# Inpaint4Drag: Repurposing Inpainting Models for Drag-Based Image Editing via Bidirectional Warping
## – *Supplementary Material* –

## Contents

## S1. Supplementary Videos

Two supplementary videos are available on our project page `https://visual-ai.github.io/inpaint4drag`: one demonstrating our bidirectional warping algorithm with visualizations, and another showcasing the real-time user interface interaction.

## S2. Integration with More Inpainting Methods

We integrate our framework with diverse image inpainting approaches, from early methods like LaMa [4] and DeepFillv2 [5] to recent generative model-based techniques [2]. While quantitative metrics in Tab. S1 show comparable drag editing performance across methods, qualitative differences emerge in Fig. S1. Early approaches offer computational efficiency, whereas generative methods sometimes produce more realistic results–a quality distinction not fully captured by existing metrics. Our framework provides users flexibility to select the inpainting method best suited to their specific requirements, balancing computational resources and visual fidelity.

| Method | | | DragBench-S | | DragBench-D | |
|---|---|---|---|---|---|---|
| | Mem(GB)↓ | Time(s)↓ | MD↓ | LPIPS↓ | MD↓ | LPIPS↓ |
| DeepFillv2 [5] | **0.8** | **0.05** | **3.2** | 13.7 | **3.7** | 9.0 |
| LaMa [4] | 1.1 | 0.07 | 3.4 | 13.6 | 3.7 | 9.0 |
| SD-XL-Inpaint [2] | 8.1 | 1.3 | **3.2** | 12.5 | 3.8 | **8.8** |
| SD-1.5-Inpaint [2] | 2.7 | 0.3 | 3.6 | **11.4** | 3.9 | 9.1 |

Table S1. Comparison of different inpainting methods. MD and LPIPS values are scaled by 100. Time and GPU memory are measured at 512×512 resolution.

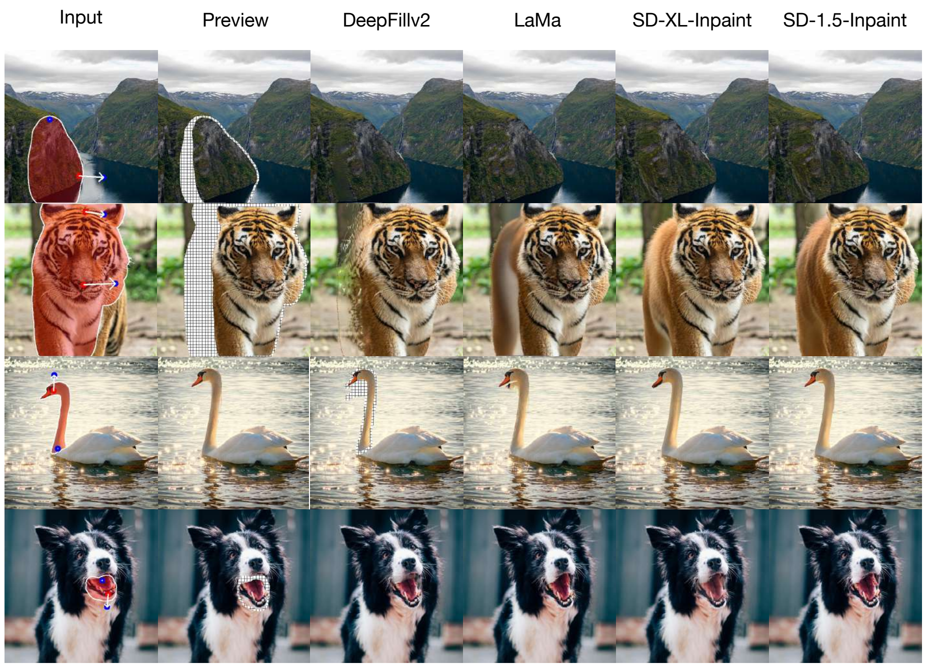

Figure S1. Qualitative comparison of different inpainting methods. The figure illustrates how various inpainting approaches affect drag editing results, highlighting differences in visual fidelity, artifact handling, and preservation of semantic content across traditional and generative model-based techniques.

