SV4D 2.0: Enhancing Spatio-Temporal Consistency in Multi-View Video Diffusion for High-Quality 4D Generation

Supplementary Material

In this supplemental document, we include the implementation details in Sec. 1, ablative analyses in Sec. 2, and additional results in Sec. 3. We also attach a teaser video (`SV4D_2.0_video.mp4`) to summarize the SV4D 2.0 framework and show more visual results.

1. Implementation Details

1.1. Data curation details

Dataset overview. We illustrate the overall data pipeline in Fig. 1. To train SV4D 2.0, we render CC-licensed animatable 3D objects from the Objaverse [4] and ObjaverseXL datasets [3] datasets. Objaverse contains 44k dynamic 3D objects and ObjaverseXL includes roughly 323k in the GitHub subset. Similar to SV4D [11], we filter out nearly half of the objects based on license, inconsistent scaling, object motion, etc. We scale each object such that the largest world-space XYZ extent of its bounding box is 1, then use Blender's CYCLES renderer to render multiple views and video frames at 576×576 resolution. For lightning, we randomly select from a set of curated HDRI environment maps. Following SV3D [10], we render both static (uniform azimuth and fixed elevation) and dynamic (irregular azimuth and elevation) orbits for training. We then encode all rendered images into the latent space using SD2.1 [9]'s VAE and CLIP [8].

Disentangling global and local motion. In our improved 4D dataset, ObjaverseDy++, we disentangle the global transformation from local motion of object parts to preserve temporal correspondence between frames during motion and prevent truncation or off-center cases. As shown in Fig. 2, we identify the most static region by computing the mean 3D displacement of each vertex over time and computing the average offset within this region as global translation. Then, we subtract the global translation from all vertex coordinates to fix the static region.

Mitigating strong shading effect. Moreover, to reduce baked-in lighting effect that sometimes causes dark back views, we filter out environment maps that have strong shading effect and only sample from near-ambient maps during object rendering. We show in Fig. 5 (snowboarding and horse riding videos) that this effectively mitigates the issue of dark back views.

1.2. SV4D 2.0 training details

Similar to SVD [1], we train our model following the widely used EMD [6] scheme with $L2$ loss. To optimize the training speed and reduce GPU VRAM, we follow SV4D to precompute the VAE latents and CLIP embeddings of all images in advance. To form a training batch, we randomly sample 4 views and 12 frames of each object from the 24 rendered views and 24 frames. We train SV4D 2.0 on 16 NVIDIA H100 GPUs with an effective batch size of 32 (each GPU fits

