Reangle-A-Video: 4D Video Generation as Video-to-Video Translation

Supplementary Material

This supplementary material is structured as follows: In Sec. A, we provide additional experimental details. Sec. B presents additional experiment results. Following this, we discuss the limitations and failure cases of Reangle-A-Video in Sec. C.

A. Additional Experimental Details

A.1. Camera Visualizations

We demonstrate six degrees of freedom in both (a) Static view transport and (b) Dynamic camera control. Fig. 10 visualizes the transported viewpoints and camera movements used in our work: *orbit left, orbit right, orbit up, orbit down, dolly zoom in, and dolly zoom out.*

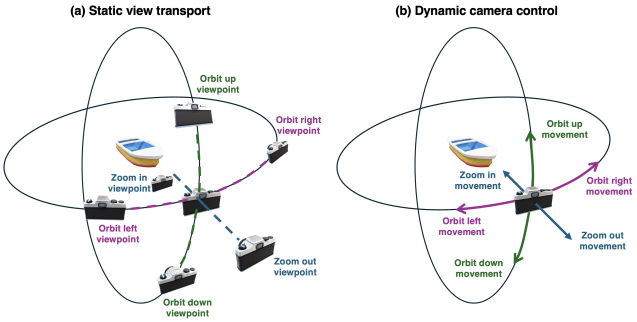

Figure 10. **Visualizations of the used camera types.**

A.2. Warped Video Dataset Composition

For (a) static view transport setup, we set M as 12, generating warped videos from 12 different viewpoints: two random-angle orbit left, two random-angle orbit right, two random-angle orbit up, two random-angle orbit down, two random-range dolly zoom in, and two random-range dolly zoom out. Including the original input video, the training set consists of 13 videos.

For (b) dynamic camera control training, we set M as 6, rendering warped videos with six different camera movements: one random-angle orbit left, one random-angle orbit right, one random-angle orbit up, one random-angle orbit down, one random-range dolly zoom in, and one random-range dolly zoom out. Including the original input video, this also results in a training set of 7 videos.

A.3. Finetuning Cost

We originally trained and inferred at a high resolution ($49\times480\times720$) on a 40GB GPU using *gradient checkpointing* to fit within memory—this significantly slows down backpropagation. However, on an 80GB GPU without gradient checkpointing, the process is 3.3× faster, completing in about 18 minutes. Alternatively, on a 40GB GPU with checkpointing but with reduced temporal resolution (25 frames), it's 2.3× faster (26 minutes).

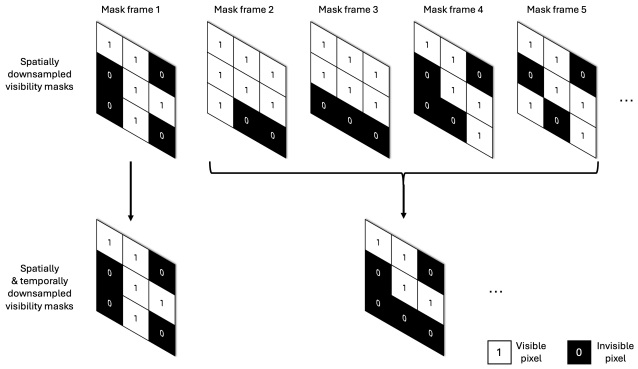

Figure 11. **Temporal downsampling of visibility masks.** Except for the first mask frame, pixel-wise (element-wise) logical AND operation is done for every four masks.

A.4. 3D Downsampling Visiblity Masks

Recent video diffusion models rely on 3D VAEs that perform both spatial and temporal compression to alleviate the computational burden of modeling long video sequences. However, such compression complicates the direct mapping of RGB pixel-space visibility masks to their corresponding latent regions. In our work, we adopt the 3D VAE architecture of CogVideoX, which features a spatio-temporal compression rate given by $H \times W \times N = 8h \times 8w \times (1+4n)$, where (H, W, N) denote the pixel-space resolutions, and (h, w, n) represent the corresponding resolutions in the latent space. Given a visibility mask in pixel-space, $M \in \mathbb{R}^{H\times W\times N}$, we first downsample the spatial dimensions by a factor of 8 using *nearest-neighbor* interpolation. For temporal downsampling, we retain the first frame intact and then compress every subsequent group of four frames via an element-wise logical AND operation (see Fig. 11 for the illustration). This procedure ensures that a latent region is marked as visible only if it is visible across all frames within each group.

