POINT-CONDITIONED DRAG EDITING FOR IN-CONTEXT DIFFUSION TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Point based drag editing enables intuitive spatial control by allowing users to specify sparse source-target point pairs. However, converting such sparse geometric directives into coherent image edits remains challenging, especially when the edited content must move accurately while preserving identity, background, and local visual details. Existing drag editing methods often rely on iterative point tracking, handcrafted motion propagation, region level assumptions, or mask guided inpainting, which can introduce additional user burden and limit generalization to complex motion. We propose PiDiT, a point conditioned drag editing framework built on an incontext diffusion transformer. PiDiT renders sparse drag pairs into spatial point maps, encodes them with a lightweight point encoder, and injects the resulting token aligned point features directly into the transformer's hidden states. This design treats point pairs as internal spatial conditions for the target generation branch while preserving the source image as incontext visual evidence. The framework is trained with a flow matching objective and requires only the source image and sparse point pairs at inference time, without user provided masks or test-time optimization. Experiments on DragBench and ReD-Bench show that PiDiT improves point reachability while maintaining competitive visual fidelity and identity preservation.

1 INTRODUCTION

Point-based drag image editing offers an intuitive interface for spatial control: users specify sparse handle-target point pairs, and the model moves the intended content accordingly. As shown in Figure 1, the same interface can describe geometric controls, such as scaling, rotation, and translation, as well as non-rigid edits, such as pose change and local deformation. The core challenge is to follow these sparse spatial constraints while preserving identity, background, and local visual details.

Existing drag editing methods can be grouped according to their core mechanisms. A first group, represented by DragGAN (Pan et al., 2023), DragDiffusion (Shi et al., 2024), and GoodDrag (Zhang et al., 2025), alternates motion supervision with point tracking to iteratively optimize image or diffusion latents. Since point tracking only follows local neighborhoods around sparse handle points, these methods struggle to infer the global semantic change implied by a drag instruction. A second group reduces this dependence by introducing explicit propagation priors. FastDrag (Zhao et al., 2024) uses a predefined latent warpage function followed by latent relocation, while DragNeXt (Zhou et al., 2026) casts editing as latent region optimization, where regions and transformation types specified by users provide geometric intervention signals. Although these designs improve efficiency or structural guidance, their priors over motion, regions, and operation types are still difficult to generalize across diverse drag intents and scene structures. A third group, built on masks or inpainting, such as Inpaint4Drag (Lu & Han, 2025) and LightningDrag (Shi et al., 2025), requires editable masks and synthesizes edited regions through warping or inpainting backbones. This mask requirement weakens the simple interaction of point control and can introduce boundary artifacts, identity inconsistency, and local detail mismatch.

Recent generative backbones have shifted toward diffusion transformers and flow matching (Peebles & Xie, 2023; Lipman et al., 2023). In-context editing systems such as OminiControl (Tan et al., 2025) and FLUX.1 Kontext (Labs et al., 2025) show that a source image can serve as visual context for conditional generation. This raises a natural question: can sparse drag control be learned as an

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.

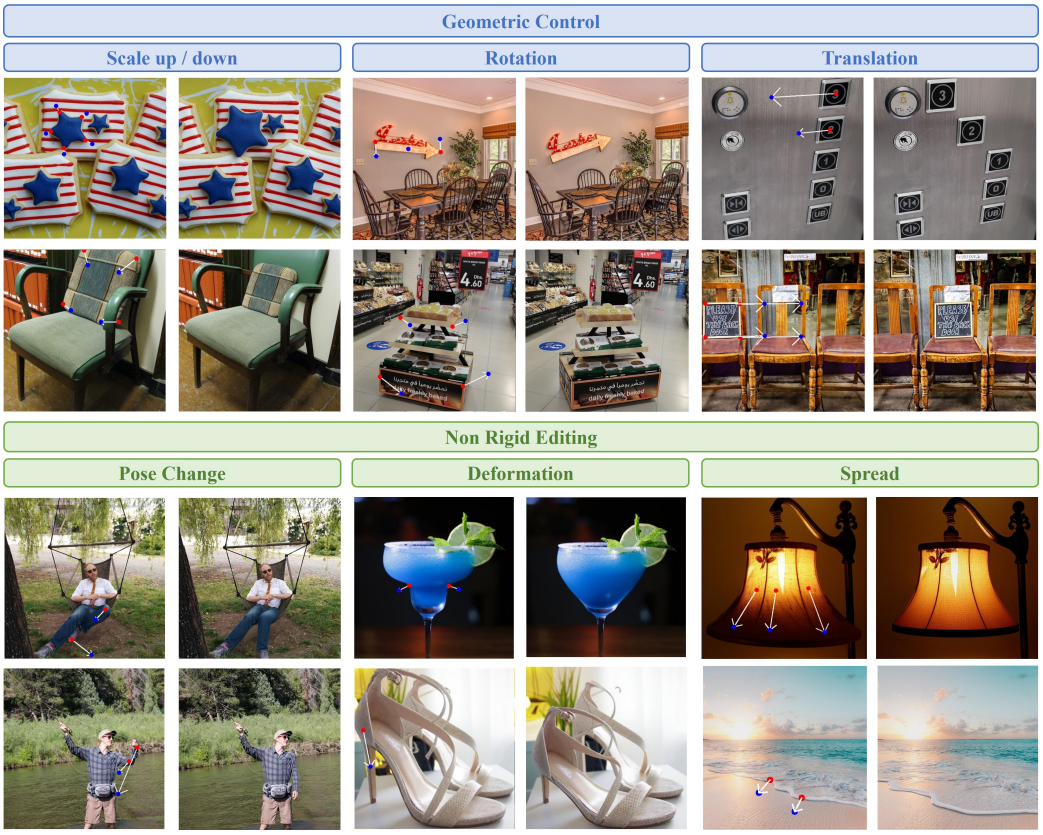

Figure 1: Qualitative overview of PiDiT's point-conditioned editing ability. The upper block shows geometric control, including scale up/down, rotation, and translation, while the lower block shows non-rigid editing, including pose change, deformation, and spread. Red points mark handle locations, blue points mark target locations, and white arrows indicate the requested motion.

internal condition for an in-context diffusion transformer, without test-time point tracking or user-provided masks?

We propose PiDiT, a point-conditioned drag-editing framework that reformulates sparse drag control as conditional generation within an in-context Diffusion Transformer backbone. PiDiT maps sparse point pairs into spatial point maps, encodes them into token-aligned point features, and injects these features directly into the DiT hidden states. The source image remains as in-context visual evidence, while the injected point features provide localized motion guidance for the target branch. This design enables mask-free and optimization-free drag editing with accurate point control and stronger identity preservation.

Our contributions can be summarized as follows:

- We reformulate point-conditioned drag editing as a conditional generation task within an in-context Diffusion Transformer (DiT), and propose a tailored pipeline that injects token-aligned point features into DiT hidden states.
- Extensive evaluations on DragBench and ReD-Bench show that PiDiT improves point reachability and visual quality without requiring user-provided masks or test-time optimization.