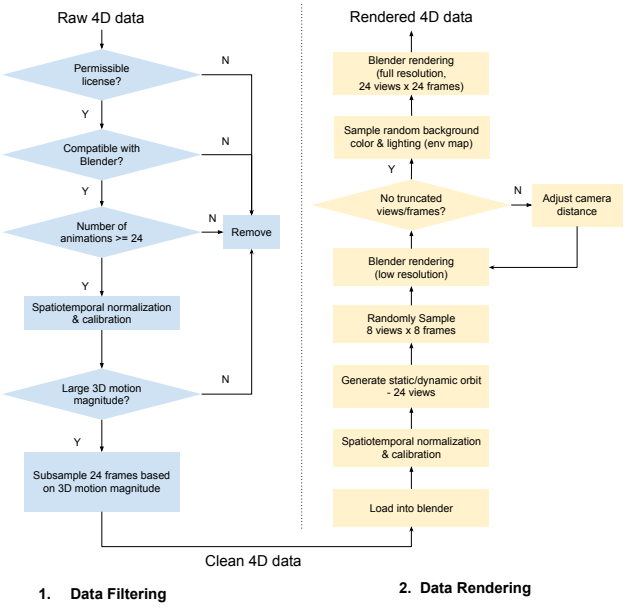

Figure 1. **Detailed data processing pipeline.**

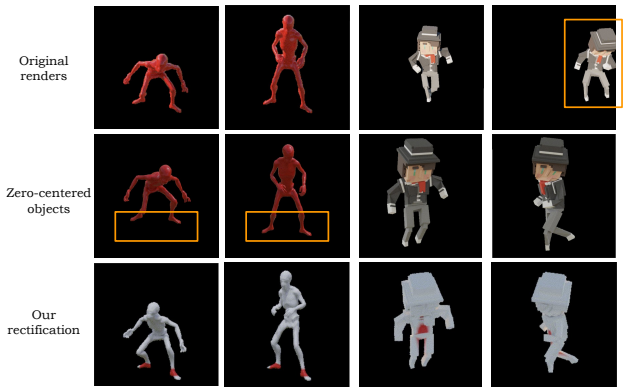

Figure 2. **Rectifying off-center objects.** We show in the top row (right) that some objects move off-center during motion. In the second row, we demonstrate that naively zero-positioning the center of object does not apply to some objects (left) as the object center is not always static. Instead, we propose rectification by identifying the most static region (marked in red) and disentangling the global translation from local motion, as shown in the bottom row.

Table 1. **Ablation study of SV4D 2.0 novel-view video synthesis on the ObjaverseDy dataset [4, 11].** By showing better metrics of SV4D 2.0 compared to the ablated models, we justify our model design of 1) using VAE latents of reference multi-views to optionally condition frame attention, 2) adding 3D attention with camera embedding as condition (instead of view indices), 3) improved data curation, and 4) progressive 3D-4D training.

Model	LPIPS↓	CLIP-S↑	PSNR↑	SSIM↑	MSE↓	FVD-F↓	FVD-V↓	FVD-Diag↓	FV4D↓
SV4D 2.0 w/o reference multi-views	0.135	0.870	17.07	0.870	0.0255	876.07	410.90	608.93	710.66
SV4D 2.0 w/ CLIP of reference multi-views	0.142	0.868	17.27	0.877	0.0244	1174.47	530.74	819.22	1327.75
SV4D 2.0 w/o 3D attention	0.115	0.934	17.89	0.878	0.0214	548.14	368.68	463.59	283.10
SV4D 2.0 w/ view index conditioning	0.121	0.928	18.08	0.881	0.210	660.67	401.48	507.55	450.12
SV4D 2.0 w/o improved data	0.112	0.931	18.19	0.884	0.0192	576.66	344.68	467.57	316.91
SV4D 2.0 w/o progressive 3D-4D training	0.140	0.896	16.50	0.871	0.0284	727.60	451.78	662.22	475.44
SV4D 2.0 (Ours)	0.105	0.939	18.67	0.886	0.0180	473.90	272.85	426.92	256.84

Table 2. **Ablation study of 4D optimization on the ObjaverseDy dataset [4, 11].** We show that our stage-2 refinement can significantly improve the 4D outputs in terms of image quality (LPIPS, PSNR, MSE, etc) and 3D consistency (FVD-V). The proposed orthogonal view and progressive frame sampling also achieves slightly higher image quality (LPIPS, CLIP-S, PSNR, etc) and temporal consistency (FVD-F) compared to random sampling.

Model	LPIPS↓	CLIP-S↑	PSNR↑	SSIM↑	MSE↓	FVD-F↓	FVD-V↓	FVD-Diag↓	FV4D↓
SV4D 2.0 w/o stage-2 refinement	0.124	0.910	17.94	0.876	0.0194	699.84	481.80	586.31	560.83
SV4D 2.0 w/ random view & frame sampling	0.105	0.918	19.41	0.889	0.0153	644.15	378.77	491.42	495.67
SV4D 2.0 (Ours)	0.104	0.921	19.44	0.889	0.0154	637.35	355.45	472.98	511.90

2 batches). The model is first trained on static 3D data for 175k iterations then finetuned on 4D data for 185k iterations.

