EditCtrl: Disentangled Local and Global Control for Real-Time Generative Video Editing

Supplementary Material

6. Additional Visualizations

We show additional results editing in distinct regions with multiple prompts as well as content propagation in Figs. 10-11. Furthermore, we provide additional visualizations of EditCtrl's video editing and inpainting results on VPBench and DAVIS in Figs. 12-15.

7. Additional Details

Content Propagation. When using a distilled autoregressive video diffusion model, video generation can be extended to an indefinite range, albeit with some loss in quality by denoising new incoming noisy latents. When used as the base video model for EditCtrl, this means we inherit the autoregressive model's properties and can edit a video of any length by using sliding window attention over a local window of frames. This is in addition to accelerated inference with respect to the target edit area. Moreover, we can extend this to real-time video editing, where we edit a video indefinitely, but without having access to the background context. This is achieved by simply copying the background context from the last known frames. If the video is static then the background context tokens used for local context can be copied over for generating next frames. In cases where the background context is dynamic, we use optical flow to propagate the last frame's pixels to roughly approximate the future local context. In both static and dynamic cases, the last downsampled background frame $\mathbf{V}^{\downarrow}_{b_N}$ is padded to $\mathbf{V}^{\downarrow}_{b}$ to provide a rough estimate for future global context. Once the future frame arrives, the generated content is pasted into the corresponding pixels along with optional edge blending. This process is repeated, where optical flow propagation is computed from the last known frame to the next future frame. These steps help generate more visually appealing content that matches the input video's motion ahead of time, whereas if we wait for future frames the local context will be much more accurate but the latency requirement would be too small, making the generated output appear laggy. An example of EditCtrl in an augmented reality setting is shown in Fig. 7, where initial frames are known and then EditCtrl propagates content to the next frames.

G_ψ **Resolution & Transferability.** To test the fixed 256×256 global embedder resolution and transferability, we test EditCtrl on a challenging 21:9 ultrawide timelapse video by itself and with a publicly available aesthetic style LoRA [9] in Fig. 8. This example demonstrates that the downsampled global context captures sufficient spatial and temporal information even for extreme aspect ratios and complex scenes, and moreover, that our adapter design composes seamlessly with style LoRAs, confirming the non-destructive design generalizes beyond the base model. This is shown by the enhanced style EditCtrl generates when using an aesthetic LoRA.

Failure Modes. We present two cases of inputs where EditCtrl fails in Fig. 9. In the first case on the left, at high motion EditCtrl introduces artifacts and incorrect interaction with the surrounding scene. The artifacts are also present for the full-attention baseline VACE, indicating that this is a structural problem that we attribute to the VAE. In the second case on the right, as the mask becomes exceedingly large finegrained editing becomes more difficult and EditCtrl changes the content by more than desirable.

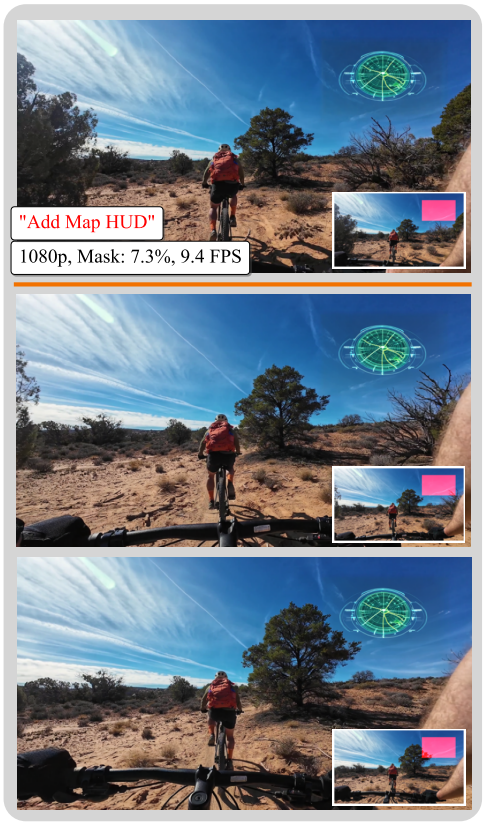


Figure 7. **Content Propagation for Augmented Reality.** EditCtrl is particularly suitable for deployment in augmented reality applications given its low latency and ability to propagate content to match the user's movement.