2 Related Work

Explicit Motion Modeling and Training-Free Control DragGAN (Pan et al., 2023) pioneered interactive editing through latent optimization with point tracking and motion supervision, and

DragDiffusion (Shi et al., 2024) extended this paradigm to latent diffusion models. Later variants improve controllability, stability, or efficiency through alternative motion propagation, stochastic editing, semantic noise propagation, and point guidance (Mou et al., 2024; Ling et al., 2024; Nie et al., 2024; Liu et al., 2024; Cui et al., 2024). FastDrag (Zhao et al., 2024), GoodDrag (Zhang et al., 2025), DragNeXt (Zhou et al., 2026), Inpaint4Drag (Lu & Han, 2025), and DragFlow (Zhou et al., 2025) further address speed, structural consistency, disocclusion, or region-level control. These methods reduce different failure modes, but typically rely on test-time optimization, explicit geometry, region assumptions, or hybrid warping-inpainting pipelines. Recent training-free DiT methods such as LazyDrag (Yin et al., 2025) and ContextDrag (He et al., 2025) instead manipulate correspondences, attention, or tokens at inference time. In contrast, PiDiT studies learned point conditioning inside an in-context diffusion transformer.

Point Conditions for Drag Editing An emerging line of research replaces iterative optimization with learned conditioning. InstantDrag (Shin et al., 2024) predicts dense optical flow from sparse drag inputs before flow-conditioned synthesis, while LightningDrag (Shi et al., 2025) converts sparse points into condition maps and attention-level guidance with IP-Adapter (Ye et al., 2023). Unlike these methods, PiDiT injects sparse point-pair features directly into the hidden states of an in-context diffusion transformer, without dense motion intermediates or inpainting-style condition maps.

3 Preliminaries

3.1 Diffusion and Flow Matching

Diffusion-based generative models (Ho et al., 2020) learn to transform a simple noise distribution into the data distribution through a time-dependent denoising process. In latent-space image generation, an image is first encoded into latent tokens, and the generative model operates on these tokens rather than directly on pixels. A common flow-matching formulation defines a linear path between a clean latent z_0 and Gaussian noise ϵ. For a time step $t \in [0, 1]$, the intermediate latent is

$$z_t = (1 - t)z_0 + t\epsilon, \tag{1}$$

and the model is trained to predict the velocity along this path,

$$\mathcal{L}_{\text{fm}} = \|v_\theta(z_t, t, c) - (\epsilon - z_0)\|_2^2, \tag{2}$$

where c denotes optional conditioning information. At inference time, the model starts from noise and follows the predicted velocity field to obtain an image latent.

3.2 In-Context Image Editing

In-context image editing treats the source image as visual context for generating the edited target image. Let z^{src} denote source-image latent tokens and z^{tgt} denote target-image latent tokens. At timestep t, the noisy target latent is denoted as z_t^{tgt}. The editing transformer takes the concatenation of the noisy target tokens and the source tokens as input:

$$z_t = [z_t^{tgt}, z^{src}], \tag{3}$$

The model then predicts the velocity of the target branch conditioned on this concatenated visual context and optional editing instructions. In our setting, sparse drag points are introduced as additional internal conditions for this in-context editing process.

4 Method

4.1 Problem Formulation

Given a source image I^{src} and paired handle target points $\mathcal{P} = \{(p_i^{src}, p_i^{tgt})\}_{i=1}^N$, drag editing aims to generate an edited image I^{edit} that moves the content around each handle point toward its corresponding target while preserving unrelated content. At inference time, PiDiT receives only the source image and sparse point pairs, without user-provided masks.

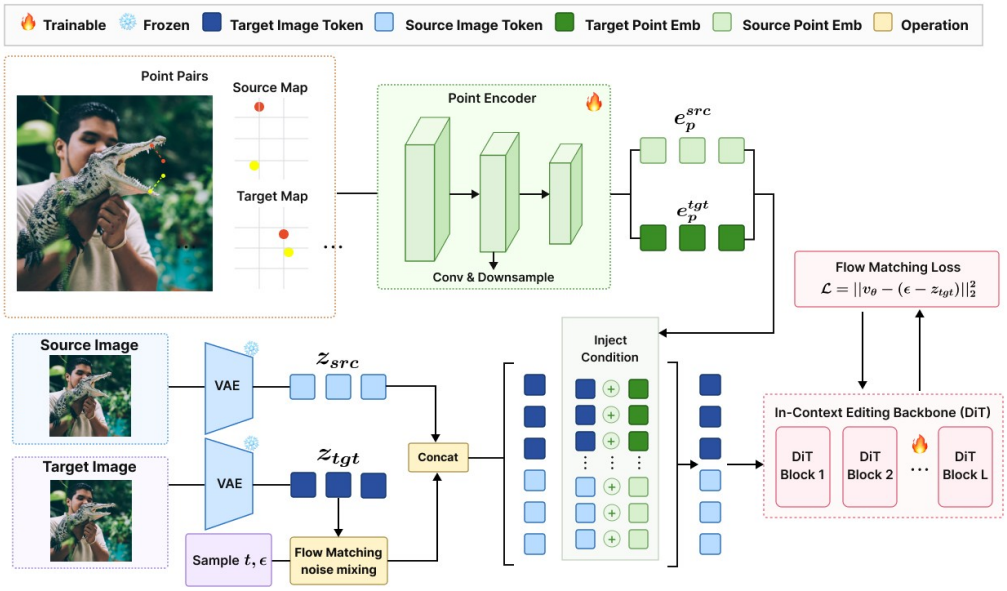

Figure 2: Overall framework of PiDiT. Sparse source-target point pairs are rendered into point maps, encoded into token-aligned point features, and concatenated in the same target-source order as the in-context visual tokens. The point features are added to the hidden states before the DiT backbone, enabling point-conditioned flow matching without user-provided masks.

4.2 PiDiT Framework

Figure 2 provides an overview of PiDiT. PiDiT is built on in-context image editing DiT backbone, activating the drag-based editing ability by introducing sparse point-pair conditions. The method consists of two main components. First, we leverage the pre-trained in-context image editing model as a frozen backbone to preserve the background, identity, and visual consistency of the source image. Second, we design a point encoder to extract features from source-target point pairs and inject these features directly into the hidden states of the transformer. By doing so, we treat sparse point directives as internal conditions that guide the in-context image editing process to perform structure-aware geometric transformations.

4.3 In-Context Editing Backbone

The in-context editing backbone takes both the noisy target branch and the source branch as visual tokens. Let z^{src} denote the source latent tokens and z^{tgt} denote the clean target latent tokens. At flow-matching timestep t, the noisy target branch is

$$z_t^{tgt} = (1-t)z^{tgt} + t\epsilon. \tag{4}$$

The visual input is constructed by concatenating the target and source branches along the sequence dimension,

$$z_t = [z_t^{tgt}, z^{src}]. \tag{5}$$

In this sequence, the target tokens are responsible for generation, while the source tokens provide image context. Since both branches are processed within the same transformer, their tokens interact through attention, allowing the model to preserve source-image appearance while generating the edited target branch.