## S3. Multi-round Interactive Editing

Our system enables fluid multi-round interactions, allowing users to execute sequential edits with minimal delay. In Fig. S2, we demonstrate this capability through a chess sequence where pieces are repositioned in rapid succession to create a checkmate scenario–highlighting our system's responsiveness to iterative user inputs.

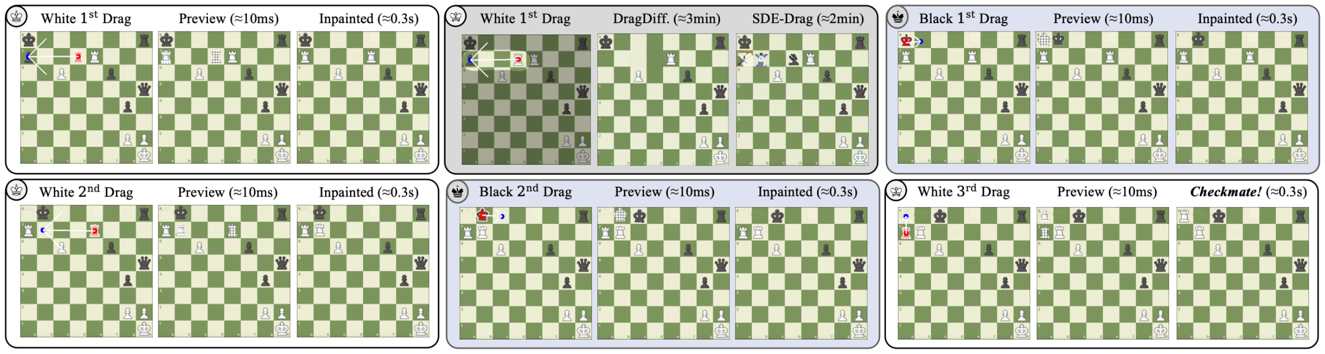

Figure S2. Multi-round interactive drag editing demonstrated through a three-move checkmate sequence including five consecutive edits. Users select deformable regions (chess pieces) and drag them from handle points to target positions. Grid overlays in the preview columns indicate areas requiring inpainting. Our method provides real-time preview ($\sim$10ms) of the warping effect, followed by high-quality inpainting results ($\sim$0.3s). Existing approaches typically require minutes for inference and fail during the initial interaction.

## S4. Discussion of Input Ambiguity

Previous drag editing methods [1, 3, 6] typically use sparse control points to guide deformation, with optional masking to restrict editable regions. However, this sparse input format (shown on the left of each row in Fig. S3) introduces fundamental ambiguity in deformation interpretation. Through our explicit region-based control (visualized in bottom-right insets), we demonstrate how a single ambiguous drag input can be precisely controlled to achieve five distinct editing intentions - from local manipulation to global translation. For instance, the same drag operation on a polar bear can be accurately interpreted as body translation, forearm bending, hand raising, upper body stretching, or scene translation. We address this ambiguity by requesting users to specify deformable regions through masking, treating each region as an elastic material where movement smoothly propagates from control points throughout the connected area.