Figure 8. **Editing complex timelapse 21:9 ultrawide footage.** EditCtrl can also be used with additional conditioning LoRAs [9].

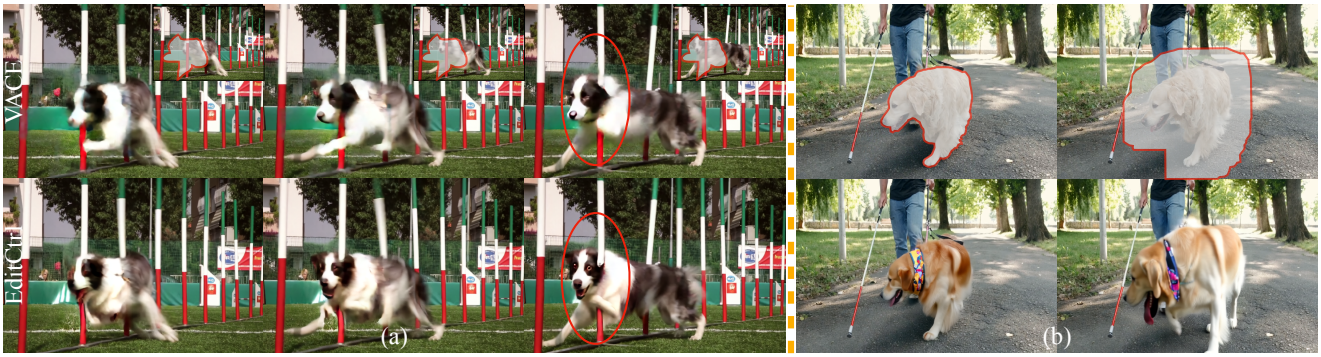
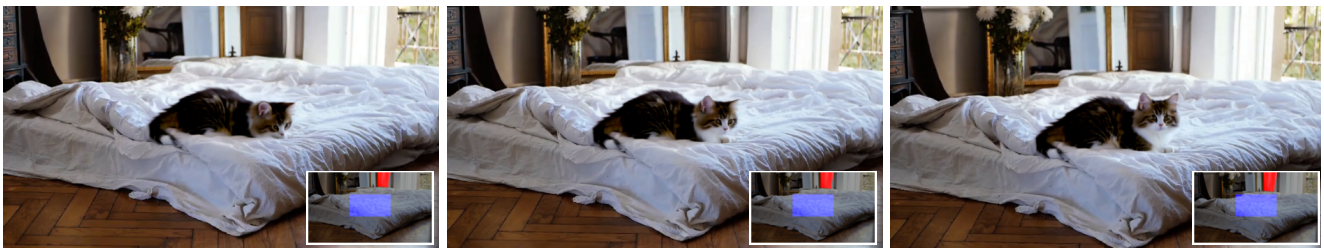


Figure 9. **Failure Modes for EditCtrl on High Motion (a) & Inaccurate Masks (b).**

(a) “Add UFO.”, “Add fire-breathing monster.”, “Add portal.”

(b) “Add seagulls.”, “Add cruise ship.”



(c) “Remove the curtain.”, “Add a cat.”

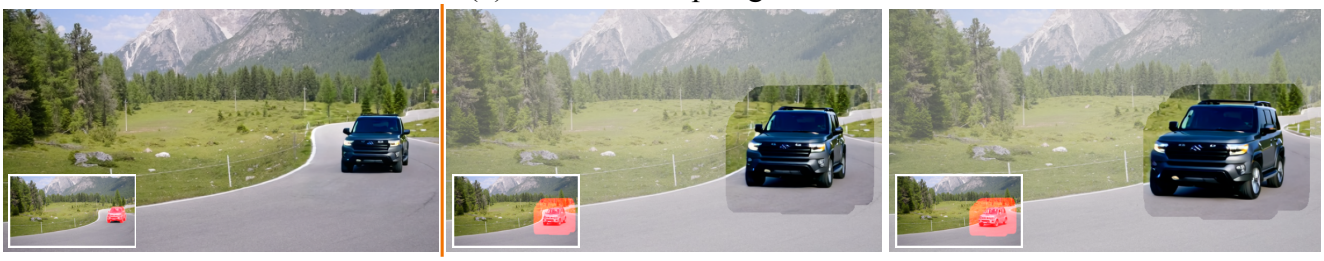
Figure 10. **Additional Multi-Prompt Generation Visualizations.**

(a) “Replace the dog with a plant.”