4.4 Point Encoding and Hidden-State Injection

Treating drag points as model conditions requires answering two questions: how to extract point-pair features, and how to use them inside the editing backbone. Since point annotations are spatially aligned with the image plane, we render them as image-sized maps rather than encode them only as

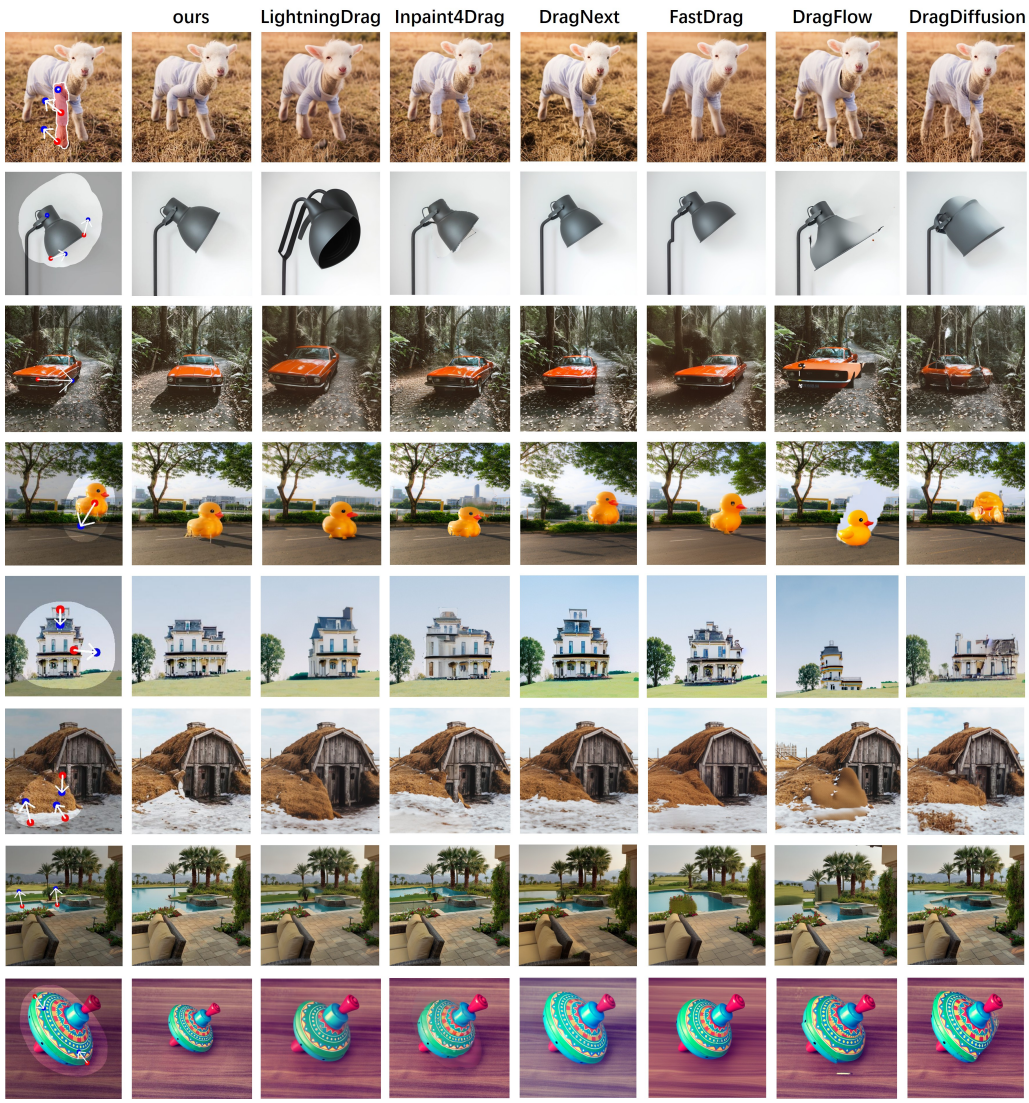

Figure 3: Qualitative comparison with other methods. Our method demonstrates improved identity consistency and visual fidelity, while maintaining a higher editing success rate.

a short coordinate list. Let $M_p^{src}, M_p^{tgt} \in \mathbb{R}^{H\times W}$ denote the source and target point maps. Because the point maps are defined on a discrete pixel grid, continuous drag coordinates are rasterized by rounding each coordinate to the nearest integer pixel. Specifically, for a point $p = (x, y)$, we use $\tilde{p} = (\mathrm{round}(x), \mathrm{round}(y))$ when constructing M_p^{src} and M_p^{tgt}; in the equations below, p_i^{src} and p_i^{tgt} denote these rasterized coordinates. This discretization is applied consistently to both training annotations and inference-time drag inputs.

For a point pair (p_i^{src}, p_i^{tgt}), where $p_i^{src} = (x_i^{src}, y_i^{src})$, $p_i^{tgt} = (x_i^{tgt}, y_i^{tgt})$, and $i \in \{1, \ldots, N\}$, the maps are defined as

$$M_p^{src}(x,y) = \begin{cases} i, & (x,y) = (x_i^{src}, y_i^{src}), \\ 0, & \text{otherwise}, \end{cases} \tag{6}$$

$$M_p^{tgt}(x,y) = \begin{cases} i, & (x,y) = (x_i^{tgt}, y_i^{tgt}), \\ 0, & \text{otherwise}, \end{cases} \tag{7}$$

where (x, y) indexes the $H \times W$ image grid. The same identifier is assigned to the source and target locations of each pair, preserving their correspondence, while background pixels are assigned zero.

The point maps are encoded by a convolutional encoder following a VAE-like architecture. For the default single channel point map, the encoder first uses a 3×3 convolution to map the input to 128 channels, followed by four resolution stages with channel widths $(128, 256, 512, 512)$, each containing two residual blocks. The first three stages are followed by stride 2 convolutional downsampling, resulting in an overall $8\times$ spatial reduction. The downsampling factor is consistent with that of the VAE, facilitating alignment with the VAE feature space. A final 3×3 convolution produces a 16 channel feature map. Although the input map is highly sparse, each nonzero point marker can influence a local neighborhood of feature activations through the convolutional receptive fields. During progressive downsampling, this localized response is aggregated into nearby token features rather than remaining an isolated single pixel signal.

For a 512×512 input, this produces a $16 \times 64 \times 64$ feature map, which is rearranged into a 32×32 grid of 64 dimensional tokens. Thus, the resulting token grid has an effective stride of 16 pixels with respect to the input image. We denote the resulting token-aligned point features as

$$e_p^{tgt} = \Phi_\psi(M_p^{tgt}), \quad e_p^{src} = \Phi_\psi(M_p^{src}). \tag{8}$$

The final point condition is concatenated in the same order as the in-context visual input,

$$e_p = [e_p^{tgt}, e_p^{src}], \tag{9}$$

matching $z_t = [z_t^{tgt}, z^{src}]$. Thus each image token has an aligned point feature.