1.3. 4D optimization details

The main supervision in our 4D optimization comes from the SV4D 2.0 NVVS results, which we sample from noise though 50 denoising steps. For stage-2 refinement, we add noise to the initial renders at noise timestep 25 and then denoise the images as improved pseudo ground-truths. Our optimization losses include the visibility-weighted MSE $\mathcal{L}_{\text{mse}} = \|\boldsymbol{W}(\boldsymbol{M} - \hat{\boldsymbol{M}})\|^2$, LPIPS [12] loss $\mathcal{L}_{\text{lpips}}$, mask loss $\mathcal{L}_{\text{mask}} = \|\boldsymbol{S} - \hat{\boldsymbol{S}}\|^2$, and a normal loss $\mathcal{L}_{\text{normal}} = 1 - (\boldsymbol{n} \cdot \bar{\boldsymbol{n}})$, where $\boldsymbol{n}$ and $\bar{\boldsymbol{n}}$ are the rendered normal and estimated pseudo ground truths from Omnidata [5], respectively. We also adopt a smooth depth loss and bilateral normal smoothness loss proposed in SV3D [10] to regularize the output geometry. The overall objective is defined as the weighted sum of these losses. We use an Adam optimizer [7] with a learning rate of 0.01 to update all parameters for 1500 iterations in stage 1 and 500 iterations in stage 2, taking roughly 20-25 minutes per object.

2. Ablative Analyses

2.1. Novel-view video synthesis

In Tab. 1, we report the ablative results of SV4D 2.0 model on the ObjaverseDy dataset [4, 11], which justify our design choices and contributions.

Optional reference multi-views. We first show that a fully multi-view unconditional model (w/o reference multi-views) performs considerably worse since it cannot leverage previous generation as multi-view condition to maintain temporal coherence between multiple generations. Next, we show that using VAE latents of reference multi-views produces better results than using CLIP embedding, as VAE latents include more detailed spatial features whereas CLIP embedding contains high-level semantic and global texture information.

3D attention and camera embedding conditioning. To ablate the network architecture, we show that using view attention in SV4D [11] (instead of 3D attention) and adding view indices as conditioning (instead of camera embedding) leads to worse image quality and multi-view consistency, especially when generating sparse novel views.

Improved data curation. We also report the model performance without using the improved data in ObjaverseDy++ for training, which further strengthens the effectiveness of our data curation approaches.

Progressive 3D-4D training. Finally, we show the importance of progressive 3D-to-4D training by evaluating the model directly trained on 4D data.

2.2. 4D optimization

In Tab. 2, we show the ablative results of our 4D optimization method on the ObjaverseDy dataset [4, 11]. In particular, we show that the proposed stage-2 refinement significantly improves all image and video metrics. The orthogonal view and progressive frame sampling also leads to better image quality (LPIPS, CLIP-S, PSNR) and temporal consistency (FVD-F) compared to random view and frame sampling.

Figure 3. **Dependency of reference multi-views.** SV4D [11] relies on the reference multi-views produced by SV3D [10], which often conflicts with the later frames of the input video (*e.g.*, jacket hood of the snowboarder and three arms of the dancing women) and leads to blurry outputs. In contrast, SV4D 2.0 can better leverage the information in all input frames to produce sharper and more faithful details.

2.3. Limitations

Since we follow SV3D [10] and SV4D [11] to use elevation and azimuth (2 degree of freedom) as our camera parametrization, the model cannot handle videos with strong perspective distortion. Additionally, SV4D 2.0 focuses on object-centric videos, limiting its applicability to general video applications like dynamic 3D scene generation. Generalizing the model to scene-level videos poses an interesting future work.

3. Additional Results

3.1. Conflicting reference multi-views

To demonstrate the importance of removing dependency on reference multi-views, we show some examples of reference multi-views conflicting with the input videos in Fig. 3. Such cases often occur when the subject occludes itself (right) or shows different views in the video due to rotation or camera motion (left). This forces the model to merge conflicting information in the input video and reference multi-views, resulting in blurry details as shown in SV4D outputs.

3.2. Robustness to occlusion

In Fig. 4, we further show an example of self-occlusion in the input video. Since the reference multi-view generation by SV3D [10] is only conditioned on the first frame of the input video, it fails to capture all 4 legs in several novel views. This results in inconsistent or blurry novel-view videos synthesized by SV4D [11]. In contrast, SV4D 2.0 generates consistent details by taking in multiple input frames as condition, showing higher robustness to occlusion.

3.3. Novel-view video synthesis

We show additional NVVS results on the real-world DAVIS [2] videos in Fig. 5, demonstrating higher-fidelity details and better generalization capability of SV4D 2.0 compared to SV4D [11].

3.4. 4D optimization

In Fig. 6, we show additional 4D results on the ObjaverseDy [4, 11] and DAVIS [2] videos. The RGB and normal renders demonstrate that SV4D 2.0 can capture higher-quality texture and geometry details in large motion compared to SV4D [11].