A.5. Pre-trained Model Checkpoints

Reangle-A-Video builds upon publicly available pre-trained image and video generative models. Reangle-A-Video builds on publicly available pre-trained image and video generative models. Here, we specify the versions used:

- Text-to-Image generation model: Flux.1-dev [4]
- Image-to-Image inpainting model: Flux-ControlNet-Inpainting-Beta [5]
- Image-to-Video generation model: CogVideoX-I2V-5b [6]

Algorithm 1 Multi-View Consistent Image Inpainting (diffusion model)

Require: Inpainted images $\{\tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_I\}$, the image to be inpainted $\hat{\boldsymbol{x}}_{I+1}$, and the image inpainting diffusion model $(\phi, \mathcal{E}, \mathcal{D})$.
$\boldsymbol{c} \leftarrow \mathcal{E}(\hat{\boldsymbol{x}}_{I+1})$
$\boldsymbol{z}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T$ **to** 1 **do**
  for $s = 1$ **to** S **do**
    $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    $\boldsymbol{z}^s_{t-1} \leftarrow \text{DDPM\_step}(\boldsymbol{z}_t, \phi, \boldsymbol{c}, t, \boldsymbol{\epsilon})$
    $\boldsymbol{z}^s_{0|t-1} \leftarrow \frac{1}{\sqrt{\bar{\alpha}_{t-1}}}(\boldsymbol{z}^s_{t-1} - \sqrt{1-\bar{\alpha}_{t-1}}\hat{\boldsymbol{\epsilon}}_\phi(\boldsymbol{z}^s_{t-1}, \boldsymbol{c}, t))$
  end for
  for $s = 1$ **to** S **do**
    for $j = 1$ **to** I **do**
      $r_{s,j} \leftarrow \text{eval\_3d\_consistency}(\mathcal{D}(\boldsymbol{z}^s_{0|t-1}), \tilde{\boldsymbol{x}}_j)$
    end for
    $r_s \leftarrow \frac{1}{I}(r_{s,1} + r_{s,2} + \cdots + r_{s,I})$
  end for
  $s^* \leftarrow \text{argmax}_s\ r_s$
  $\boldsymbol{z}_{t-1} \leftarrow \boldsymbol{z}^{s^*}_{t-1}$
end for
$\tilde{\boldsymbol{x}}_{I+1} \leftarrow \mathcal{D}(\boldsymbol{z}_0)$
return $\tilde{\boldsymbol{x}}_{I+1}$

A.6. Multi-view Consistent Images Completion

We present the detailed algorithm for our proposed multi-view consistent image completion. As described in the main text, our approach sequentially performs multi-view inpainting using Algorithm 1 or 2, based on previously completed images. We introduce two versions: a diffusion model-based method (Algorithm 1) and a flow model-based method (Algorithm 2).

To generate multiple sample versions at each denoising step for the stochastic control guidance [42, 66, 88], the diffusion model-based approach employs DDPM [33] sampling, while the flow model-based approach solves an SDE [54] that shares the same marginal distribution as its corresponding ODE. Specifically, for the Wiener process W_t:

$$dZ_t = \boldsymbol{v}_t(Z_t)dt - Z_{0|t}(1-t)dt + \sqrt{2(1-t)^2}dW_t, \quad \text{where} \quad Z_{0|t} = Z_t - t\boldsymbol{v}_t(Z_t). \tag{7}$$

The SDE is solved using the Euler–Maruyama method.

Algorithm 2 Multi-View Consistent Image Inpainting (flow-based model)

