CANNYDRAG: UNIFIED POINT, LINE, AND REGION DRAG EDITING VIA CANNY GUIDANCE

Anonymous authors
Paper under double-blind review



Figure 1: CannyDrag enables unified point-, line-, and region-level drag editing by applying task-specific transformations to the canny map. The transformed canny condition provides explicit structural guidance for (a) local deformation, (b) curve bending, and (c) similarity transformation, including translation, rotation, and scaling.

ABSTRACT

As generative models continue to advance, precise spatial controllability in image editing remains an important challenge. Drag-based image editing aims to control spatial changes through handle-target point pairs. However, existing drag-based editing suffers from ambiguity: the same sparse point-pair instruction can lead to different spatial editing results. This ambiguity arises from two related factors. At the instruction level, a point displacement specifies only the motion of the handle point, leaving the behavior of nearby content unspecified. At the supervision level, motion supervision is restricted to local neighborhoods around handle points and imposes limited constraints on surrounding regions. Motivated by this observation, we propose CannyDrag, which represents drag instructions with task-specific canny maps and incorporates them into inversion and progressive denoising for structural guidance beyond local point neighborhoods. The framework first supports point-level local deformation, line-level curve bending, and region-level similarity transformation, thereby refining ambiguous drag instructions into explicit editing objectives. We further construct DragBench-Line and DragBench-Region for evaluating curve bending and similarity transformation, respectively. Qualitative and quantitative experiments demonstrate the effectiveness of CannyDrag.

1 INTRODUCTION

Diffusion models have become a common foundation for image synthesis and editing (Rombach et al., 2022; Meng et al., 2022), enabling flexible control over image content. Image editing can be guided by text prompts (Brooks et al., 2023; Hertz et al., 2023), spatial layouts (Li et al., 2023; Zhang et al., 2025b; Huang et al., 2026), or image references (Zhang et al., 2023; Ye et al., 2023), each offering a different form of semantic or structural guidance. Nevertheless, these forms of guidance provide limited control over the precise spatial changes of specific visual elements.

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

Drag-based image editing specifies handle-target point pairs to move the corresponding visual content toward the target locations while preserving the plausibility of the remaining image. DragGAN first established this task for GAN-based models, where motion supervision constrains local feature movement while point tracking iteratively updates handle positions (Pan et al., 2023). Diffusion-based methods extend this task to multi-step denoising through inversion-based initialization and latent optimization (Shi et al., 2024; Zhang et al., 2025d). Subsequent studies further advance drag-based editing in efficiency and controllability (Ling et al., 2024; Cui et al., 2024; Liu et al., 2024; Zhao et al., 2024; Choi et al., 2025; Pu et al., 2026).

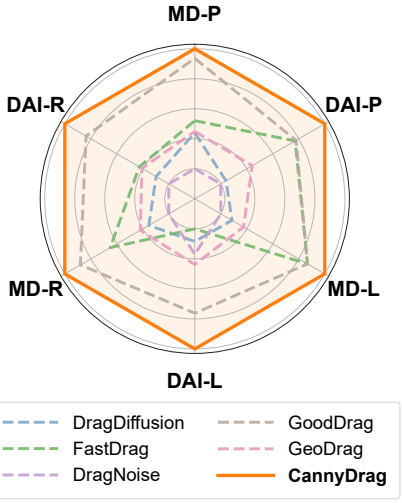


Figure 2: **Overall drag-editing performance across three task types.** P/L/R denote point-, line-, and region-level drag editing, respectively. DAI averages DAI_1, DAI_{10}, and DAI_{20}.

Despite these advances, existing drag-based methods still suffer from spatial editing ambiguity. At the instruction level, as illustrated in Fig. 3(a), a given handle-target displacement may correspond to different editing intentions, including local deformation, curve bending, or similarity transformation. Point-pair instructions specify the displacement of selected points while leaving the intended motion of surrounding pixels unspecified. At the supervision level, the motion-supervision formulation in Eq. (15) restricts feature matching to local neighborhoods around each handle point, while the behavior of surrounding regions is largely determined by the model's generative prior. Such **instruction ambiguity** and **spatial constraint incompleteness** may lead to a mismatch between the resulting edit and the intended manipulation.

To alleviate this limitation, we propose CannyDrag, a canny-guided drag editing framework that provides task-specific canny conditions for explicit structural guidance. **For instruction ambiguity**, we further extend drag editing to three task types: point-level local deformation, line-level curve bending, and region-level similarity transformation. Based on the sparse point-pair instructions, the original canny map is manipulated according to the selected task, providing explicit spatial constraints and an interpretable representation that users can inspect and refine interactively. **For spatial constraint incompleteness**, the canny condition is incorporated into the diffusion model

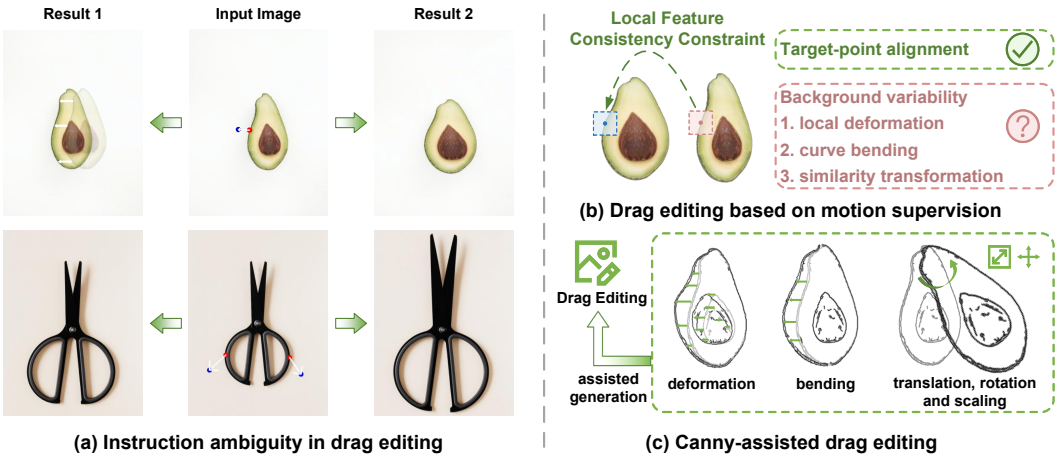


Figure 3: Ambiguity in point-based drag editing and our canny-assisted solution. (a) Sparse point pairs may correspond to different spatial editing intentions. (b) Local motion supervision mainly constrains neighborhoods around handle points, leaving surrounding regions weakly specified. (c) CannyDrag transforms the canny map according to the intended point, line, or region edit and uses it as structural guidance.

during inversion and multi-step denoising, providing broader structural guidance beyond local point neighborhoods. The original canny map helps preserve the source structure during inversion, while the manipulated canny map is progressively updated during denoising to guide the intended edit. Existing point-level benchmarks cannot fully assess continuous line deformation or coherent region transformation. For systematic evaluation, we construct DragBench-Line and DragBench-Region by extending DragBench with annotations for line-level and region-level editing. These extensions enable unified evaluation across point-, line-, and region-level drag editing.