The injected point feature is spatially sparse and token-aligned. It is generated from source and target point maps and then downsampled to the same latent grid as the visual tokens, so nonzero or high-response features are concentrated around the user-specified handle and target locations. Therefore, the injection does not introduce an unconstrained perturbation over the entire image representation. Instead, it provides localized spatial evidence about where motion should originate and where it should be directed, allowing the remaining tokens to continue relying on the original source-image context.

By injecting these features into the backbone's intermediate hidden states, we treat the point directives as dynamic spatial priors. Specifically, the point features act as local visual residuals on the token grid, modulating the model's latent representation. From a flow-matching perspective, this injects a point-dependent perturbation into the predicted velocity field, modifying $v_\theta(z_t, t, c)$ to $v_\theta(z_t, t, c, e_p)$, while preserving the backbone's original attention structure and identity-preserving capabilities.

4.5 Training and Inference

PiDiT is trained with the same flow-matching target as the in-context editing backbone, while conditioning the velocity predictor on the point feature e_p:

$$\mathcal{L}_{\text{PiDiT}} = \left\| v_\theta(z_t, t, c, e_p) - (\epsilon - z^{tgt}) \right\|_2^2, \tag{10}$$

where c denotes text conditioning. In our current experiments, we use empty text prompts to isolate the effect of the point condition.

At inference time, the source image is encoded once as z^{src}. The target branch is initialized from Gaussian noise and concatenated with the source branch at each denoising step. The same point condition e_p is injected into the image-token hidden states throughout generation. After the flow trajectory is completed, the target branch is decoded to obtain the edited image I^{edit}.

Notably, PiDiT retains the original instruction based editing capabilities, enabling joint drag-and-text editing by conditioning the velocity predictor on both text prompts c and point features e_p.

5 Experiments

5.1 Implementation Details

We train PiDiT on a compact yet distribution-balanced drag dataset containing over 2K annotated training pairs. Rather than increasing the dataset size indiscriminately, we curate the data to cover

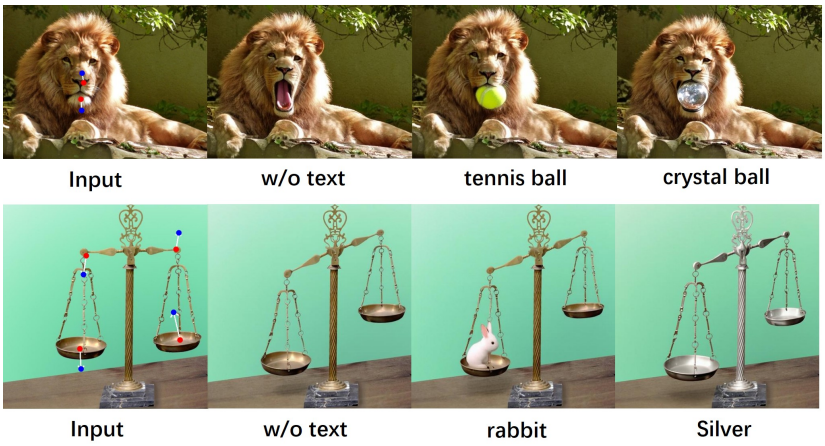

Figure 4: Joint drag-and-text editing with PiDiT. The examples illustrate that point conditions and text instructions are jointly consumed.

Table 1: Quantitative comparison with existing drag-editing methods on DragBench.

Method	LPIPS ↓	MD ↓	G_PQ ↑
Inpaint4Drag	**0.0904**	22.7127	7.6683
DragNeXt	0.1482	46.83	<u>8.7318</u>
LightningDrag	0.1645	<u>18.6208</u>	8.5508
DragFlow	0.4622	100.8929	3.5733
FastDrag	0.1418	30.9899	8.7266
DragDiffusion	<u>0.1188</u>	33.1146	8.6805
PiDiT (ours)	0.1455	**17.0644**	**8.7352**

diverse motion and content distributions, including translation, rotation, scale change, pose change, non-rigid deformation, and both local and object-level manipulations. This design prevents the training set from being dominated by simple motions or a narrow set of object categories. The data come from two sources: annotated frame pairs sampled from collected videos, and image-editing results generated from collected images with constructed editing instructions followed by point annotation. The model is trained for 55K optimization steps on two NVIDIA RTX A6000 GPUs with a learning rate of 2×10^{-4}. During training, the VAE, CLIP encoder, and T5 encoder are frozen. We optimize the point-map encoder and LoRA parameters (Hu et al., 2022) inserted into the in-context DiT backbone. To prevent the model from overfitting to a fixed point order, we randomly shuffle the order of point pairs during training while preserving each source-target correspondence. At inference time, we use 28 denoising steps and classifier-free guidance (Ho & Salimans) with a guidance scale of 3.5.

5.2 Quantitative Comparison

We evaluate PiDiT on DragBench and ReD-Bench. DragBench (Shi et al., 2024) covers diverse point-based editing cases, while ReD-Bench (Zhou et al., 2025) groups region-aligned operations such as relocation, deformation, and rotation. For baselines that require editable masks, we use the official benchmark masks, while PiDiT takes only the source image and handle-target point pairs as input. We report mean distance (MD) based on DIFT correspondence (Tang et al., 2023), LPIPS (Zhang et al., 2018), and VIEScore (Ku et al., 2024), denoted as G_PQ. Lower MD indicates better point reachability; lower LPIPS measures source similarity but may favor conservative or mask-assisted edits; higher G_PQ indicates better visual quality.

Table 1 compares PiDiT with representative drag-editing baselines on DragBench. PiDiT achieves the best MD and G_PQ, showing improved point reachability and strong perceptual quality. Although Inpaint4Drag obtains the lowest LPIPS, its edited regions often suffer from unacceptable

Table 2: Per-operation quantitative comparison on ReD-Bench. Samples are grouped by the task labels in the annotation files.

	Rotation (46)			Deformation (36)			Transformation (38)		
Method	LPIPS ↓	MD ↓	G_PQ ↑	LPIPS ↓	MD ↓	G_PQ ↑	LPIPS ↓	MD ↓	G_PQ ↑
Inpaint4Drag	0.1081	16.7035	6.6087	**0.0563**	30.2635	7.8889	0.1322	21.1338	7.3421
DragNeXt	0.0912	40.6782	8.5435	0.1079	39.5759	8.8889	0.1427	51.4691	8.6842
LightningDrag	0.1034	26.1837	8.4130	0.0718	36.4319	8.8444	0.1146	51.5155	8.3947
DragFlow	0.1309	63.3508	4.3478	0.0605	44.6137	4.2222	0.0937	70.7621	4.7368
FastDrag	0.1034	40.8433	8.4348	0.0869	38.8554	8.9167	0.1283	65.5788	8.6842
DragDiffusion	**0.0722**	38.8765	8.4348	0.0632	37.8428	8.8889	**0.0922**	73.1049	8.6579
PiDiT (ours)	0.1406	**16.5613**	**8.6117**	0.1184	**17.5508**	**8.9375**	0.1501	**19.5181**	**8.7105**

Table 3: Ablation study on PME input encodings on DragBench. Both variants use hidden-state injection. We report the best seed per sample selected by MD.