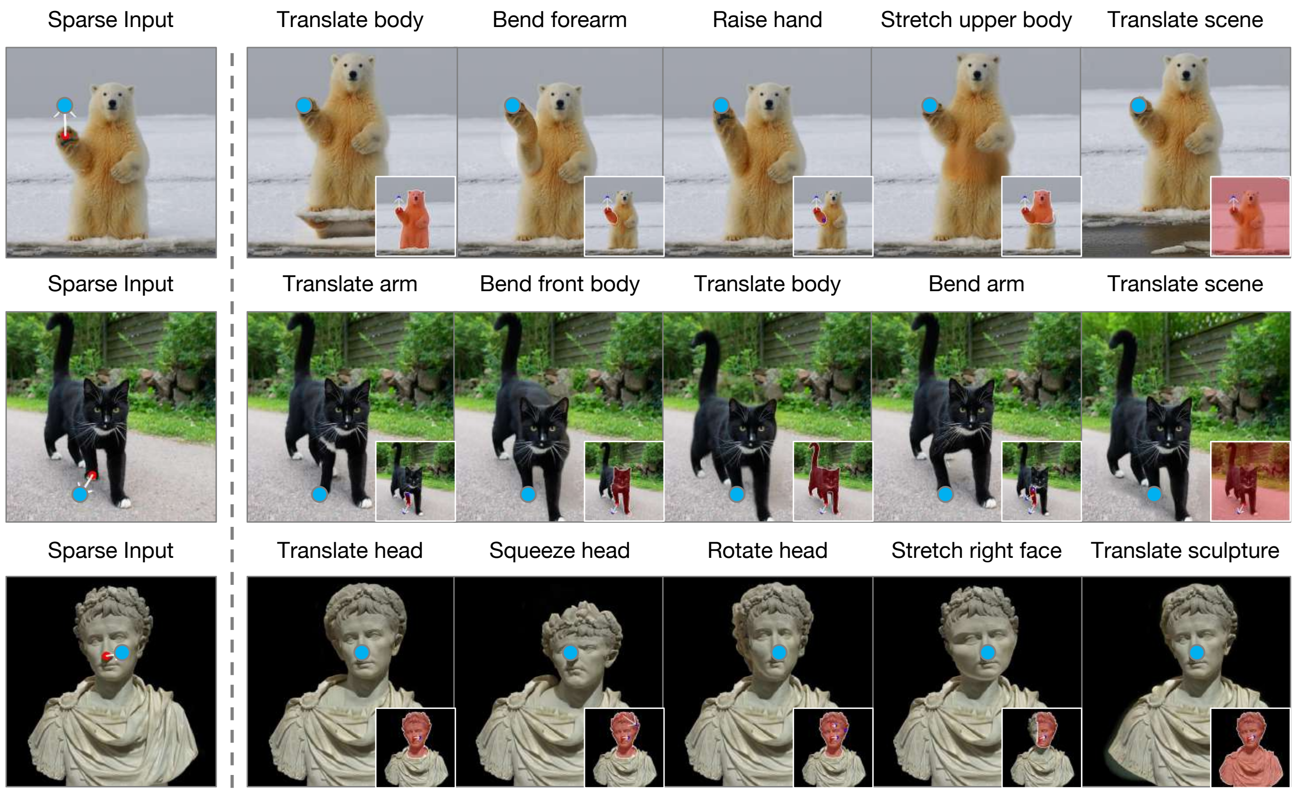

Figure S3. Precise control over ambiguous drag operations. **Left**: Ambiguous sparse input from previous methods can represent at least five different user intentions. **Right**: Through our explicit deformation-based control interface (bottom-right insets), we precisely implement each distinct user intention, effectively eliminating ambiguity while maintaining intuitive interaction.

## S5. More Qualitative Results

We present extensive qualitative results in Figs. S4 to S6. Our method allows users to specify handle points (red) and target points (blue) with arrows defining deformation regions (highlighted in red). By applying elastic material principles directly in pixel space, we achieve superior performance across diverse editing scenarios. The results demonstrate our method's effectiveness in facial edits, large-scale deformations, and precise local manipulations while maintaining geometric stability. This advantage is particularly evident when handling significant boundary changes and occlusions, where our inpainting models realistically complete both texture and background.