Our contributions are summarized as follows:

- We propose CannyDrag, the first drag-editing framework that progressively manipulates canny maps as task-specific structural conditions. The original canny map supports structure-preserving inversion, while its progressive transformation guides the denoising trajectory toward the intended spatial edit.
- CannyDrag provides a unified and practical framework for point-level local deformation, line-level curve bending, and region-level similarity transformation. All three tasks share the same diffusion backbone.
- We formalize the three drag-editing tasks and construct DragBench-Line and DragBench-Region for evaluating curve bending and similarity transformation, respectively.

2 Related Work

2.1 Generative Image Editing

Image editing has evolved with the development of image generation models (Karras et al., 2019; Rombach et al., 2022; An et al., 2026; Shao & Li, 2026). Early editing methods (Karras et al., 2019; 2020; Patashnik et al., 2021; Pan et al., 2023) are primarily based on GAN architectures (Goodfellow et al., 2014; Mirza & Osindero, 2014; Reed et al., 2016). In recent years, image editing methods based on diffusion models (Ho et al., 2020; Song et al., 2021) and flow-based models (Lipman et al., 2023; Liu et al., 2023) have shown better performance in fidelity and controllability. Training-free methods (Meng et al., 2022; Hertz et al., 2023) achieve image editing without additional training by using inversion and attention replacement on frozen diffusion models. Test-time optimization (TTO) methods (Mokady et al., 2023; Kim et al., 2022; Ruiz et al., 2023) achieve text-guided or subject-driven image editing by fine-tuning model parameters or embeddings. To alleviate the computational burden during inference, several methods (Brooks et al., 2023; Ye et al., 2023; Zhang et al., 2025c) learn editing or conditioning capabilities in advance, avoiding per-instance optimization. Beyond task-specific editing models, recent multi-task or unified frameworks (Sheynin et al., 2024; Xiao et al., 2025; Black Forest Labs et al., 2025) further integrate diverse editing and generation tasks into a single model. However, these methods primarily focus on semantic-level editing and provide limited explicit spatial controllability. Layout-driven methods (Li et al., 2023; Zhang et al., 2025b; Huang et al., 2026) provide control over the spatial positions and arrangements of multiple objects, while drag-based methods enable pixel-level spatial control.

2.2 Drag-based Image Editing

Drag-based image editing aims to move the visual content around user-specified source points toward corresponding target points, while preserving the visual plausibility and semantic consistency of the original image. DragGAN (Pan et al., 2023) proposes motion supervision and point tracking, which establish an effective paradigm for this task and inspire subsequent studies. Building on this paradigm, DragDiffusion (Shi et al., 2024) extends drag-based editing from GANs to diffusion models by optimizing inverted noisy latents at the initial timestep. Compared with GANs, diffusion models generate images through multi-step denoising, while DragDiffusion does not further supervise the positions of points in subsequent timesteps. Based on this observation, GoodDrag (Zhang et al., 2025d) performs supervised optimization over multiple timesteps, improving editing fidelity and stability. Other studies further improve this paradigm from different aspects, including tracking reliability (Ling et al., 2024; Cui et al., 2024), latent optimization choices (Liu et al., 2024), text guidance (Jiang et al., 2025; Choi et al., 2025), and 3D geometry guidance (Pu et al., 2026). Despite these advances, most motion supervision losses are restricted to local neighborhoods around target

points and lack holistic guidance. FastDrag (Zhao et al., 2024) constructs a latent warpage field within the mask to relocate noisy latents in one step, thereby improving editing efficiency. RegionDrag (Lu et al., 2024) and Inpaint4Drag (Lu & Han, 2025) enable region-level dragging through latent-level pasting or pixel-space warping, but lack the ability to explicitly control complex scaling, rotation, and non-rigid deformation. In this paper, we distinguish drag instructions into point, line, and region forms, allowing the editing process to better align with user intentions.

3 CANNYDRAG

3.1 AMBIGUITY IN POINT-BASED DRAGGING

Although drag-based editing has been widely adopted for its concise and interactive editing paradigm, point-pair instructions still suffer from ambiguous editing intentions. As illustrated in Fig. 3(a), under the same editing instruction, existing drag-based editing methods may produce different results. The movement from the source point to the target point may drive the deformation of surrounding pixels or move the object as a whole. This ambiguity burdens users with repeated stochastic trials to obtain the desired result, while providing no explicit channel to further communicate their editing intentions to the model.

The motion-supervision objective in Eq. (15) helps explain this limitation. In the equation, the objective matches features in the source local region $\Omega(\mathbf{h}_i^k, r_1)$ with those in the shifted region $\Omega(\mathbf{h}_i^k + \mathbf{d}_i^k, r_1)$. This constraint can ensure feature matching within the point neighborhood, while the remaining regions largely rely on the model's own generative prior for completion. The model has to infer one possible editing intention associated with the point movement. Therefore, the ambiguity arises from two aspects: the inherent uncertainty of point-pair drag instructions and insufficient constraints over surrounding regions.

To address these two issues, we refine task categories and introduce an extra conditional constraint. For task categorization, we categorize the editing task into finer-grained types, including point-level local deformation, line-level curve bending, and region-level similarity transformation. This categorization provides a clearer task distinction and alleviates the inherent uncertainty of point-pair drag instructions. For the conditional constraint, we introduce the canny map to provide explicit structural information. As shown in Fig. 3(c), different transformations are applied to the canny map according to the categorized task type, and the transformed canny map guides the generation process. Despite being a weak structural condition, the canny map introduces image-level constraints that help clarify editing intentions.

3.2 PIPELINE OF CANNYDRAG

Overview. As shown in Fig. 4, CannyDrag organizes the inference process through canny-guided structural conditioning. Prior to inference, the denoising UNet and ControlNet (Zhang et al., 2023) are fine-tuned on the input image and canny map, enabling the model to better preserve the source subject under canny-based control. During inversion, the source image is inverted under the guidance of the original canny map to obtain an inverted latent that can faithfully reconstruct the input image after denoising. After inversion, a sub-task is selected according to the editing intention, which determines how the original canny map is manipulated. These sub-tasks include point-level local warping, line-level curve bending, and region-level similarity transformation, as detailed in Sec. 3.3. The manipulated canny maps at different denoising timesteps are then fed into ControlNet, guiding the denoising process to follow the target structure. In this way, the sparse drag instruction is converted into a task-specific structural condition, ensuring that the denoising process follows the overall editing intention.