(b) “Remove the duck.”

(c) “Change the train’s colors.”

(d) “Remove the paraglider.”



(e) “Change the car to a black SUV.”

Figure 11. **Additional Content Propagation Visualizations.**

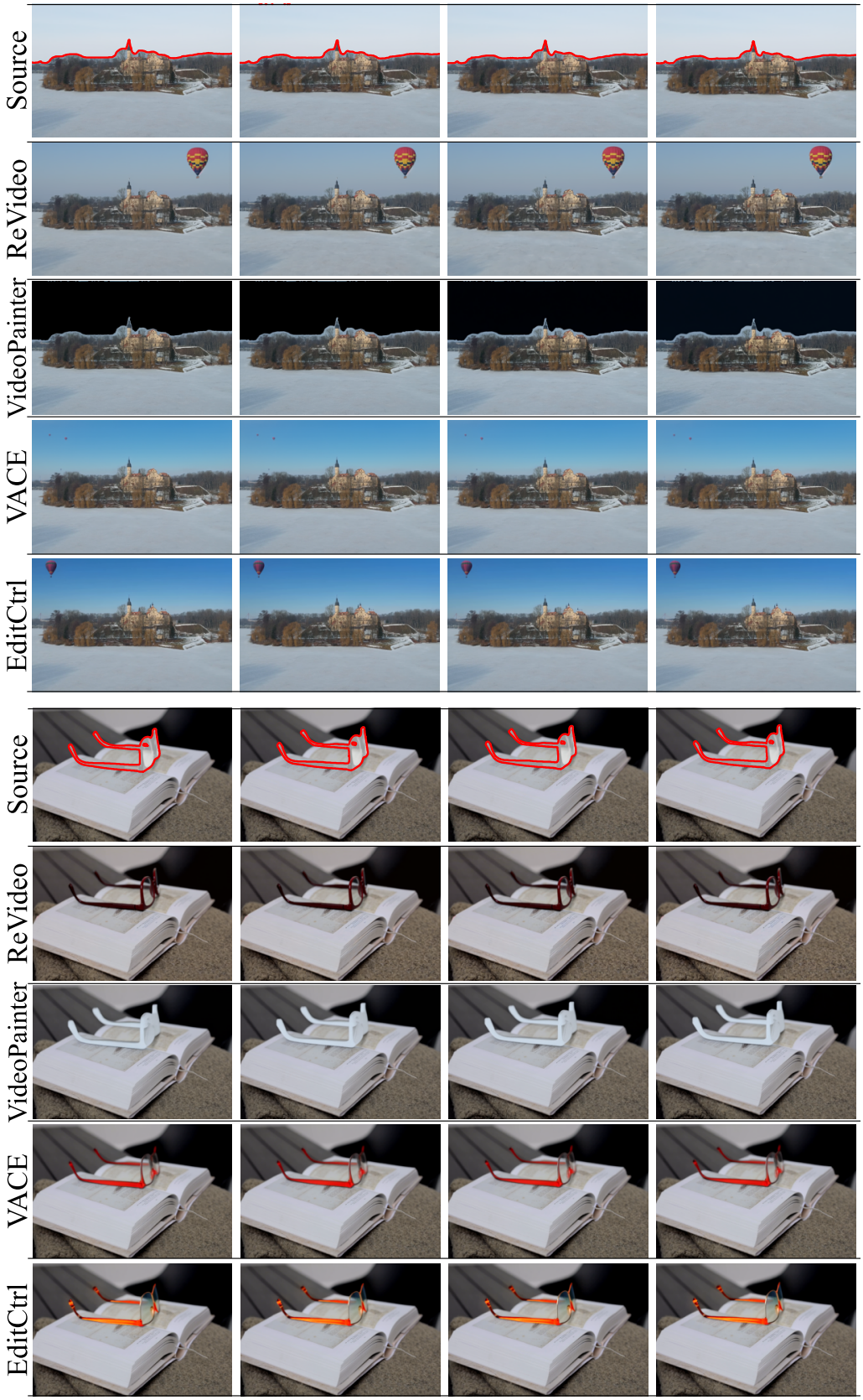


Figure 12. **Additional Video Editing Comparisons.** (Top) "Add a hot air balloon" (Bottom) "Turn the sunglasses shiny red."