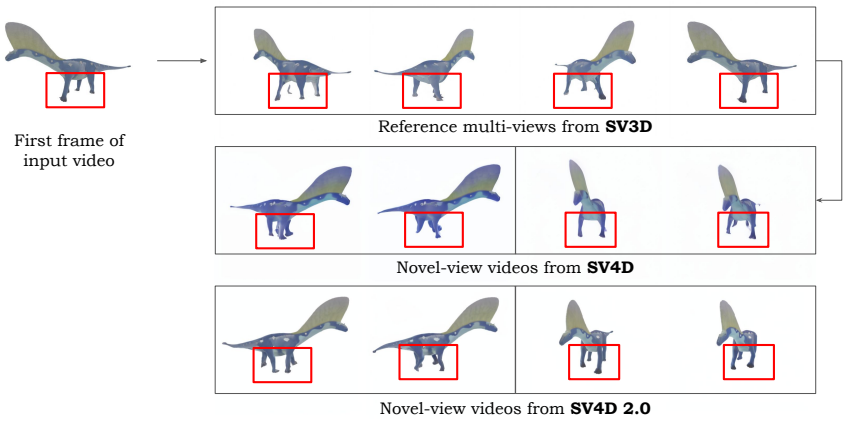

Figure 4. **Robustness to occlusion.** SV4D [11] relies on the reference multi-views generated by SV3D [10], conditioning on the first frame of the input video. We show that it suffers from self-occlusion in the first frame and often produces results with blurry artifacts or missing parts. In contrast, SV4D 2.0 is more robust to such occlusion by jointly taking multiple input frames to generate the novel-view videos.