Canny-Guided Inversion. By leveraging noise (Li, 2024; Zhang et al., 2025a), diffusion models establish a progressive trajectory for image synthesis and editing. Standard DDIM inversion obtains a latent that can reconstruct the source image along the text-conditioned trajectory, while introducing the canny condition during denoising would break this reversible trajectory. To maintain reversibility, we introduce the canny condition into the inversion stage as well. Specifically, we first extract the original canny condition from the input image:

$$\mathbf{c}_{\mathrm{e}}^0 = \mathcal{E}(I_0), \tag{1}$$

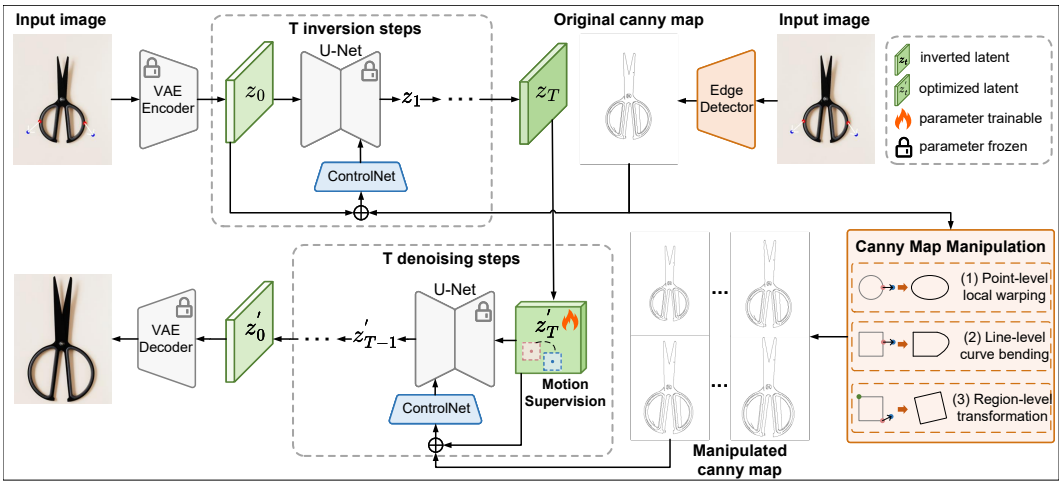


Figure 4: **Inference Pipeline of CannyDrag.** The canny-based control is introduced into both inversion and denoising to ensure faithful image reconstruction. The canny map used in the denoising stage is modified according to different drag intentions. The modified canny maps then provide structural guidance for the subsequent denoising process.

where $\mathcal{E}$ denotes the edge detector and $\mathbf{c}_{\mathrm{e}}^0$ is the original canny condition. Starting from the VAE latent $\mathbf{z}_0$, the inversion process then maps the image latent toward $\mathbf{z}_T$ under both the text condition $\mathbf{c}_{\mathrm{t}}$ and the canny condition $\mathbf{c}_{\mathrm{e}}^0$. At the selected inversion steps, the noise prediction is computed by $\epsilon_{\theta,\phi}\left(\mathbf{z}_t, t, \mathbf{c}_{\mathrm{t}}, \mathbf{c}_{\mathrm{e}}^0\right)$, where θ and ϕ denote the parameters of the denoising U-Net and ControlNet, respectively. This prediction $\epsilon_{\theta,\phi}$ is then applied to the DDIM inversion update in Eq. (14). In practice, we apply canny guidance to high-noise inversion steps, where coarse spatial structures are primarily established.

Progressive Canny-Guided Denoising. To introduce explicit structure-level constraints, we manipulate the original canny map $(\mathbf{c}_{\mathrm{e}}^0)$ according to the drag instruction and feed the transformed canny map $\mathbf{c}_{\mathrm{e}}^t$ into ControlNet during denoising. As shown in Fig. 4, the corresponding manipulation is performed progressively along timestep t, which provides a sequence of intermediate structural conditions and avoids abrupt geometry changes caused by one-step warping. This manipulation is instantiated differently for point, line, and region tasks, as detailed in Sec. 3.3.

Specifically, let $\tau \in \{\mathrm{point}, \mathrm{line}, \mathrm{region}\}$ denote the task type and T_{c} the total number of progressive canny manipulation steps. The manipulation progress is defined as $\rho(t) = (T - t + 1)/T_{\mathrm{c}}$. The dragged-area mask is denoted by A_τ, with A_{line} and A_{region} specifying the line and region areas, respectively; this term is omitted for point-level dragging. The manipulated canny condition is defined as

$$\mathbf{c}_{\mathrm{e}}^t = \mathcal{G}_\tau\left(\mathbf{c}_{\mathrm{e}}^0; \rho(t), \mathcal{P}, M, A_\tau\right), \tag{2}$$

where $\mathcal{G}_\tau$ denotes the task-specific geometric manipulation function on the canny map. With this transformed canny condition $\mathbf{c}_{\mathrm{e}}^t$, the noise prediction is computed as $\epsilon_{\theta,\phi}(\mathbf{z}_t, t, \mathbf{c}_{\mathrm{t}}, \mathbf{c}_{\mathrm{e}}^t)$. After canny-guided denoising, we apply motion supervision in Eq. (15) to optimize the latent and point tracking in Eq. (16) to update handle positions under the current canny condition.

3.3 Task-Adaptive Canny Mapping

The mapping function $\mathcal{G}_\tau$ is instantiated by drag type $\tau \in \{\mathrm{point}, \mathrm{line}, \mathrm{region}\}$ to produce task-specific canny transformations. Given the handle-target point pairs $\mathcal{P} = \{(\mathbf{h}_i, \mathbf{g}_i)\}_{i=1}^N$, each pair provides a displacement vector $\mathbf{d}_i = \mathbf{g}_i - \mathbf{h}_i$. For a binary canny edge map $\mathbf{c}_{\mathrm{e}}$, its edge-pixel set is written as $\mathcal{S}(\mathbf{c}_{\mathrm{e}}) = \{\mathbf{p} \mid \mathbf{c}_{\mathrm{e}}(\mathbf{p}) = 1\}$. The transformed edge-pixel set at timestep t is denoted by $\mathcal{S}_\tau^t$. Since canny maps are binary, we avoid directly interpolating their pixel values. Instead, each task-specific mapping explicitly moves edge pixels and reconnects adjacent source pixels to form a valid binary canny condition. The following paragraphs describe how $\mathcal{S}_\tau^t$ is constructed for point, line, and region dragging.

Point-level Mapping with Local Warping. For point-level dragging, the editable mask M defines the region affected by the drag instruction. Following the latent warpage field design (Zhao et al., 2024), sparse point displacements are propagated to each canny pixel coordinate $\mathbf{p}$ satisfying $M(\mathbf{p}) = 1$:

$$\mathbf{v}_{\text{point}}(\mathbf{p}) = \sum_{i=1}^{N} w_i(\mathbf{p})\lambda_i(\mathbf{p})\mathbf{d}_i. \tag{3}$$

The weight $w_i(\mathbf{p})$ gives the relative contribution of the i-th handle among all N handles, and $\lambda_i(\mathbf{p})$ serves as a distance-based attenuation factor. FastDrag (Zhao et al., 2024) computes this attenuation with a circumscribed circle, whereas CannyDrag follows the actual mask boundary to better fit irregular editable regions. The detailed computation of $w_i(\mathbf{p})$ and $\lambda_i(\mathbf{p})$ is provided in Appendix B.1. The progressive point-level edge set at timestep t is then written as

$$\mathcal{S}^t_{\text{point}} = \left\{ \mathbf{p} + \rho(t)\mathbf{v}_{\text{point}}(\mathbf{p}) \mid \mathbf{p} \in \mathcal{S}(\mathbf{c}^0_{\text{e}}),\ M(\mathbf{p}) = 1 \right\}. \tag{4}$$

Adjacent source pixels are reconnected to preserve canny edge continuity.