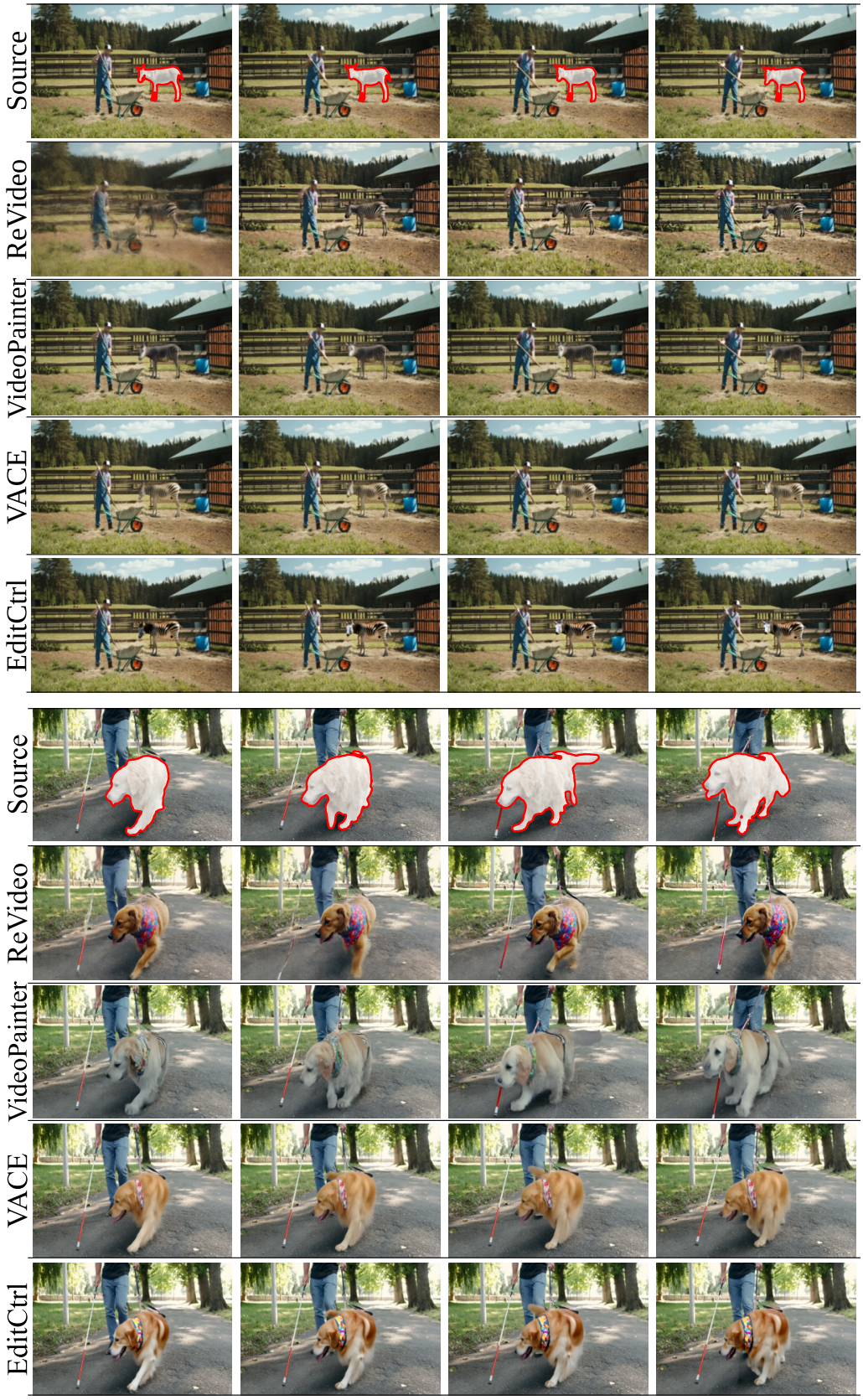


Figure 13. **Additional Video Editing Comparisons.** (Top) "Turn the donkey into a zebra." (Bottom) "Add a bandana to the dog."