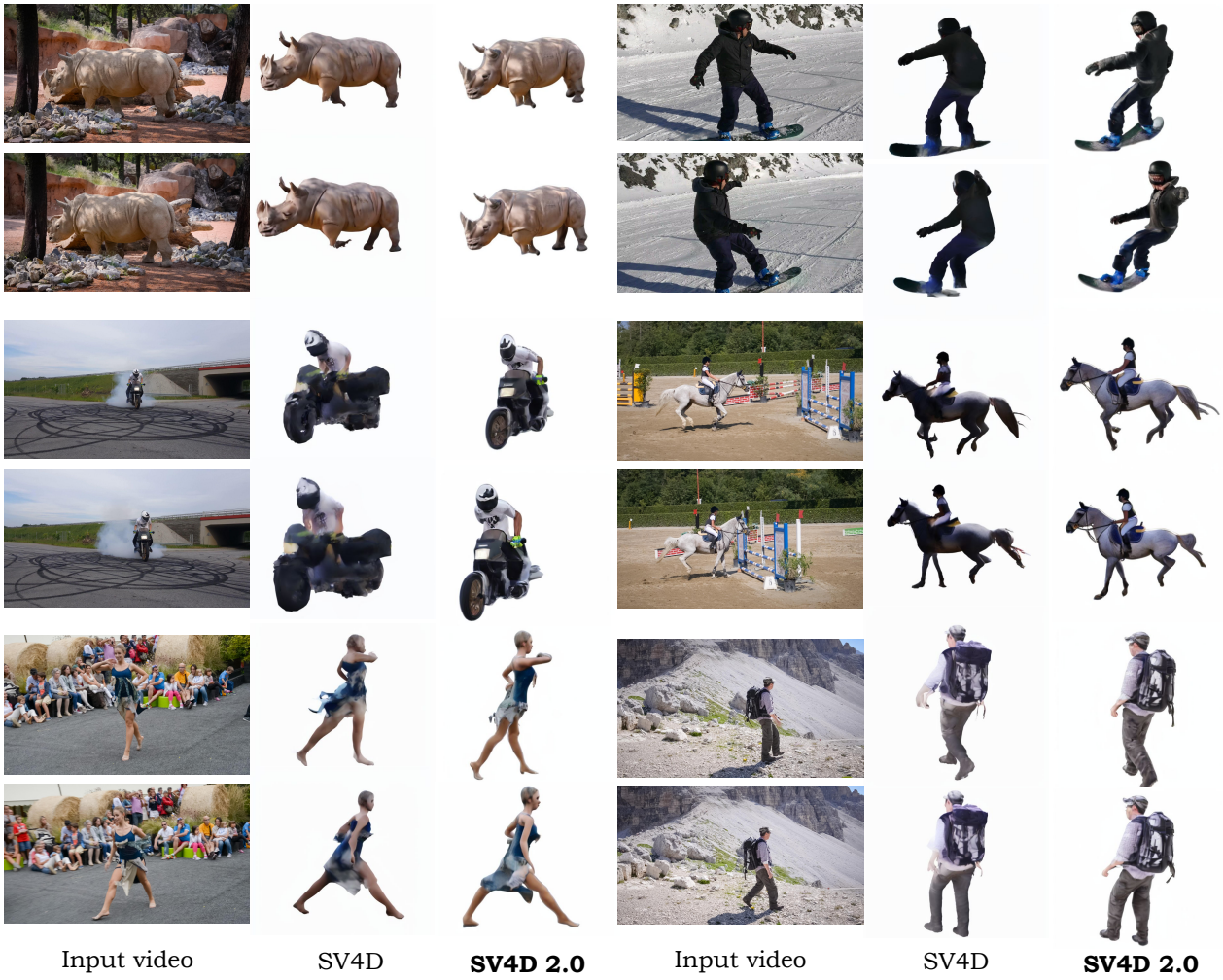

Figure 5. **Additional results of novel-view video synthesis on DAVIS [2].** We show that SV4D 2.0 can generalize better to the real-world data and produce higher-fidelity videos compared to SV4D [11].