Point-map Encoding	LPIPS ↓	MD ↓
Sparse index map	**0.1455**	**17.0644**
Dense heatmap	0.1983	19.7755

distortion and inpainting artifacts, resulting in unnatural geometry and visual inconsistency. Table 2 reports the ReD-Bench comparison. PiDiT again achieves the best MD and G_PQ across operation types, whereas lower-LPIPS baselines often have substantially higher MD, indicating more conservative edits rather than successful regional motion.

5.3 Ablation on Point-Map Encoding

We ablate the input representation of the point map encoder (PME) while keeping the hidden state injection strategy fixed. Our default representation is a single channel sparse index map, where the source and target locations of the i-th point pair are assigned the value i and all other pixels are set to zero. We compare it with a Gaussian heatmap representation, which produces a smooth response around each point and provides a denser input to the convolutional PME.

As shown in Table 3, the sparse index map outperforms the Gaussian heatmap on both LPIPS and MD. This result indicates that the convolutional PME can propagate sparse point responses through its receptive fields and progressive downsampling without requiring a dense heatmap. The higher MD of the Gaussian variant may result from spatial smoothing: spreading each point response over a neighborhood can blur the location cue after convolutional encoding and downsampling, reducing target localization precision. We therefore use the single channel sparse index map as the default representation.

5.4 Ablation on Injection Strategy

Besides the hidden-state injection used in PiDiT, we also evaluate a QK injection variant. In this variant, point features are projected into query and key representations and added to the corresponding attention queries and keys, so that point conditions directly modulate token-to-token attention. As shown in Table 4, the two strategies achieve similar MD, indicating comparable point reachability. However, hidden-state injection obtains a clearly lower LPIPS, suggesting better overall image consistency. This trend is consistent with our qualitative observations, where QK injection tends to produce weaker source-image consistency. This indicates that adding point features to the query and key representations can introduce stronger disturbance to the source-image representation, while hidden-state injection provides spatial guidance with less disruption to the original in-context representation.

5.5 Training Dataset Scaling Analysis

Fig. 5 studies how training-data scale affects PiDiT. With smaller subsets, LPIPS remains relatively high, suggesting that limited data coverage makes it harder to preserve the global consistency while

Table 4: Ablation study on point-condition injection strategies on DragBench.

Injection Strategy	LPIPS ↓	MD ↓	G_PQ ↑
QK injection	0.1692	17.2702	8.7317
Hidden-state injection	0.1455	17.0644	8.7352

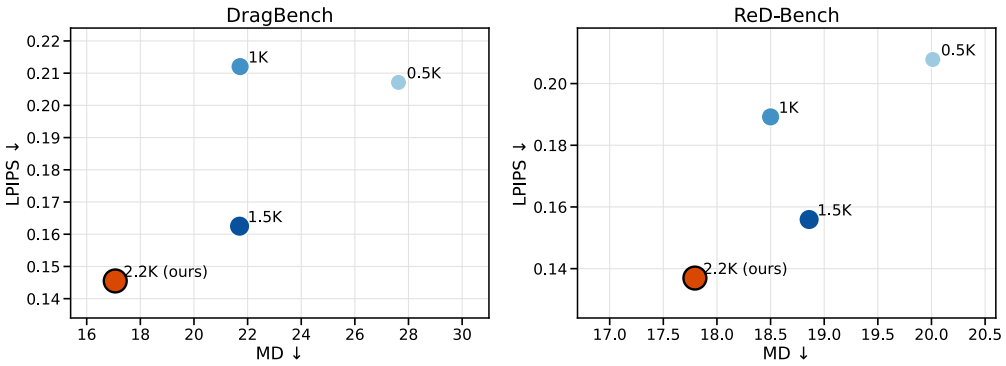

Figure 5: Training-data scaling analysis on DragBench and ReD-Bench. Each point represents a PiDiT model trained with a different number of annotated training pairs. The full 2.2K setting uses the whole training set.

following drag instructions. As the training set grows toward the full 2.2K curated set, LPIPS decreases clearly and MD also moves to a lower range on both DragBench and ReD-Bench, yielding a better balance between image fidelity and point reachability. This trend suggests that the 2.2K setting should be viewed not simply as a data-size limitation, but as a compact and distribution-balanced training regime that effectively activates PiDiT for robust drag editing.

5.6 Joint Drag-and-Text Editing

PiDiT also supports joint drag-and-text editing. At inference time, a non-empty text prompt c is provided together with the source image and point condition e_p within the same denoising trajectory. The point features specify where the content should move, while the text prompt controls the accompanying semantic or appearance change.

Figure 4 illustrates this capability. In the lion examples, identical drag points produce mouth opening or prompt-specific objects such as a tennis ball and a crystal ball. In the balance-scale examples, the point pairs control spatial rearrangement while the prompts alter the inserted content or material appearance. Thus, PiDiT performs spatial and semantic editing jointly without a separate drag-then-text pipeline.

6 Conclusion

We presented PiDiT, a point-conditioned drag-editing framework built on an in-context diffusion transformer. By rendering sparse source-target point pairs into spatial point maps and injecting token-aligned point features into the DiT hidden states, PiDiT turns sparse drag instructions into internal spatial conditions without requiring user-provided masks or test-time optimization. Experiments on DragBench and ReD-Bench show that PiDiT improves point reachability while preserving identity and maintaining competitive visual quality. We further show that the same inference process can combine point-based drag control with text instructions, enabling joint spatial and semantic editing within a single denoising trajectory.

AI USE STATEMENT

We used generative AI tools solely to polish the language and improve the clarity and readability of portions of the manuscript. We did not use generative AI tools for any research tasks requiring disclosure under the ICLR 2027 AI Policy, including developing the methodology, designing or conducting experiments, generating or cleaning datasets, interpreting results, formulating mathematical claims, implementing the method, or performing qualitative or statistical analysis. All AI-assisted language polishing was reviewed by the authors. We take full responsibility for the final content of this work, including all text, claims, results, and figures.

REFERENCES

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. Springer, 2024.

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

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*.

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

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

Max Ku, Dongfu Jiang, Cong Wei, Xiang Yue, and Wenhu Chen. Viescore: Towards explainable metrics for conditional image synthesis evaluation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12268–12290, 2024.

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

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 TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *11th International Conference on Learning Representations, ICLR 2023*, 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.