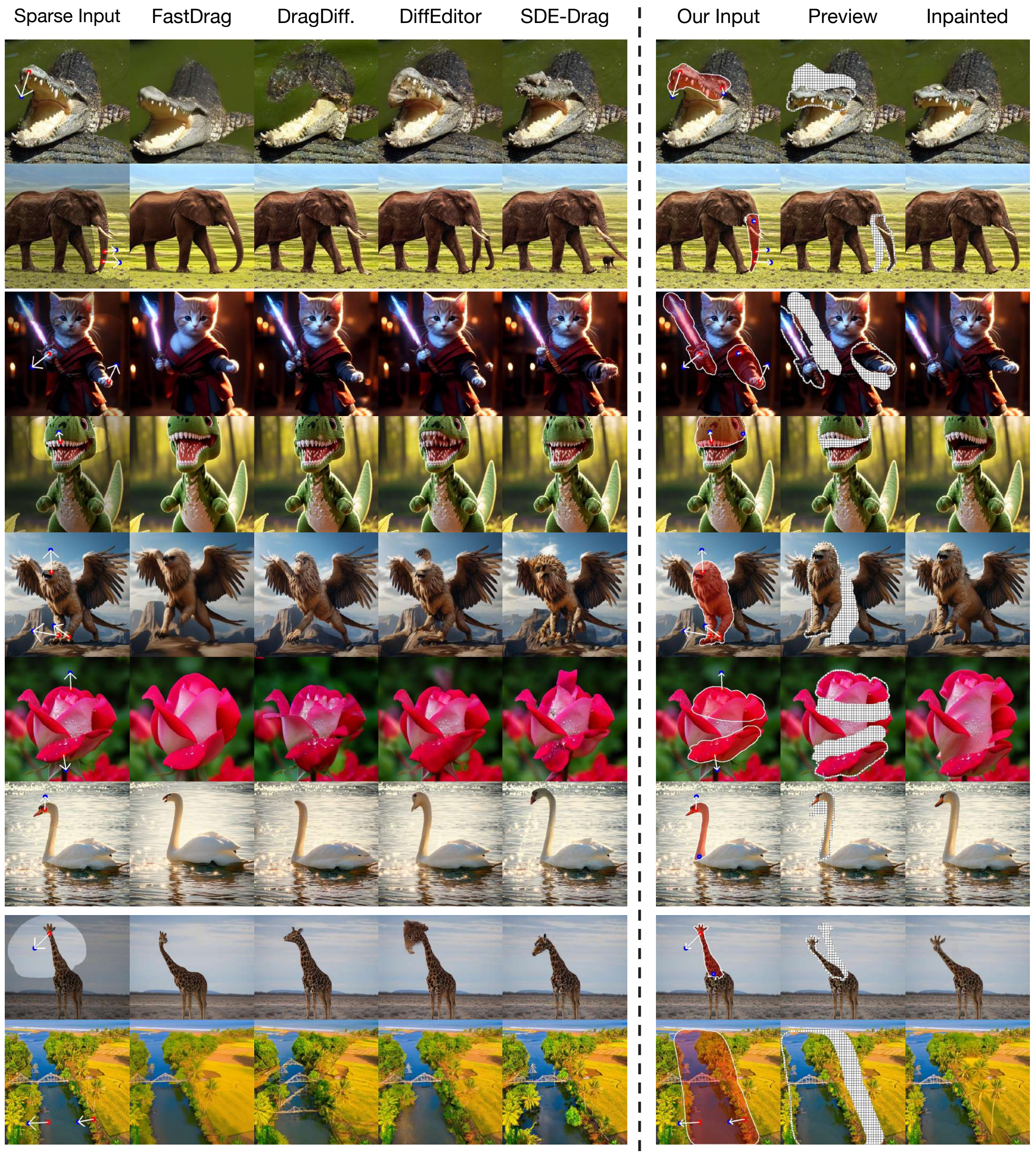

Figure S4. Qualitative comparison of Inpaint4Drag with state-of-the-art methods: wildlife, artworks, flowers, birds, and landscapes.