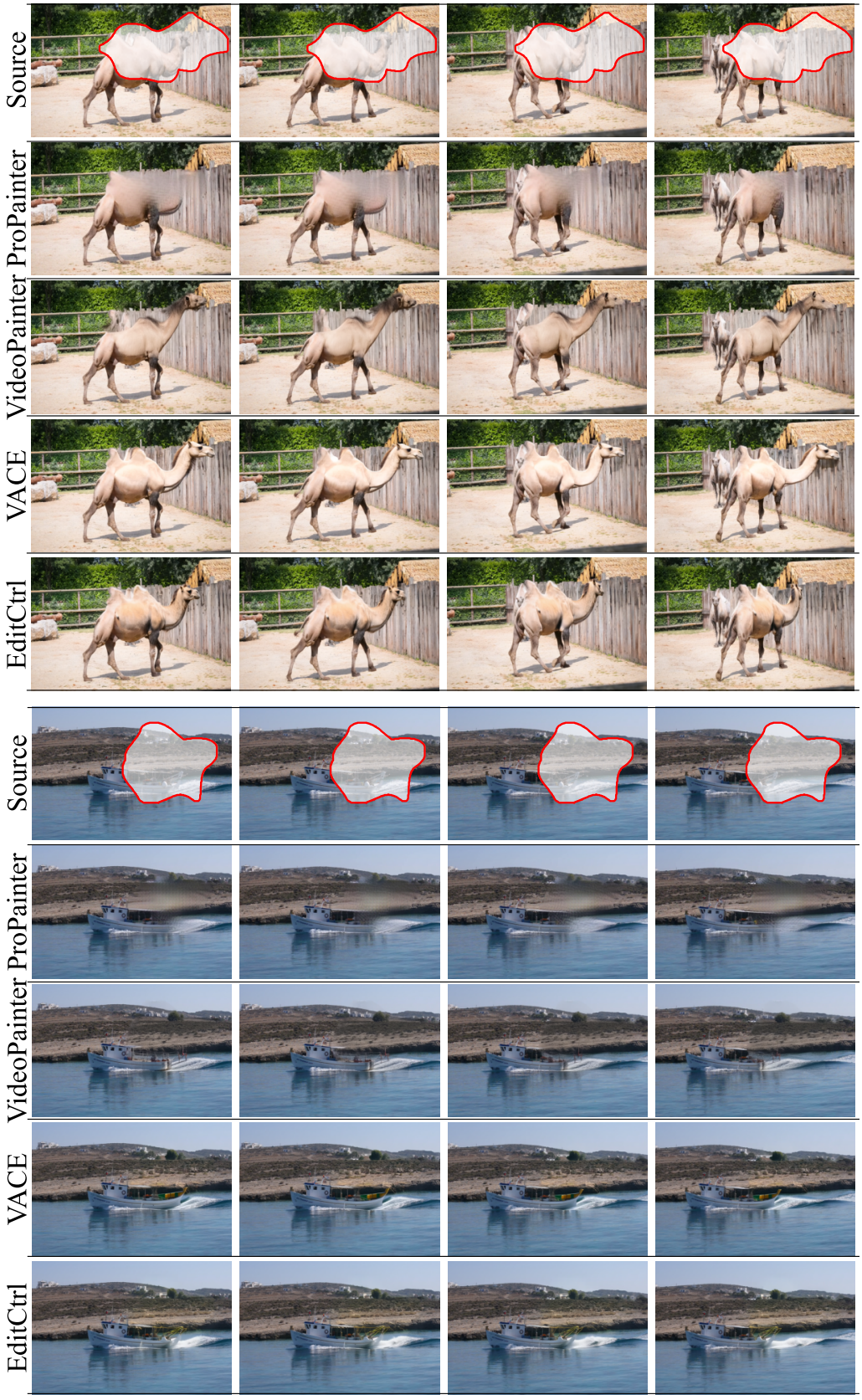


Figure 14. **Additional Video Inpainting Comparisons on DAVIS.**