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.

Chong Mou, Xintao Wang, Jiechong Song, Ying Shan, and Jian Zhang. Dragondiffusion: Enabling drag-style manipulation on diffusion models. In *International Conference on Learning Representations*, volume 2024, pp. 31620–31631, 2024.

Shen Nie, Hanzhong Guo, Cheng Lu, Yuhao Zhou, Chenyu Zheng, and Chongxuan Li. The blessing of randomness: Sde beats ode in general diffusion-based image editing. In *International Conference on Learning Representations*, volume 2024, pp. 41763–41799, 2024.

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, 2023.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4195–4205, 2023.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pp. 8748–8763, 2021.

Yujun Shi, Chuhui Xue, Jun Hao Liew, Jiachun Pan, Hanshu Yan, Wenqing Zhang, Vincent YF Tan, and Song Bai. Dragdiffusion: Harnessing diffusion models for interactive point-based image editing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8839–8849, 2024.

Yujun Shi, Jun Hao Liew, Hanshu Yan, Vincent Tan, and Jiashi Feng. Lightningdrag: Lightning fast and accurate drag-based image editing emerging from videos. In *International Conference on Machine Learning*, pp. 54981–54993. PMLR, 2025.

Joonghyuk Shin, Daehyeon Choi, and Jaesik Park. Instantdrag: Improving interactivity in drag-based image editing. In *SIGGRAPH Asia 2024 conference papers*, pp. 1–10, 2024.

Zhenxiong Tan, Songhua Liu, Xingyi Yang, Qiaochu Xue, and Xinchao Wang. Ominicontrol: Minimal and universal control for diffusion transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 14940–14950, 2025.

Luming Tang, Menglin Jia, Qianqian Wang, Cheng Perng Phoo, and Bharath Hariharan. Emergent correspondence from image diffusion. *Advances in neural information processing systems*, 36: 1363–1389, 2023.

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.

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.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 586–595. IEEE, 2018. doi: 10.1109/CVPR.2018.00068. URL https://doi.org/10.1109/CVPR.2018.00068.

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*, volume 2025, pp. 13785–13808, 2025.

Xuanjia Zhao, Jian Guan, Congyi Fan, Dongli Xu, Youtian Lin, Haiwei Pan, and Pengming Feng. Fastdrag: Manipulate anything in one step. *Advances in Neural Information Processing Systems*, 37:74439–74460, 2024.

Yuan Zhou, Junbao Zhou, Qingshan Xu, Kesen Zhao, Yuxuan Wang, Hao Fei, Richang Hong, and Hanwang Zhang. Dragnext: Rethinking drag-based image editing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 13818–13825, 2026.

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.

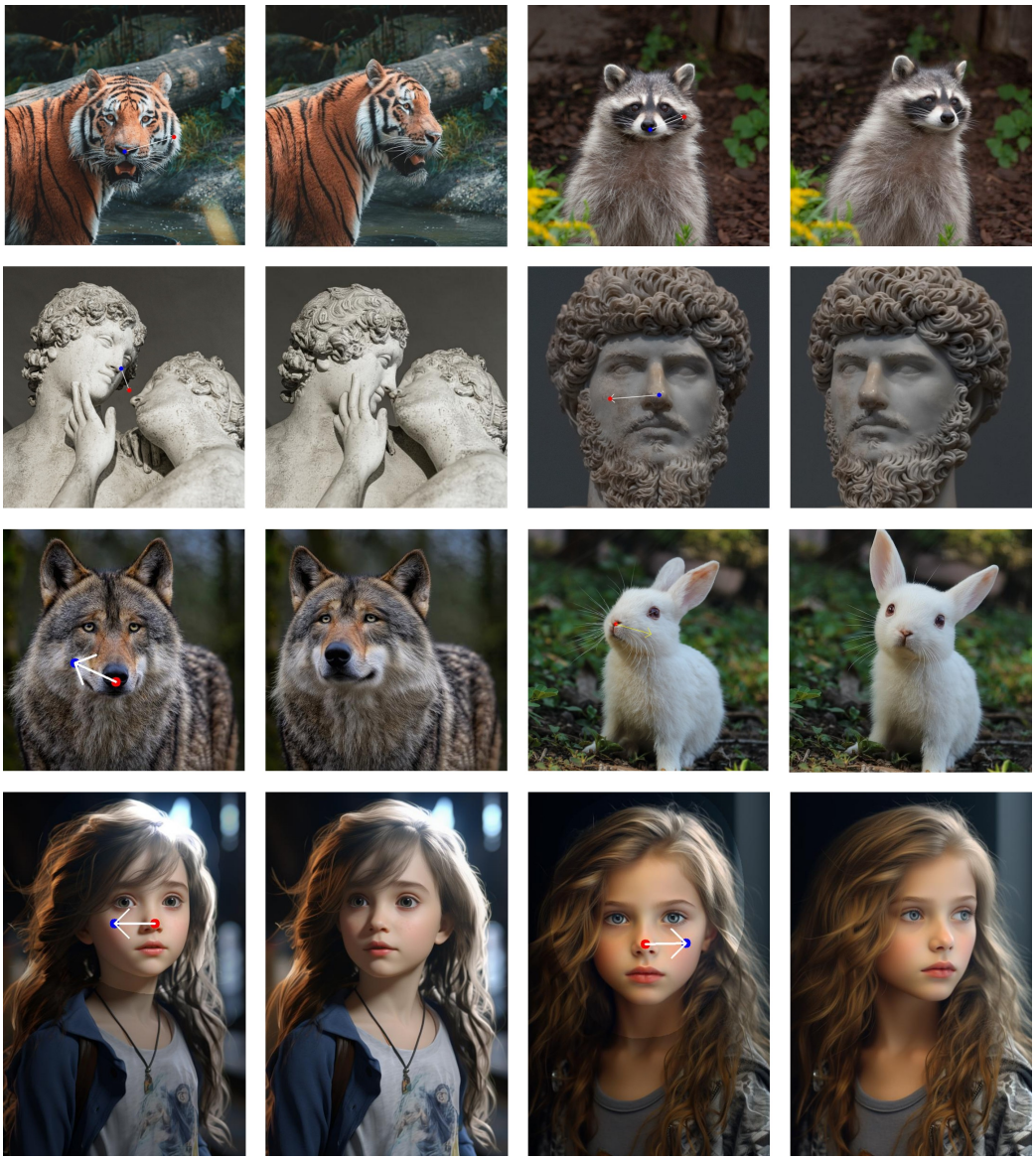

Figure 6: Additional qualitative result.

SUPPLEMENTARY MATERIAL

A ADDITIONAL QUALITATIVE RESULTS

We provide additional qualitative results to further evaluate PiDiT across diverse drag-editing scenarios in the media supplement. The examples cover geometric transformations, including translation, rotation, and scale change, as well as non-rigid manipulations, such as pose editing and local deformation. For all examples, red points denote handle locations, blue points denote target locations, and white arrows indicate the requested motion. Unless otherwise specified, PiDiT uses only the source image and sparse point pairs, without user-provided masks or test-time optimization.