Line-level Mapping with Curve Bending. For line-level dragging, the dragged-area mask A_{line} indicates the pixels covered by the annotated line area. The line is first reduced to an ordered centerline $\boldsymbol{\gamma}(s) : [0, 1] \to \mathbb{R}^2$, where s is the normalized arc-length coordinate. Each handle point $\mathbf{h}_i$ is projected onto the centerline by

$$s_i = \arg\min_{s\in[0,1]} \|\boldsymbol{\gamma}(s) - \mathbf{h}_i\|_2. \tag{5}$$

The displacement along the centerline is obtained by spline interpolation over the projected control points:

$$\mathbf{u}_t(s) = \text{Spline}\left(\{(s_i, \rho(t)\mathbf{d}_i)\}_{i=1}^{N} \cup \{(0, \mathbf{0}), (1, \mathbf{0})\} \right). \tag{6}$$

Each tuple contains a centerline coordinate and its displacement vector; the additional tuples $(0, \mathbf{0})$ and $(1, \mathbf{0})$ impose zero displacement at the two endpoints of the annotated line. The target centerline at timestep t is defined as $\boldsymbol{\gamma}_t(s) = \boldsymbol{\gamma}(s) + \mathbf{u}_t(s)$. For an edge pixel $\mathbf{p} \in \mathcal{S}(\mathbf{c}^0_{\text{e}}) \cap A_{\text{line}}$, its nearest centerline coordinate is

$$s(\mathbf{p}) = \arg\min_{s\in[0,1]} \|\boldsymbol{\gamma}(s) - \mathbf{p}\|_2. \tag{7}$$

The relative offset between $\mathbf{p}$ and its nearest centerline point is

$$\Delta_{\text{line}}(\mathbf{p}) = \mathbf{p} - \boldsymbol{\gamma}(s(\mathbf{p})). \tag{8}$$

The line-level transformed edge set is therefore

$$\mathcal{S}^t_{\text{line}} = \left\{ \boldsymbol{\gamma}_t(s(\mathbf{p})) + \Delta_{\text{line}}(\mathbf{p}) \mid \mathbf{p} \in \mathcal{S}(\mathbf{c}^0_{\text{e}}) \cap A_{\text{line}} \right\}. \tag{9}$$

This mapping bends the line according to the control points while preserving the local offset of edge pixels around the centerline. Similar to point-level mapping, adjacent source pixels are reconnected to preserve edge continuity.

Region-level Mapping with Similarity Transformation. For region-level dragging, A_{region} denotes the user-specified dragged area, and the canny edges inside this area are transformed as a coherent structure. At timestep t, the target position of each control point is interpolated as $\mathbf{h}_i + \rho(t)\mathbf{d}_i$. CannyDrag estimates a partial affine transformation from the original control points to these interpolated positions:

$$(\alpha^*_t, \mathbf{R}^*_t, \mathbf{b}^*_t) = \arg\min_{\alpha,\mathbf{R},\mathbf{b}} \sum_{i=1}^{N} \|\alpha\mathbf{R}\mathbf{h}_i + \mathbf{b} - (\mathbf{h}_i + \rho(t)\mathbf{d}_i)\|_2^2. \tag{10}$$

This partial affine form corresponds to a global similarity transform, where α represents uniform scaling, $\mathbf{R}$ represents rotation, and $\mathbf{b}$ represents translation. When only one point pair is available, the transformation degenerates to a translation given by the corresponding point displacement. With two non-degenerate point pairs, the partial affine transform is directly determined; with more point pairs, it is estimated by robust fitting. The region-level transformed edge set is given by

$$\mathcal{S}^t_{\text{region}} = \left\{ \alpha^*_t \mathbf{R}^*_t \mathbf{p} + \mathbf{b}^*_t \mid \mathbf{p} \in \mathcal{S}(\mathbf{c}^0_{\text{e}}) \cap A_{\text{region}} \right\}. \tag{11}$$

Compared with independent point warping, this transformation keeps the selected region as a coherent unit, making it better suited for object-level movement, rotation, and resizing.



Figure 5: **Qualitative results on point-level drag editing.** The visual comparison demonstrates that CannyDrag achieves more accurate spatial editing and produces more plausible edits consistent with the intended manipulation.

4 Experiments

4.1 Experimental Details

All experiments are built on Stable Diffusion v1.5 (Rombach et al., 2022) with a ControlNet branch (Zhang et al., 2023) for canny-conditioned denoising. We adopt DragBench (Shi et al., 2024) for point-level evaluation and construct DragBench-Line and DragBench-Region to evaluate the two additional drag editing tasks, curve bending and similarity transformation, respectively. We compare CannyDrag with representative drag-based image editing methods, including DragDiffusion (Shi et al., 2024), FastDrag (Zhao et al., 2024), DragNoise (Liu et al., 2024), GoodDrag (Zhang et al., 2025d), and GeoDrag (Pu et al., 2026). Further implementation, benchmark construction, and evaluation details are provided in Appendix C.

4.2 Qualitative Results

Figure 5 compares the point-level editing results of CannyDrag with representative baselines. As shown in the figure, CannyDrag provides more accurate spatial control and better translates sparse point pairs into coherent image-level edits. As shown in the fourth and fifth rows, CannyDrag accurately straightens the sail and scales down the light bulb, respectively, demonstrating more precise editing control than competing methods. Moreover, CannyDrag achieves the intended geometric transformations while preserving overall visual plausibility, as exemplified in the third row, where it produces a natural facial rotation that remains coherent with the rest of the image. These advantages arise from the manipulated canny condition, which converts point movements into explicit structural changes and guides denoising beyond local control neighborhoods.