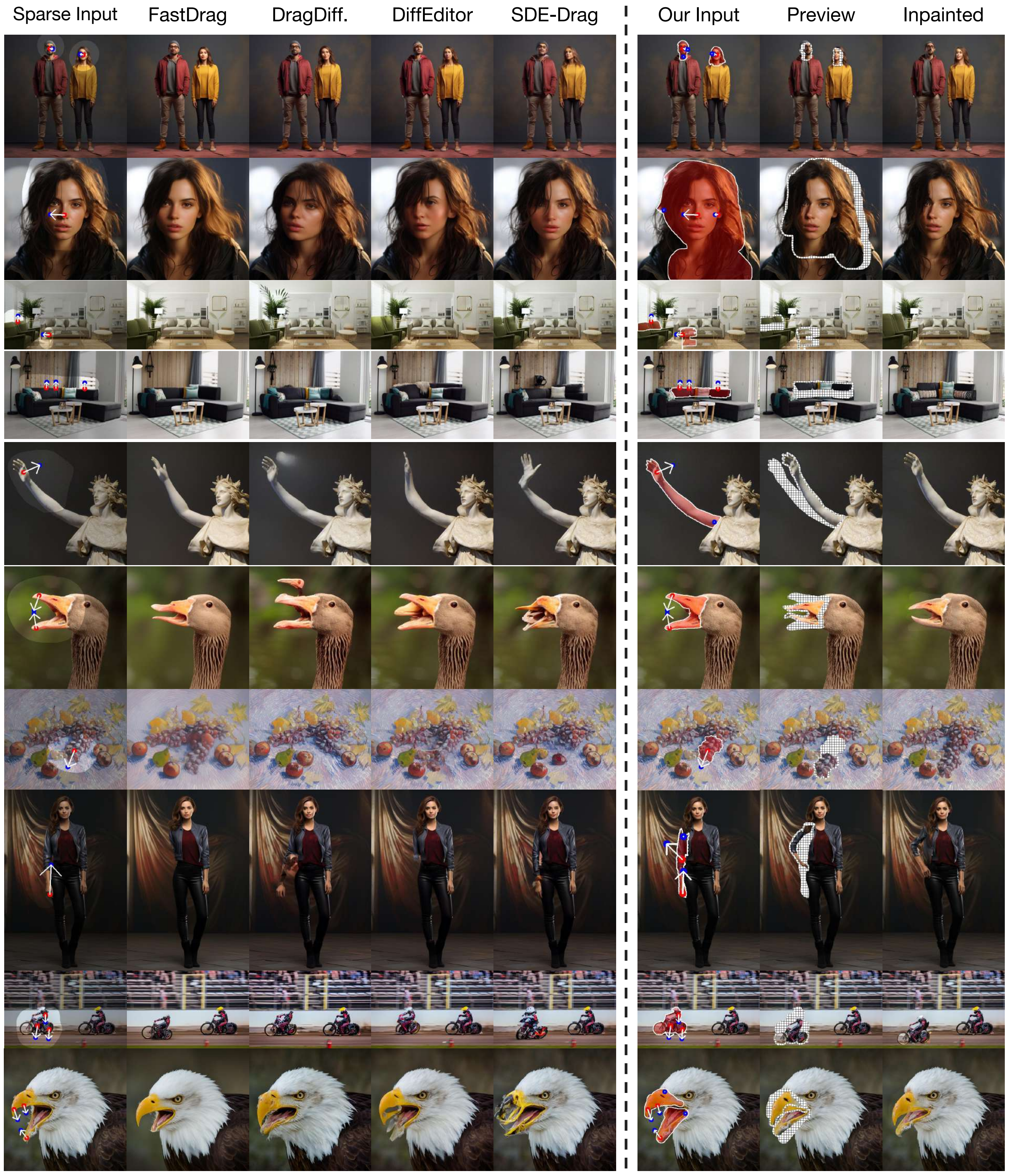

Figure S5. Qualitative comparison of Inpaint4Drag with state-of-the-art methods: portraits, interiors, statues, wildlife, still life, and sports.