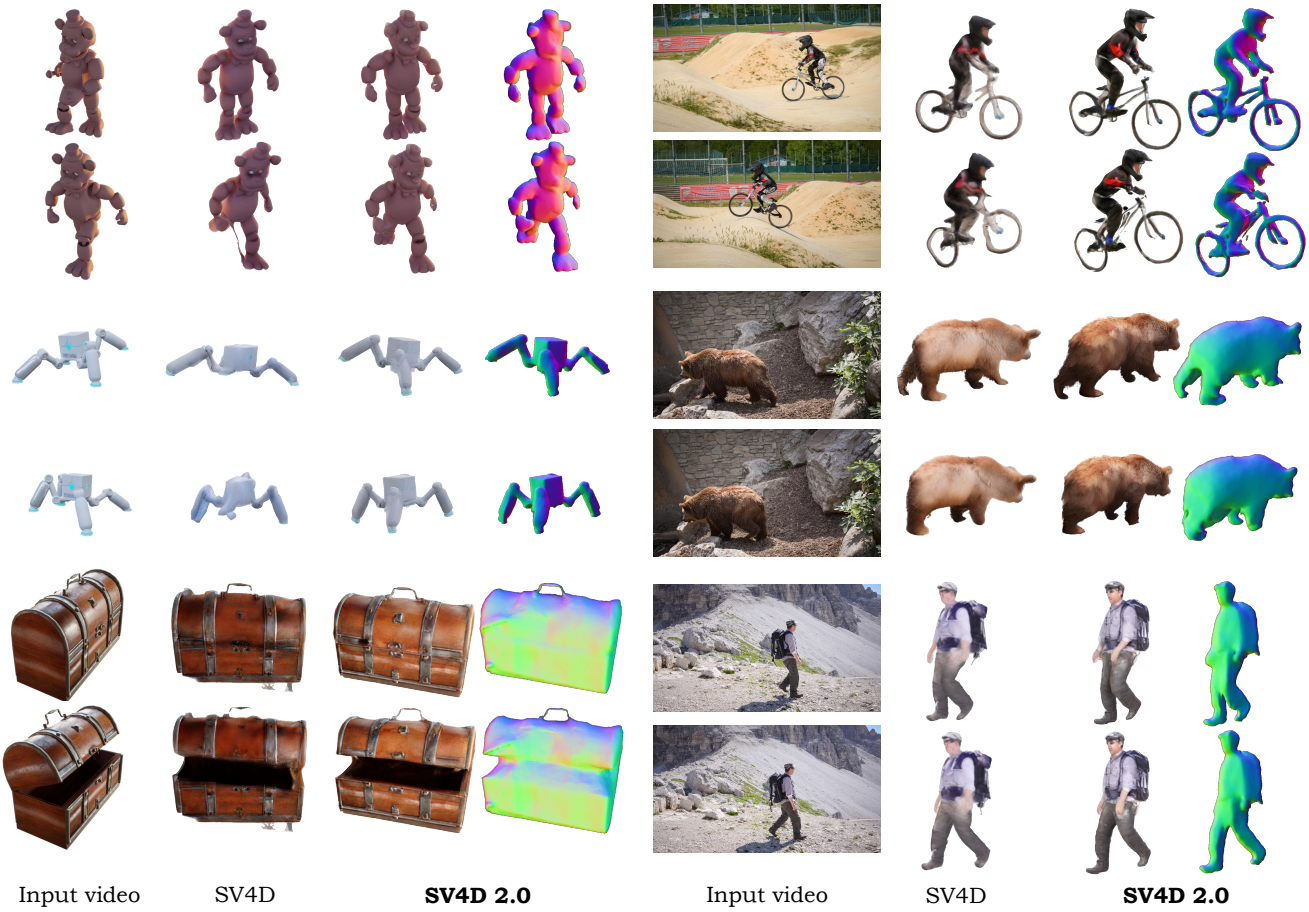

Figure 6. **Additional results of 4D optimization on ObjaverseDy [4, 11] (left) and DAVIS [2] (right).** We show the RGB renders of 4D outputs by SV4D [11] and SV4D 2.0, as well as our rendered normals. By comparison, our 4D assets can better capture the intricate geometry and texture details during motion.

3.5. 4D representation: DyNeRF *vs* 4D Gaussian

We show some results using 4D Guassians in Fig. 7. As mentioned in SV4D [11], 4D Gaussians suffer from temporal flickering and blurry artifacts due to its discrete nature, while Dy-NeRF interpolates better across sparse views and fast motion.

3.6. More L4GM results

We show more comparisons with L4GM on real-world videos in Fig. 8 and video with non-zero elevation in Fig. 9. L4GM does not generalize well on real-world data (no video prior like ours) and struggles with videos at non-zero elevation (training data primarily at 0° elevation).

3.7. Efficiency analysis.

Our model takes roughly 1 minute to generate 4 videos of 12 frames at 512 $\times$ 512 resolution on a H100 GPU, and 21 minutes for 4D optimization. It requires up to 30GB of memory.

4. Supplementary Video

In addition to the paper and appendix, we also include a comprehensive video in the supplementary material, providing an in-depth introduction to our framework and more visual results to demonstrate the effectiveness of SV4D 2.0.

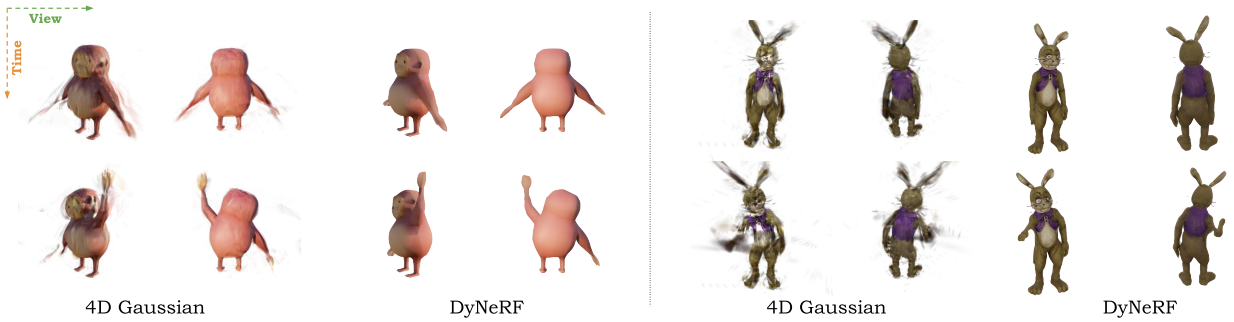

Figure 7. **4D Optimization with 4D Gaussians.** 4D Gaussians suffer from temporal flickering and blurry artifacts due to its discrete nature, while Dy-NeRF interpolates better across sparse views and fast motion.