Figure 6 presents the qualitative results of CannyDrag for curve bending and similarity transformation. The left-side results show that CannyDrag smoothly bends continuous structures, such as

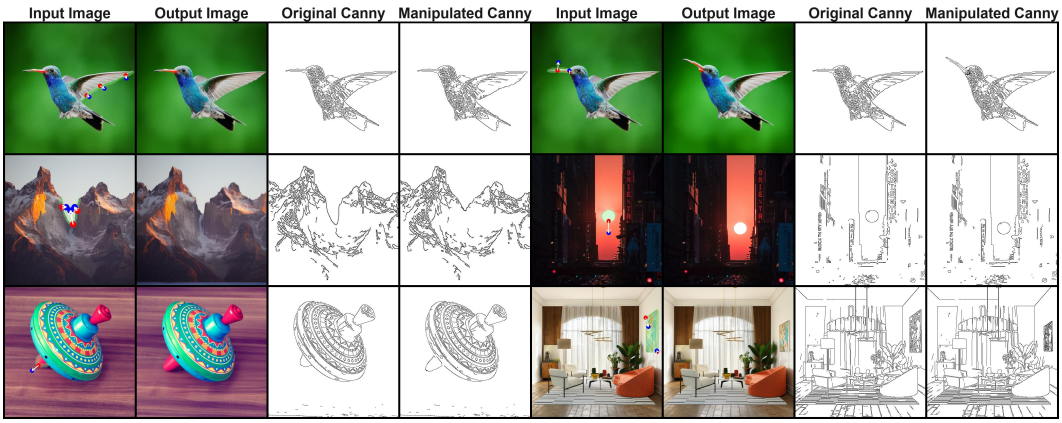


Figure 6: **Qualitative results on curve bending (left) and similarity transformation (right).** CannyDrag effectively bends continuous line structures and performs region-level translation, rotation, and scaling while faithfully preserving the surrounding content.

Table 1: Quantitative comparison on DragBench. Lower **MD/DAI** is better and higher **IF** is better. DAI values are multiplied by 100.

Method	MD↓	DAI_1↓	DAI_{10}↓	DAI_{20}↓	IF↑
DragDiffusion (Shi et al., 2024)	31.55	18.24	17.02	16.09	88.36
DragNoise (Liu et al., 2024)	36.51	18.69	17.54	16.41	**90.07**
FastDrag (Zhao et al., 2024)	29.87	12.00	11.52	10.81	85.23
GoodDrag (Zhang et al., 2025d)	21.27	12.55	11.16	10.40	86.26
GeoDrag (Pu et al., 2026)	31.35	16.11	15.09	13.70	85.95
CannyDrag (ours)	**19.95**	**9.47**	**8.80**	**8.77**	86.32

the hummingbird wing, mountain contour, and spinning-top boundary, while maintaining structural continuity and preserving unrelated content. Results on the right demonstrate coherent region-level translation, rotation, and scaling. Through Canny-guided intermediate feature interactions, sparse point pairs convey editing instructions more explicitly, enabling the model to better interpret the editing intent and produce coherent image-level transformations. Unlike local point displacement, curve bending and similarity transformation require coordinated movement over continuous structures. The manipulated canny map explicitly represents this coordination, preventing isolated control points from reaching their targets while the intervening structure remains inconsistent.

4.3 Quantitative Results

Figure 2 summarizes spatial accuracy and local preservation across the three drag-editing tasks, showing the overall advantage of CannyDrag in MD and DAI. On DragBench, CannyDrag achieves the best MD of 19.95 and the lowest DAI values of 9.47/8.80/8.77, with an IF of 86.32, indicating accurate target positioning and well-controlled local changes. For curve bending, it reduces MD by 6.3% over the strongest baseline and consistently obtains the lowest DAI across all neighborhood radii (Table 2), while the slightly lower IF reflects the structural changes required by this task. On DragBench-Region, CannyDrag also achieves the best MD and DAI (Table 2). Since IF measures similarity to the source image, an intended transformation over a larger area can reduce IF even when the editing result is geometrically correct. MD and DAI therefore provide complementary evidence of editing accuracy. The consistent DAI improvements at all neighborhood radii further indicate that the gain extends beyond target-point alignment to the surrounding structure. This trend supports the role of task-adaptive canny maps, whose broader structural guidance becomes increasingly important as the edit expands from a continuous line to a coherent region.

Table 2: Quantitative comparison on DragBench-Line and DragBench-Region. Lower **MD/DAI** is better and higher **IF** is better. DAI values are multiplied by 1,000.

Method	DragBench-Line					DragBench-Region				
	MD↓	DAI_1↓	DAI_{10}↓	DAI_{20}↓	IF↑	MD↓	DAI_1↓	DAI_{10}↓	DAI_{20}↓	IF↑
DragDiffusion	10.97	27.26	21.48	16.11	94.75	39.58	14.63	15.97	16.87	**88.04**
FastDrag	8.61	27.90	22.31	16.63	93.34	28.40	13.70	14.62	14.84	86.38
DragNoise	11.32	24.62	21.29	16.90	94.51	45.35	15.03	17.46	18.24	87.87
GoodDrag	8.61	23.40	17.18	12.96	**94.83**	19.78	8.58	10.16	10.77	85.71
GeoDrag	10.61	25.43	20.36	15.44	93.33	37.09	13.31	14.93	15.62	86.68
CannyDrag (ours)	**8.07**	**21.20**	**14.93**	**11.77**	94.46	**15.28**	**6.59**	**8.41**	**9.08**	84.71

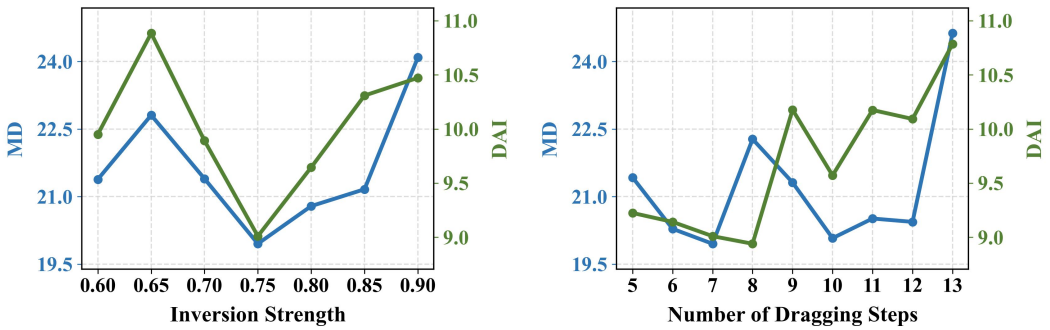


Figure 7: Ablation results for **inversion strength** and the **number of dragging steps**.

4.4 Ablation Study

We study the sensitivity of CannyDrag to the inversion strength and the number of progressive guidance steps using MD and DAI. As shown in Fig. 7, the two factors exhibit non-monotonic effects: an inversion strength of 0.75 achieves the best balance between spatial movement and source-image preservation, while increasing the dragging steps from 5 to 7 improves spatial accuracy. Although 8 steps slightly reduces DAI, its MD increases, and further steps lead to unstable performance. The opposite trends of MD and DAI after seven steps suggest that prolonged guidance continues to preserve local correspondence but begins to over-constrain target displacement. This explains why neither metric alone is sufficient for selecting the guidance duration. Weaker inversion limits spatial movement, whereas stronger inversion reduces consistency with the source image. A moderate guidance duration introduces structure progressively without over-constraining later denoising, indicating that the initial noise level and guidance duration jointly balance editability and preservation. Accordingly, we set the inversion strength to 0.75 and $T_c = 7$ for point-level drag editing.