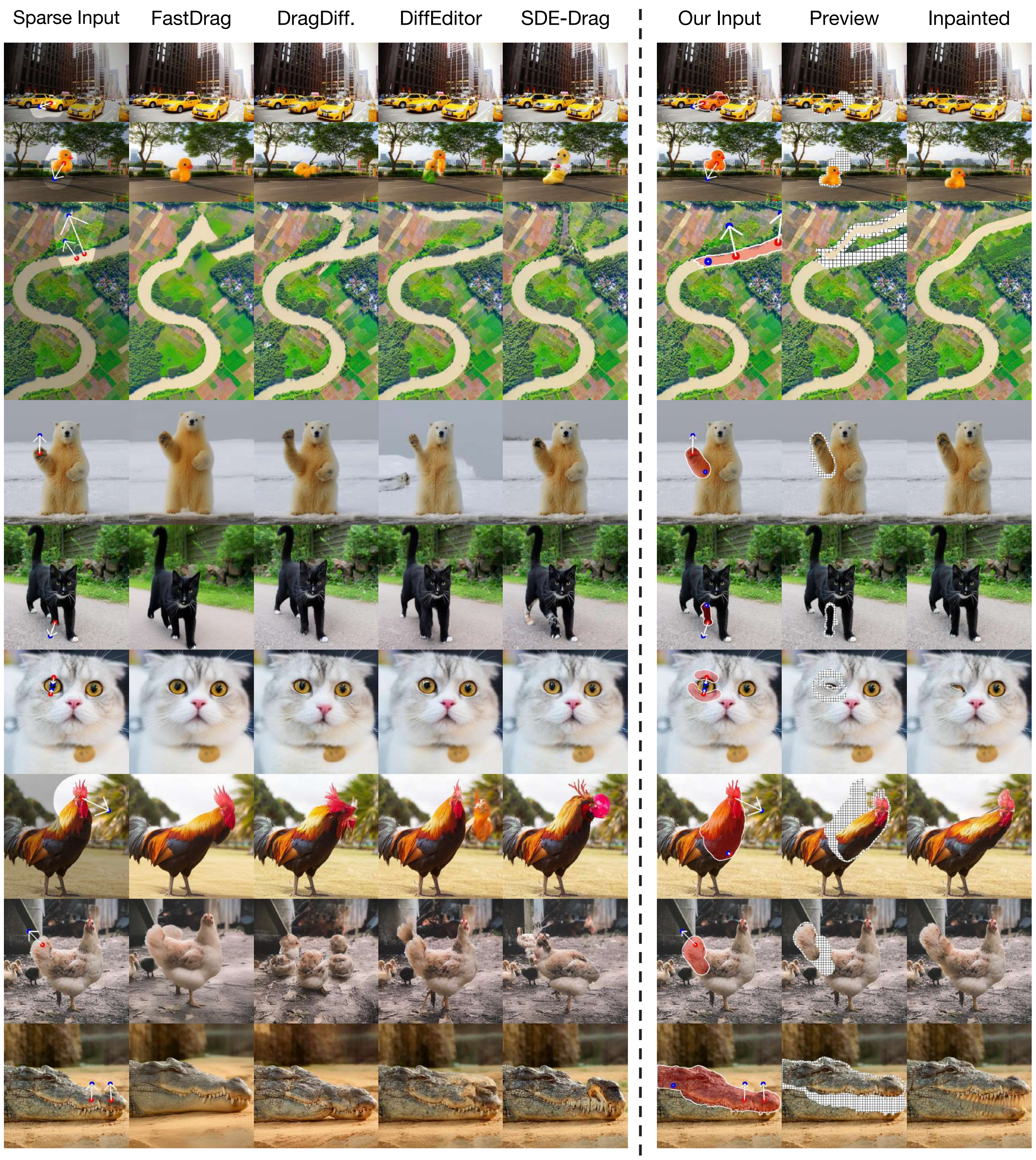

Figure S6. Qualitative comparison of Inpaint4Drag with state-of-the-art methods: urban scenes, landscapes, animals, pets, and reptiles.

## S6. Pseudo Code for Inpaint4Drag

To complement the detailed description of our method of the main paper, we provide a concise algorithmic representation of the data flow in Algorithm 1.