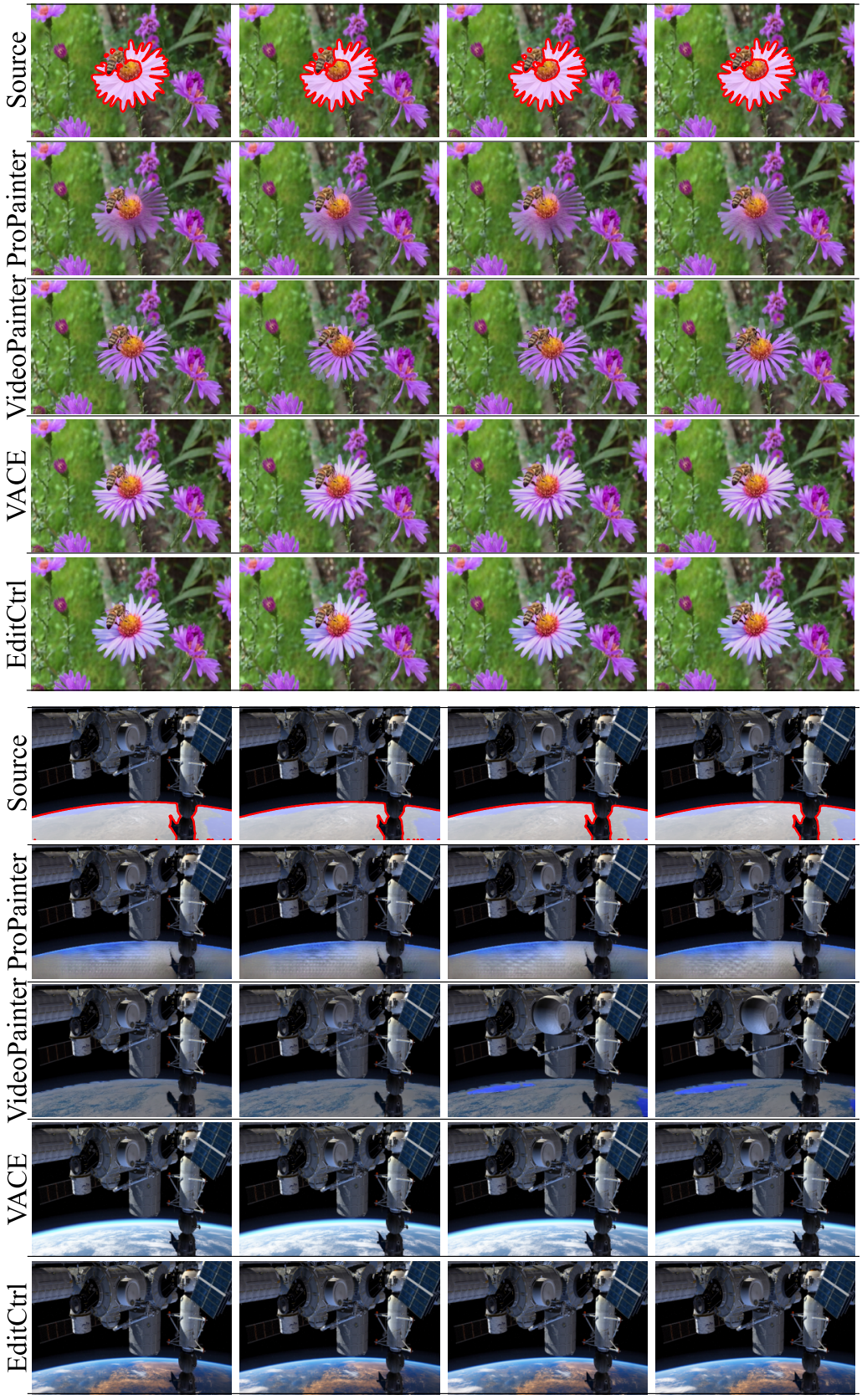


Figure 15. **Additional Video Inpainting Comparisons on VPBench.**