5 Conclusion

In this work, we propose CannyDrag for canny-guided drag-based image editing. The framework transforms sparse point-pair instructions into task-specific canny maps for point-level local deformation, line-level curve bending, and region-level similarity transformation. These maps are incorporated into inversion and progressive denoising to provide structural guidance beyond local point neighborhoods. Experiments across the three task types demonstrate improved spatial accuracy while maintaining competitive image fidelity. Future work will explore more complex non-rigid transformations and interactions among multiple objects.

AI USE STATEMENT

Generative AI tools were used to polish the manuscript and assist with code development. All AI-assisted text and code were reviewed and verified by the authors. The authors take full responsibility for the final content of this work.

REFERENCES

Hongjun An, Wenhan Hu, Sida Huang, Siqi Huang, Ruanjun Li, Yuanzhi Liang, Jiawei Shao, Yiliang Song, Zihan Wang, Cheng Yuan, Chi Zhang, Hongyuan Zhang, Wenhao Zhuang, and Xuelong Li. AI Flow: Perspectives, scenarios, and approaches. *Vicinagearth*, 3, 2026. doi: 10.1007/s44336-025-00031-y.

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 Mueller, 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.

Tim Brooks, Aleksander Holynski, and Alexei A. Efros. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18392–18402, 2023.

Gayoon Choi, Taejin Jeong, Sujung Hong, and Seong Jae Hwang. DragText: Rethinking text embedding in point-based image editing. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 441–450, 2025.

Yutao Cui, Xiaotong Zhao, Guozhen Zhang, Shengming Cao, Kai Ma, and Limin Wang. StableDrag: Stable dragging for point-based image editing. In *European Conference on Computer Vision*, pp. 340–356, 2024.

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

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

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Sida Huang, Siqi Huang, Ping Luo, and Hongyuan Zhang. Laytrol: Preserving pretrained knowledge in layout control for multimodal diffusion transformers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 5113–5121, 2026. doi: 10.1609/aaai.v40i7.37425.

Ziqi Jiang, Zhen Wang, and Long Chen. CLIPDrag: Combining text-based and drag-based instructions for image editing. In *International Conference on Learning Representations*, 2025.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4401–4410, 2019.

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8110–8119, 2020.

Gwanghyun Kim, Taesung Kwon, and Jong Chul Ye. Diffusionclip: Text-guided diffusion models for robust image manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2426–2435, 2022. doi: 10.1109/CVPR52688.2022.00246.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun (eds.), *International Conference on Learning Representations*, 2014. URL https://arxiv.org/abs/1312.6114.

Xuelong Li. Positive-incentive noise. *IEEE Transactions on Neural Networks and Learning Systems*, 35(6):8708–8714, 2024. doi: 10.1109/TNNLS.2022.3224577.

Yuheng Li, Haotian Liu, Qingyang Wu, Fangzhou Mu, Jianwei Yang, Jianfeng Gao, Chunyuan Li, and Yong Jae Lee. GLIGEN: Open-set grounded text-to-image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22511–22521, 2023.

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

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

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

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

Jingyi Lu and Kai Han. Inpaint4Drag: Repurposing inpainting models for drag-based image editing via bidirectional warping. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 18304–18313, 2025.

Jingyi Lu, Xinghui Li, and Kai Han. RegionDrag: Fast region-based image editing with diffusion models. In *European Conference on Computer Vision*, pp. 231–246, 2024.

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

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

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

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*, pp. 1–11. Association for Computing Machinery, 2023. doi: 10.1145/3588432.3591500.

Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. Styleclip: Text-driven manipulation of stylegan imagery. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2085–2094, 2021.

Xinyu Pu, Hongsong Wang, Jie Gui, and Pan Zhou. Dragging with geometry: From pixels to geometry-guided image editing. In *The Fourteenth International Conference on Learning Representations*, 2026.

Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *International conference on machine learning*, pp. 1060–1069. PMLR, 2016.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bjorn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022.

Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22500–22510, 2023.

Jiawei Shao and Xuelong Li. AI Flow at the network edge. *IEEE Network*, 40(1):330–336, 2026.

Shelly Sheynin, Adam Polyak, Uriel Singer, Yuval Kirstain, Amit Zohar, Oron Ashual, Devi Parikh, and Yaniv Taigman. Emu edit: Precise image editing via recognition and generation tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8871–8879, 2024.

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8839–8849, 2024.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=St1giarCHLP.

Luming Tang, Menglin Jia, Qianqian Wang, Cheng Perng Phoo, and Bharath Hariharan. Emergent correspondence from image diffusion. In *Advances in Neural Information Processing Systems*, volume 36, pp. 1363–1389, 2023.

Shitao Xiao, Yueze Wang, Junjie Zhou, Huaying Yuan, Xingrun Xing, Ruiran Yan, Chaofan Li, Shuting Wang, Tiejun Huang, and Zheng Liu. Omnigen: Unified image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13294–13304, 2025.

Hu Ye, Jun Zhang, Sibo Liu, Xiao Han, and Wei Yang. IP-Adapter: Text compatible image prompt adapter for text-to-image diffusion models. *arXiv preprint arXiv:2308.06721*, 2023.

Hongyuan Zhang, Sida Huang, Yubin Guo, and Xuelong Li. Variational positive-incentive noise: How noise benefits models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47 (9):8313–8320, 2025a. doi: 10.1109/TPAMI.2025.3575295.

Hui Zhang, Dexiang Hong, Yitong Wang, Jie Shao, Xinglong Wu, Zuxuan Wu, and Yu-Gang Jiang. CreatiLayout: Siamese multimodal diffusion transformer for creative layout-to-image generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 18487–18497, 2025b.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3813–3824, 2023.

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 transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025c. arXiv:2504.20690.

Zewei Zhang, Huan Liu, Jun Chen, and Xiangyu Xu. GoodDrag: Towards good practices for drag editing with diffusion models. In *International Conference on Learning Representations*, 2025d.

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*, volume 37, pp. 74439–74460, 2024. doi: 10.52202/079017-2369.

Appendix

A PRELIMINARIES

A.1 DIFFUSION-BASED IMAGE EDITING

Given an input image I, a pretrained text-to-image diffusion model maps it into a latent representation $\mathbf{z}_0$ through the VAE encoder (Kingma & Welling, 2014). In the forward diffusion process, Gaussian noise is gradually added to $\mathbf{z}_0$ according to a predefined noise schedule (Ho et al., 2020):

$$\mathbf{z}_t = \sqrt{\alpha_t}\mathbf{z}_0 + \sqrt{1-\alpha_t}\epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I}), \tag{12}$$

The accumulated noise schedule is denoted by α_t. At each timestep t, the denoising network predicts the noise $\epsilon_\theta(\mathbf{z}_t, t, \mathbf{c}_\mathrm{t})$, with $\mathbf{c}_\mathrm{t}$ denoting the text condition. The predicted noise is used to estimate the clean latent:

$$\hat{\mathbf{z}}_0 = \frac{\mathbf{z}_t - \sqrt{1-\alpha_t}\epsilon_\theta(\mathbf{z}_t, t, \mathbf{c}_\mathrm{t})}{\sqrt{\alpha_t}}. \tag{13}$$

