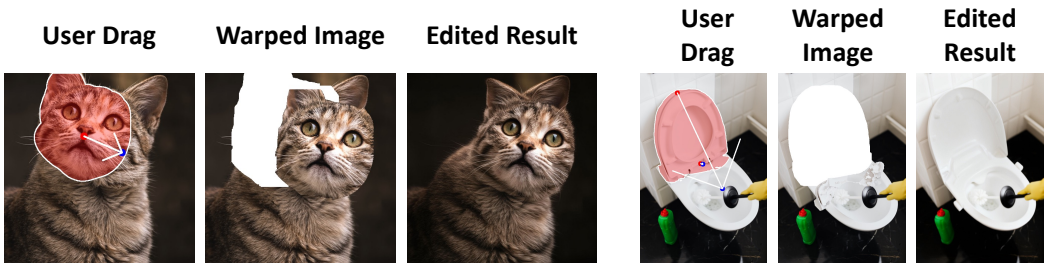

Figure 13: Failure cases. When the warped observation contains severe errors or ambiguous dis-occlusions, the injected motion evidence may not faithfully represent the intended edit. In these cases, *MoRe-Drag* may favor plausible restoration over the desired motion, such as incomplete head manipulation or unsuccessful lid articulation.



Figure 14: Failure cases of drag-aware instruction inference. The LoRA-adapted MLLM may still infer imperfect instructions when drag annotations are too close to each other, especially for fine-grained shape changes.

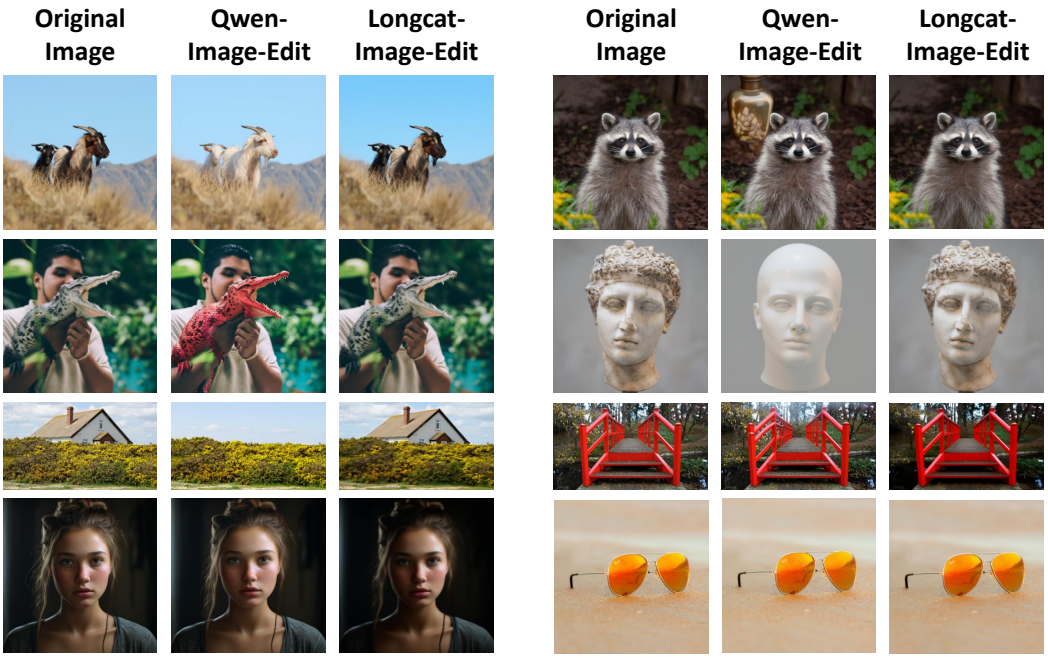

Figure 15: **Comparison of base editor reconstruction ability.** Under null conditions, LongCat-Image-Edit reconstructs the source images more faithfully than Qwen-Image-Edit, with better preservation of identity, structure, and appearance. This stronger reconstruction stability benefits motion-evidence injection and non-edited region preservation in *MoRe-Drag*.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction accurately reflect the paper's contributions and scope by summarizing the main contributions, technical components, and evaluation.

   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 have discussed the limitations in Sec. 5

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [N/A]

Justification: Our paper does not involve theoretical results.

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: Full implementation details are provided in Appendix C.

   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: The data are publicly available, and the code is included in the submitted supplementary materials.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: The main experimental details are provided Appendix C

   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: [Yes]

   Justification: We report results on the full benchmark sets. For experiments involving repeated runs, we report the mean and standard deviation. We additionally include efficiency statistics, including average editing time and peak GPU memory usage, in Table 16.

   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 provide the type of compute workers used for training and inference in Appendix C. We also report efficiency-related resource information, including average editing time and peak GPU memory usage, in Table 16.

   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: We have carefully reviewed the NeurIPS Code of Ethics and confirm that our research adheres to its guidelines.

   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: The broader impacts are discussed in Appendix A

    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: We do not release a new pretrained model or a newly scraped image dataset. Our data are derived annotations and processing metadata constructed from previously released datasets and pretrained models, which have already undergone their own release and safety procedures. Therefore, the release does not introduce additional high-risk data or model assets beyond those existing resources.

    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: We properly cite the original publications for all existing assets used in this paper, including the datasets and compared methods, in Sec. 4. The processed dataset used in our experiments is based on the publicly released ByteMorph-6M (demo) dataset, which is distributed under the CC0-1.0 Creative Commons Zero v1.0 Universal License. We use the dataset for academic research purposes and comply with the corresponding license and terms of use.

    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: We introduce a drag-instruction annotation dataset based on previously released data. Appendix B documents its construction pipeline, filtering criteria, quality scoring protocol, and statistics. The release will include the derived annotations and data-format documentation.

    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: [Yes]

    Justification: We conduct an anonymous online questionnaire with volunteer participants, who rank the edited results of three methods by naturalness, drag faithfulness, and preservation. Participation was voluntary and uncompensated.

    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: [No]

Justification: Our user study involved minimal-risk, anonymous participation in a ranking task conducted via an online questionnaire. No personal or sensitive information is collected, and our institution does not require IRB approval for such low-risk, non-identifiable human subject research.

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 use large vision-language models for instruction generation and editing-result scoring, and we describe this usage in Appendix F.

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.