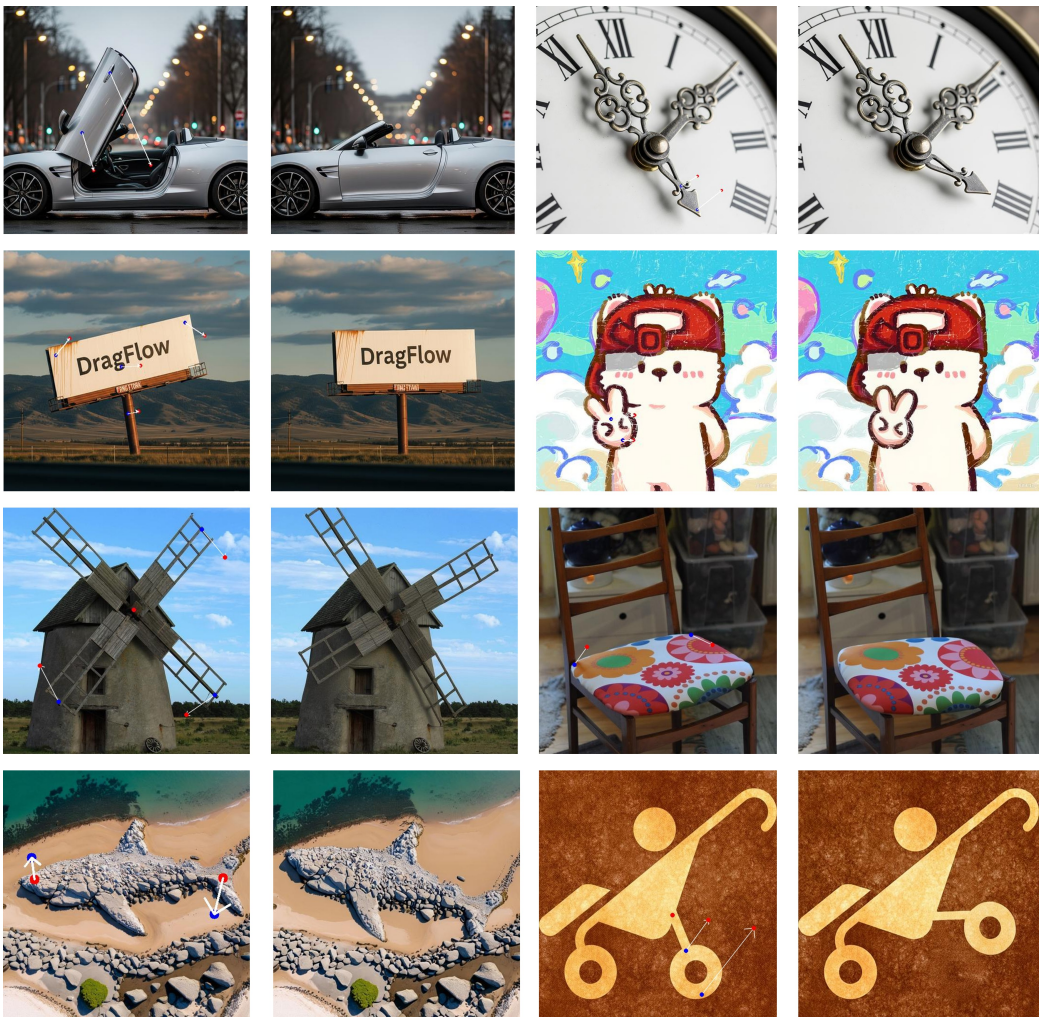

Figure 7: Additional qualitative result.

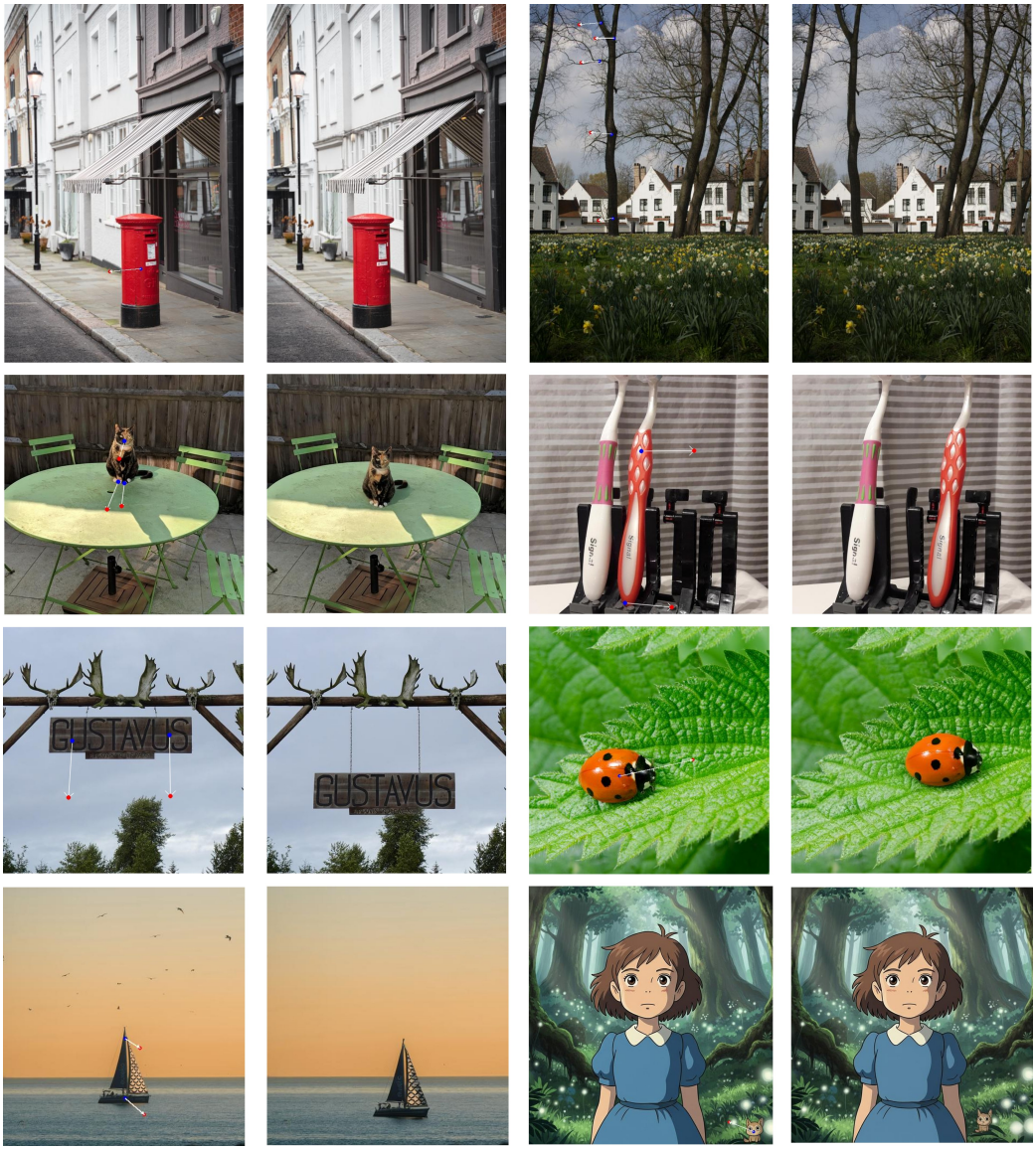

Figure 8: Additional qualitative result.