**Algorithm 1:** Inpaint4Drag: Drag-based Image Editing via Bidirectional Warping and Inpainting

```
Input: Image $I$, user-drawn mask $M$, handle points $\{h_i\}$ and target positions $\{t_i\}$
Output: Edited image $I_{\text{edit}}$
Region Specification and Boundary Refinement:;
$P_s \leftarrow$ SampleGridPoints($M$) ;                                // Sample grid points from user mask
$M_{pred} \leftarrow f_{SAM}(I, P_s)$ ;                                          // SAM prediction
$M_{dilated} \leftarrow$ Dilate($M$, $K_1$); $M_{eroded} \leftarrow$ Erode($M$, $K_1$) ;      // Create boundary constraints
$M \leftarrow (M_{pred} \cap M_{dilated}) \cup M_{eroded}$ ;                        // Boundary-guided refinement
Bidirectional Warping:;
$\mathcal{C} \leftarrow$ ExtractContours($M$) ;                                // Get deformable regions
foreach contour $C \in \mathcal{C}$ do
    Associate control points: $(h_i, t_i) \leftarrow \{(h_i, t_i) \mid h_i$ inside $C\}$;
    Forward Warping: ;                                          // Define target region boundary
    foreach point $p$ in $C$ do
        $w_i \leftarrow \frac{1/(\|p-h_i\|+\epsilon)}{\sum_j 1/(\|p-h_j\|+\epsilon)}$;
        $p_t \leftarrow p + \sum_i w_i(t_i - h_i)$ ;                      // Weighted interpolation
        Store mapping pair $(p, p_t)$;
    end
    $C' \leftarrow$ transformed contour from forward warping;
    Backward Mapping: ;                                        // Ensure complete pixel coverage
    foreach pixel $p_t$ within boundary of $C'$ do
        Find $N_n$ nearest matched pixels $\{p_i^{tgt}\}$ with source positions $\{p_i^{src}\}$;
        $p_s \leftarrow p_t + \sum_{i=1}^{N_n} w_i(p_i^{src} - p_i^{tgt})$ ;         // Local neighborhood interpolation
        if $p_s \in [0, W) \times [0, H)$ and $p_t \in [0, W) \times [0, H)$ then
            Store valid mapping $(p_s, p_t)$;
        end
    end
end
Compute Warped Image and Inpainting Mask:;
foreach valid mapping pair $(p_s, p_t)$ do
    $I_{\text{warped}}(p_t) \leftarrow I(p_s)$ ;                                 // Transfer pixel values
end
$M_{\text{warped}} \leftarrow$ mask of pixels filled in warped image;
$M_{\text{temp}} \leftarrow M \setminus M_{\text{warped}}$;
$\partial M_{\text{warped}} \leftarrow$ boundary of $M_{\text{warped}}$ ;                   // Identify unmapped regions
$M_{\text{inpaint}} \leftarrow$ Dilate($M_{\text{temp}} \cup \partial M_{\text{warped}}, K_2$) ;                // Create buffer zone
Image Inpainting:;
$I_{\text{edit}} \leftarrow$ Inpaint($I_{\text{warped}}, M_{\text{inpaint}}$) ;                       // Apply inpainting model
return $I_{edit}$
```

## S7. Limitations

While our method achieves significant improvements in efficiency and precision, it relies on accurate user-specified masks and control points for optimal performance. Imprecise user inputs, such as masks that inadvertently include background elements or poorly positioned control points, can lead to undesired deformation artifacts. Future work could explore understanding-enabled models that automatically filter irrelevant background elements or provide intelligent suggestions for mask and control point placement, reducing the burden on users to provide perfectly accurate inputs while maintaining the intuitive nature of drag-based interaction.

## References

[1] 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. 4

[2] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 2

[3] Yujun Shi, Jun Hao Liew, Hanshu Yan, Vincent YF Tan, and Jiashi Feng. Instadrag: Lightning fast and accurate drag-based image editing emerging from videos. *arXiv preprint arXiv:2405.13722*, 2024. 4

[4] Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. Resolution-robust large mask inpainting with fourier convolutions. In *WACV*, 2022. 2

[5] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Free-form image inpainting with gated convolution. In *ICCV*, 2019. 2

[6] Xuanjia Zhao, Jian Guan, Congyi Fan, Dongli Xu, Youtian Lin, Haiwei Pan, and Pengming Feng. Fastdrag: Manipulate anything in one step. In *NeurIPS*, 2024. 4