Require: Inpainted images $\{\tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_I\}$, the image to be inpainted $\hat{\boldsymbol{x}}_{I+1}$, and the image inpainting flow-based model $(\phi, \mathcal{E}, \mathcal{D})$.
$\boldsymbol{c} \leftarrow \mathcal{E}(\hat{\boldsymbol{x}}_{I+1})$
$\boldsymbol{z}_{t_1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $i = 0$ **to** $T-1$ **do**
  $\Delta t \leftarrow t_{i+1} - t_i$
  $\boldsymbol{z}_{0|t_i} \leftarrow \boldsymbol{z}_{t_i} - t_i\hat{\boldsymbol{v}}_\phi(\boldsymbol{z}_{t_i}, \boldsymbol{c}, t_i)$
  for $s = 1$ **to** S **do**
    $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    $\boldsymbol{z}^s_{t_{i+1}} \leftarrow \boldsymbol{z}_{t_i} + \hat{\boldsymbol{v}}_\phi(\boldsymbol{z}_{t_i}, \boldsymbol{c}, t_i)\Delta t - \boldsymbol{z}_{0|t_i}(1-t_i)\Delta t + \sqrt{2(1-t_i)^2\Delta t}\boldsymbol{\epsilon}$
    $\boldsymbol{z}^s_{1|t_{i+1}} \leftarrow \boldsymbol{z}^s_{t_{i+1}} + (1-t)\hat{\boldsymbol{v}}_\phi(\boldsymbol{z}^s_{t_{i+1}}, \boldsymbol{c}, t_{i+1})$
  end for
  for $s = 1$ **to** S **do**
    for $j = 1$ **to** I **do**
      $r_{s,j} \leftarrow \text{eval\_3d\_consistency}(\mathcal{D}(\boldsymbol{z}^s_{0|t_{i+1}}), \tilde{\boldsymbol{x}}_j)$
    end for
    $r_s \leftarrow \frac{1}{I}(r_{s,1} + r_{s,2} + \cdots + r_{s,I})$
  end for
  $s^* \leftarrow \text{argmax}_s\ r_s$
  $\boldsymbol{z}_{t_{i+1}} \leftarrow \boldsymbol{z}^{s^*}_{t_{i+1}}$
end for
$\tilde{\boldsymbol{x}}_{I+1} \leftarrow \mathcal{D}(\boldsymbol{z}_{t_T})$
return $\tilde{\boldsymbol{x}}_{I+1}$

After obtaining denoised estimates ($Z_{0|t}$ for the diffusion model, $Z_{1|t}$ for the flow model), they are decoded and compared against previously completed images using the method from [1] to ensure multi-view consistency. The best sample is selected for the next step. Specifically, for the initial view completion, the first two views are inpainted simultaneously by extending the given algorithm, which evaluates consistency across all possible sample pairs and selects the best pair for the next step.

B. Additional Experimental Results

B.1. Using Warped Videos for Fine-tuning

Due to the absence of an automatic metric for *multi-view motion fidelity*, we conduct a human evaluation to assess the necessity of using warped videos during fine-tuning (Sec. 3.3). Participants were shown two randomly selected videos and asked, "Does the generated video accurately preserve

[4] https://huggingface.co/black-forest-labs/FLUX.1-dev
[5] https://huggingface.co/alimama-creative/FLUX.1-dev-Controlnet-Inpainting-Beta
[6] https://huggingface.co/THUDM/CogVideoX-5b-I2V

the input video's motion?" before choosing the superior video. The results are shown in Tab. 3.

Table 3. **Quantitative ablation on warped videos during fine-tuning.** Multi-view motion fidelity is evaluated via human studies.

	w/ warped videos	w/o warped videos
Multi-view motion fidelity	**80.44%**	19.56%

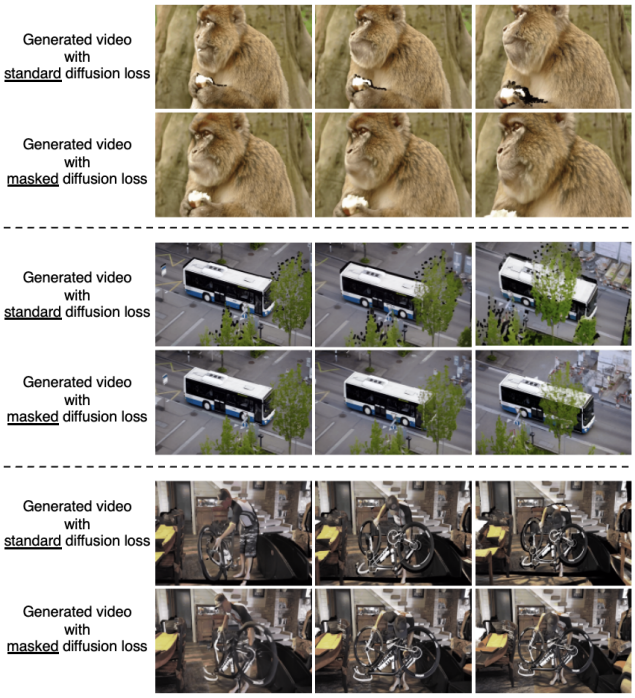

Figure 12. **Impact of masked diffusion loss on video quality.** Masking the diffusion loss effectively prevents artifacts.