The scheduler then updates $\mathbf{z}_t$ toward a less noisy latent $\mathbf{z}_{t-1}$. To edit a real image while preserving its original content, the clean latent $\mathbf{z}_0$ is first mapped to a noisy latent $\mathbf{z}_T$ by deterministic DDIM inversion (Song et al., 2021). This inversion can be viewed as reversing the DDIM sampling trajectory. Following the DDIM update rule, the inversion step from $\mathbf{z}_t$ to $\mathbf{z}_{t+1}$ can be written as

$$\mathbf{z}_{t+1} = \sqrt{\alpha_{t+1}}\hat{\mathbf{z}}_0 + \sqrt{1-\alpha_{t+1}}\epsilon_\theta(\mathbf{z}_t, t, \mathbf{c}_\mathrm{t}). \tag{14}$$

The inverted latent $\mathbf{z}_T$ provides the initialization for the following denoising process, allowing real-image editing to start from a diffusion trajectory aligned with the input image.

A.2 DRAG-BASED EDITING FORMULATION

Drag-based editing (Pan et al., 2023) moves specified content from handle points to target points while preserving image plausibility and consistency. A drag instruction is represented by a set of handle-target point pairs $\mathcal{P} = \{(\mathbf{h}_i, \mathbf{g}_i)\}_{i=1}^N$, where $\mathbf{h}_i = (x_i^h, y_i^h)$ is the handle point selected on the source image and $\mathbf{g}_i = (x_i^g, y_i^g)$ is the corresponding target point. Most drag-based methods (Pan et al., 2023; Zhang et al., 2025d) decompose this task into motion supervision and point tracking, which are repeated for optimization iterations $k = 1, \ldots, K$.

A.3 MOTION SUPERVISION AND POINT TRACKING

Motion Supervision. Motion supervision optimizes the latent representation by enforcing feature movement at each iteration. DragDiffusion (Shi et al., 2024) performs this optimization at a selected timestep, whereas GoodDrag (Zhang et al., 2025d) extends it to multiple denoising timesteps by alternating motion supervision and denoising. Let $\mathbf{h}_i^k$ be the tracked handle position of the i-th point pair at iteration k, with $\mathbf{h}_i^0 = \mathbf{h}_i$. The intermediate feature extractor of the denoising U-Net is denoted by $F(\cdot)$, and $F_\mathbf{x}(\mathbf{z}_t^k)$ represents the feature at point $\mathbf{x}$ from latent $\mathbf{z}_t^k$. For each current handle point $\mathbf{h}_i^k$, motion supervision encourages the feature patch around $\mathbf{h}_i^k$ to move toward $\mathbf{g}_i$. The local motion vector at iteration k is defined as $\mathbf{d}_i^k = \beta(\mathbf{g}_i - \mathbf{h}_i^k)/\|\mathbf{g}_i - \mathbf{h}_i^k\|_2$, where β controls the step size. The optimization objective is then defined as:

$$\mathcal{L}_{ms}^k(\mathbf{z}_t^k) = \sum_{i=1}^{N} \sum_{\mathbf{q}\in\Omega(\mathbf{h}_i^k, r_1)} \left\| F_{\mathbf{q}+\mathbf{d}_i^k}(\mathbf{z}_t^k) - \mathrm{sg}\left[F_\mathbf{q}(\mathbf{z}_t^k)\right] \right\|_1 + \lambda \left\| (\mathbf{1} - M) \odot \left(\mathbf{z}_{t-1}^k - \mathbf{z}_{t-1}^{\mathrm{ref}}\right) \right\|_1 \tag{15}$$

In this equation, $\Omega(\mathbf{h}_i^k, r_1) = \{\mathbf{q} \mid \|\mathbf{q} - \mathbf{h}_i^k\|_\infty \le r_1\}$ is a local region. The operator $\mathrm{sg}[\cdot]$ stops gradients on the source feature. The latent $\mathbf{z}_{t-1}^k$ is denoised from $\mathbf{z}_t^k$, and $\mathbf{z}_{t-1}^{\mathrm{ref}}$ is the fixed latent from the unoptimized DDIM trajectory. The binary editable mask M satisfies $M(\mathbf{q}) = 1$ in the editable region and $M(\mathbf{q}) = 0$ elsewhere, and λ weights background preservation.

Point Tracking. After motion supervision, the tracked handle point may not be spatially aligned with the target point. Point tracking therefore estimates $\mathbf{h}_i^{k+1}$ by matching features in the local region around $\mathbf{h}_i^k$:

$$\mathbf{h}_i^{k+1} = \underset{\mathbf{q}\in\Omega(\mathbf{h}_i^k, r_2)}{\operatorname{argmin}} \left\| F_{\mathbf{q}}(\mathbf{z}_t^{k+1}) - F_{\mathbf{h}_i^0}(\mathbf{z}_t^{\mathrm{ref}}) \right\|_1. \tag{16}$$

The set $\Omega(\mathbf{h}_i^k, r_2) = \{\mathbf{q} \mid \|\mathbf{q} - \mathbf{h}_i^k\|_\infty \le r_2\}$ defines the tracking region. The minimization selects the point whose current feature is closest to the reference feature at the initial handle position. This update provides the handle location for the next iteration, allowing motion supervision and point tracking to progressively move the handle points toward the targets.

B Implementation Details

B.1 Mask-Boundary LWF

Following FastDrag (Zhao et al., 2024), CannyDrag assigns each editable pixel an inverse-distance weight for blending the displacements of multiple handle points. For a pixel $\mathbf{p} \in M$, the normalized weight of the i-th handle point is

$$w_i(\mathbf{p}) = \frac{\|\mathbf{p} - \mathbf{h}_i\|_2^{-1}}{\sum_{j=1}^{N} \|\mathbf{p} - \mathbf{h}_j\|_2^{-1}}. \tag{17}$$

Thus, the displacement at $\mathbf{p}$ is obtained by combining the handle displacements according to the same spatial weighting rule. The difference lies in the attenuation factor $\lambda_i(\mathbf{p})$. FastDrag estimates this factor from the intersection of the ray from $\mathbf{h}_i$ through $\mathbf{p}$ with a circumscribed circle around the editable region (Zhao et al., 2024). CannyDrag instead follows this ray until it exits the actual editable mask; the exit point is denoted by $\mathbf{q}_i(\mathbf{p})$. The mask-aware attenuation factor is then

$$\lambda_i(\mathbf{p}) = \frac{\|\mathbf{q}_i(\mathbf{p}) - \mathbf{p}\|_2}{\|\mathbf{q}_i(\mathbf{p}) - \mathbf{h}_i\|_2}. \tag{18}$$

This replacement adapts the displacement decay to irregular mask boundaries: pixels near a handle receive a larger portion of its displacement, while pixels close to the actual boundary receive less. By contrast, the circumscribed-circle approximation may assign inappropriate attenuation to boundary-near pixels, causing excessive or insufficient displacement for irregular editable regions.