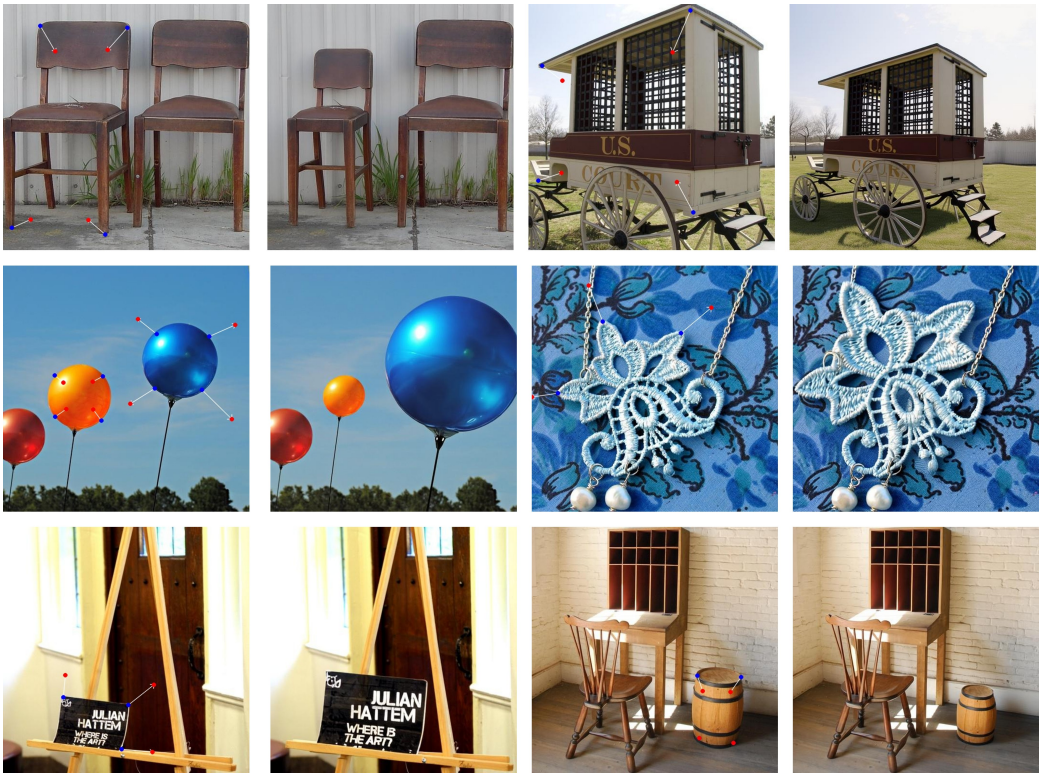

Figure 9: Additional qualitative result.



Figure 10: Additional qualitative result.

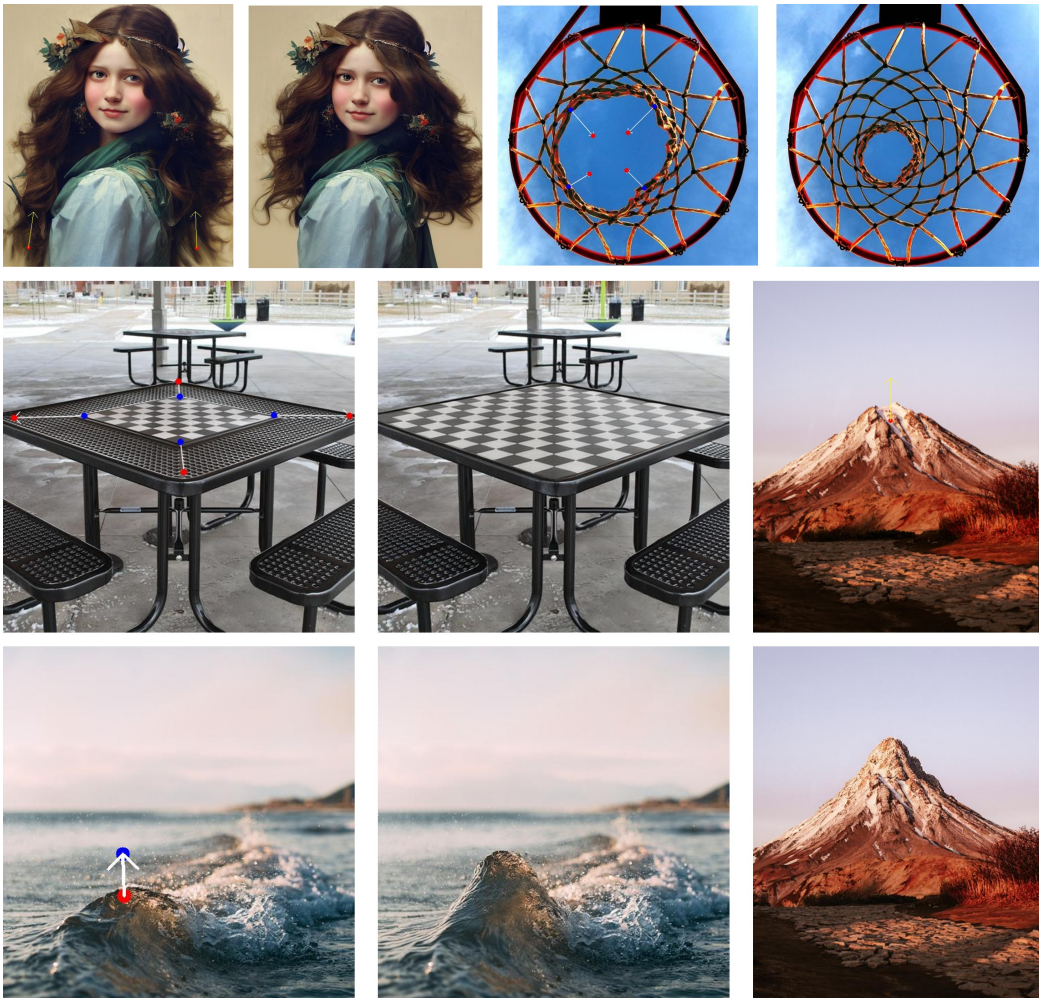

Figure 11: Additional qualitative result.



Figure 12: Additional qualitative result.