B.2. Masking Diffusion Loss

Building on the flexible compositionality of diffusion objectives, masked diffusion loss has been applied to diffusion-based customizations [2, 92], video interpolation [17], and efficient image diffusion model training [25, 96]. In our work, we employ masked diffusion loss on a pre-trained video diffusion transformer architecture to distill the 4D motion prior of an arbitrary scene. As shown in Fig. 12, this objective effectively prevents artifacts and eliminates warped (black) regions in the generated videos.

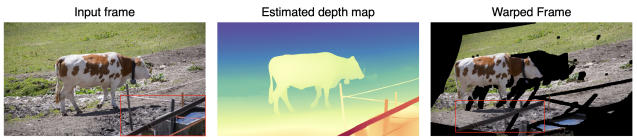

Figure 13. **Geometric misalignment in the warped frame.** In this example, the target camera view is shifted 10 degrees to the (horizontal orbit) right of the input frame.

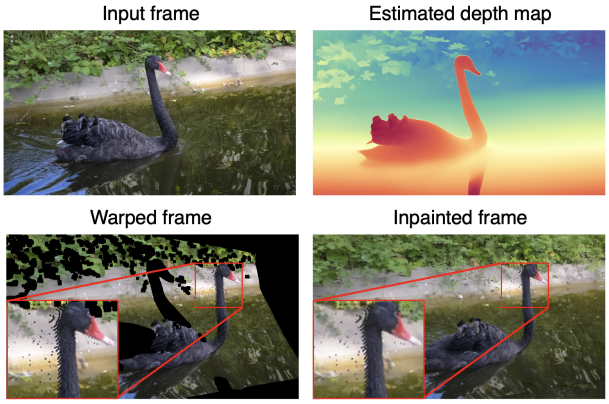

Figure 14. **Pixel-scale artifacts in the warped frame.** In this example, the target camera view is shifted 10 degrees to the (horizontal orbit) left of the input frame.

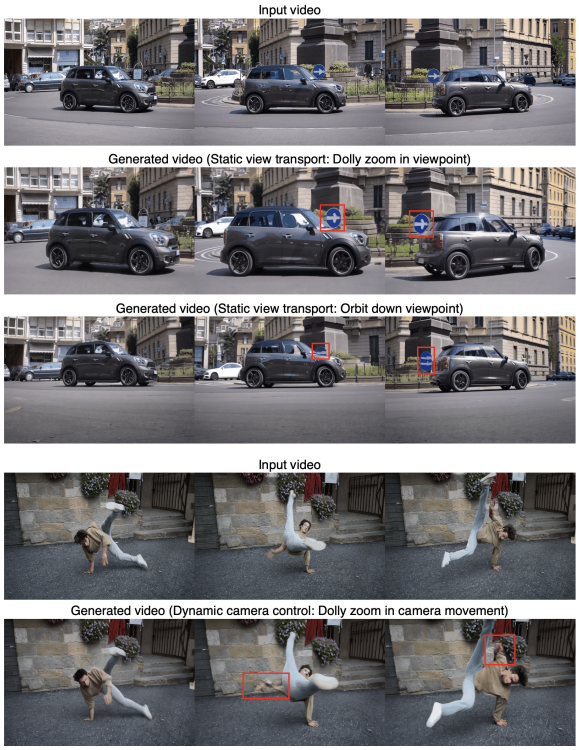

Figure 15. **Failure cases of Reangle-A-Video.**

C. Limitations and Failure Cases

Our method relies on point-based warping using estimated depth maps, making it inherently vulnerable to errors from inaccurate depth estimation and incorrect camera intrinsics. These inaccuracies can distort the warping process, leading to geometric misalignment and depth inconsistencies. For instance, Fig. 13 illustrates an example of geometric misalignment in the warped frame. Moreover, as shown in Fig. 14, warping errors can introduce pixel-level artifacts that may not be fully masked by visibility estimations, allowing these small distortions to persist in the inpainted

results. Addressing these limitations would require more accurate depth estimation and better handling of occlusion constraints. In Fig. 15, we present failure cases. For instance, our method produces inaccurate reconstructions in videos with small regions (e.g., a blue sign in the background) or fast, complex motion (e.g., a man breakdancing). We attribute these issues partly to the fact that video fine-tuning and inference are performed in a spatially and temporally compressed latent space.