C Experimental Details

Implementation settings All experiments are built on Stable Diffusion v1.5 (Rombach et al., 2022) with a ControlNet branch (Zhang et al., 2023) for canny-conditioned denoising. The canny map is extracted by the Canny edge detector with low and high thresholds of 100 and 200 for hysteresis edge linking. Before inference, LoRA fine-tuning is performed for each input image with rank 16, learning rate 5×10^{-4}, and batch size 4. We use 50 denoising steps and set the inversion strength to 0.75. For drag optimization, the latent learning rate is 0.02, the background weight is $\lambda = 0.1$, the motion supervision radius is $r_1 = 4$, the tracking radius is $r_2 = 12$, and point tracking is performed every 10 optimization iterations. For point-level drag editing, latent optimization and progressive canny manipulation share a common step count, with $T_{\mathrm{c}} = 7$ dragging steps and 70 LoRA fine-tuning steps. For line-level drag editing, we use $T_{\mathrm{c}} = 15$ dragging steps and 80 LoRA fine-tuning steps. For region-level drag editing, we use $T_{\mathrm{c}} = 10$ dragging steps and 100 LoRA fine-tuning steps.

Benchmark construction For point-level dragging, each sample follows the standard DragBench handle-target point-pair format. For line-level dragging, the annotation additionally contains a line-area mask identifying the pixels belonging to the dragged line structure, together with control points and their target positions. For region-level dragging, the annotation contains a closed region-area mask and control points defining the expected similarity transformation. The editable mask M is retained for background preservation, while the line-area mask A_{line} and region-area mask A_{region} specify the structure to be manipulated. To evaluate the whole dragged structure rather than only

sparse point controls, DragBench-Line samples 5 auxiliary point pairs uniformly along the centerline arc length of A_{line}. DragBench-Region samples 10 auxiliary point pairs inside A_{region} using coverage-based farthest-point sampling and maps them with the corresponding geometric transformation.

Evaluation Settings Spatial accuracy and local preservation are evaluated by mean distance (MD) and the Dragging Accuracy Index (DAI), while image fidelity is measured by Image Fidelity (IF), defined as $1 - \text{LPIPS}$. The detected point is obtained by matching the DIFT feature (Tang et al., 2023) of each source handle point in the edited image, and the metrics are computed from its distance to the corresponding target point and from local image changes. Point-level evaluation uses the original DragBench point pairs, whereas line- and region-level evaluation uses the auxiliary point pairs to cover the annotated dragged area more comprehensively. We compare against DragDiffusion (Shi et al., 2024), FastDrag (Zhao et al., 2024), DragNoise (Liu et al., 2024), GoodDrag (Zhang et al., 2025d), and GeoDrag (Pu et al., 2026).

D Additional Qualitative Results

Figure 8 provides additional qualitative comparisons across the three drag-editing tasks. Across these examples, CannyDrag demonstrates improved spatial precision, source-structure preservation, and background consistency.

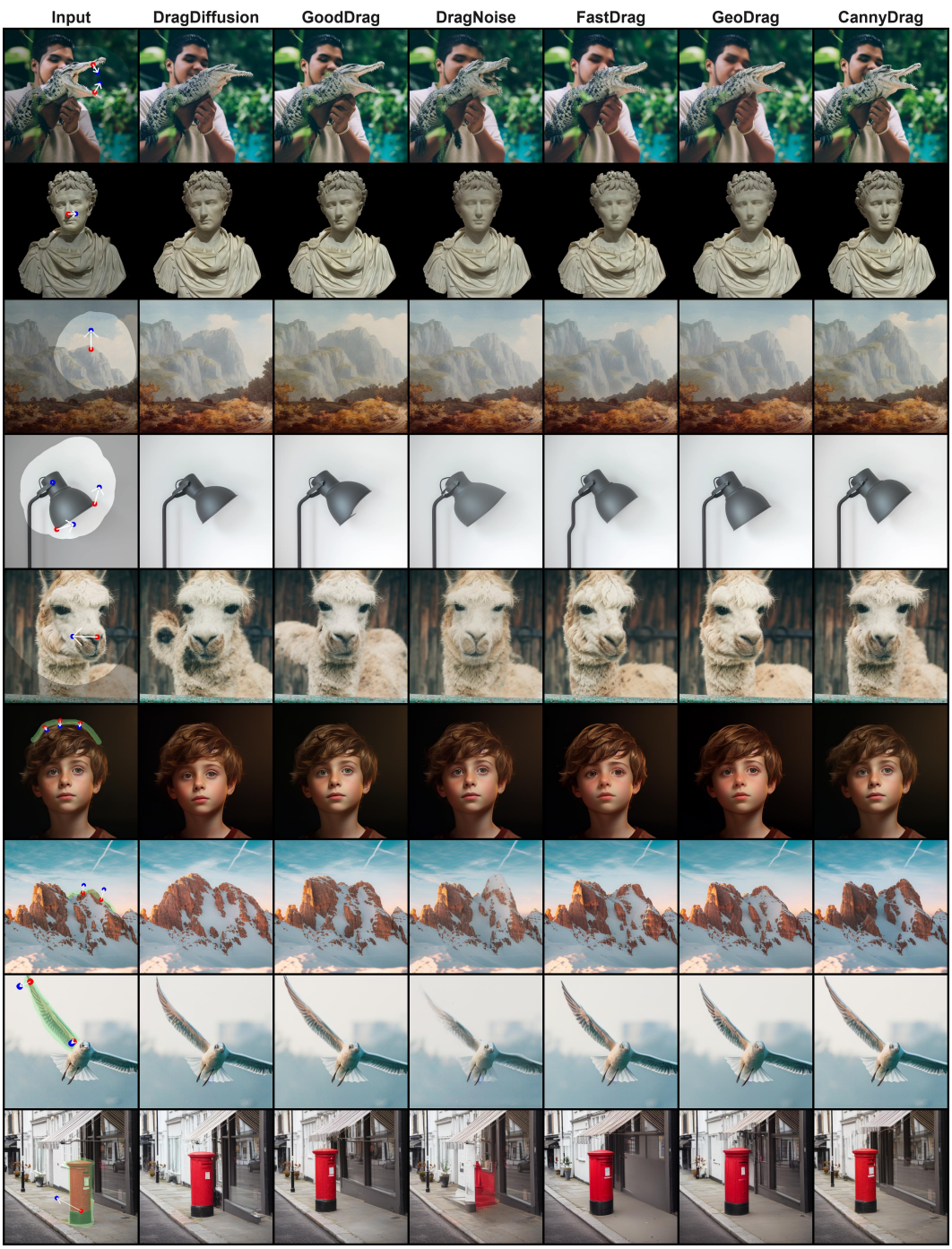


Figure 8: Additional qualitative comparisons across point-level drag editing (rows 1–5), line-level curve bending (rows 6–7), and region-level similarity transformation (rows 8–9).