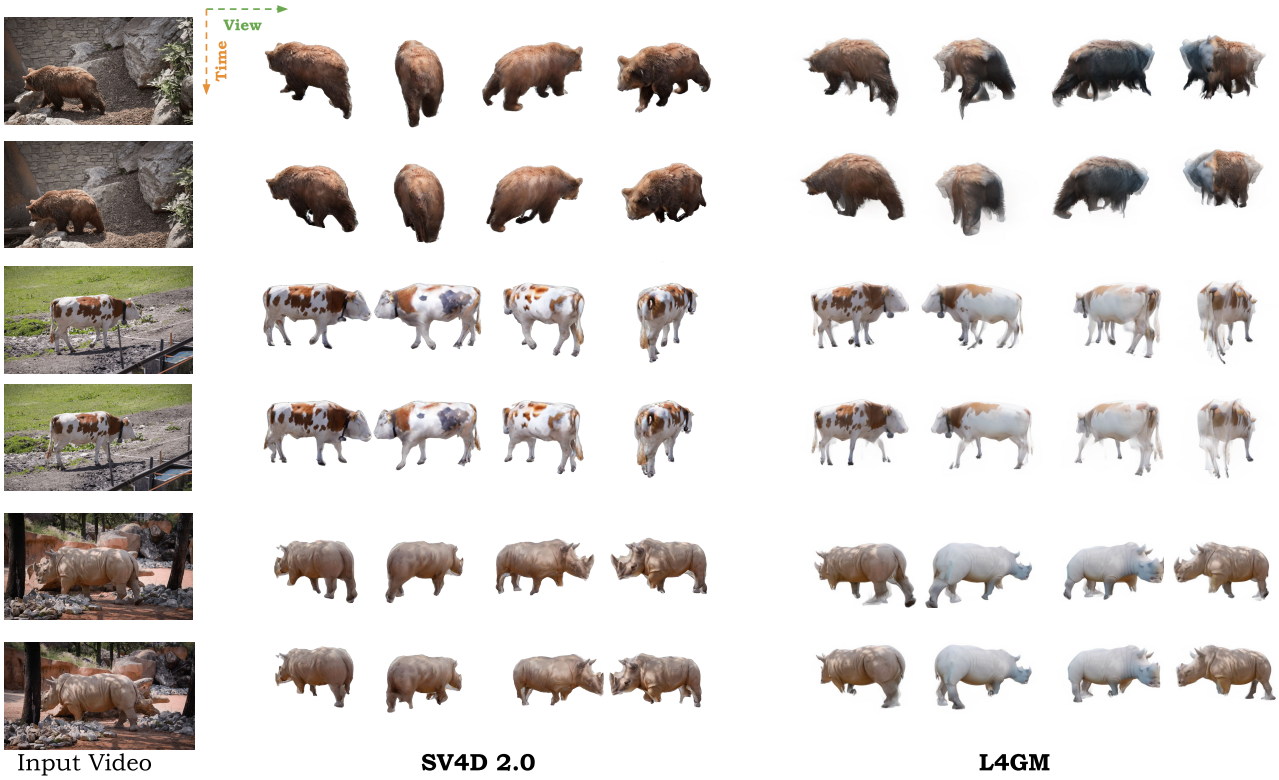

Figure 8. **Comparing to L4GM with real-world video input.** Compared to the L4GM, SV4D 2.0 generalizes better with real-world inputs because we leverage video priors by resuming pre-trained weights from the video generative model.

References

[1] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *CVPR*, 2023. 1

[2] Sergi Caelles, Jordi Pont-Tuset, Federico Perazzi, Alberto Montes, Kevis-Kokitsi Maninis, and Luc Van Gool. The 2019 davis challenge on vos: Unsupervised multi-object segmentation. *arXiv preprint arXiv:1905.00737*, 2019. 3, 4, 5

[3] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. Objaverse-xl: A universe of 10m+ 3D objects. In *NeurIPS*, 2023. 1

[4] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3D objects. In *CVPR*, 2023. 1, 2, 3, 5

[5] Ainaz Eftekhar, Alexander Sax, Roman Bachmann, Jitendra Malik, and Amir Zamir. Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3D scans. In *ICCV*, 2021. 2

[6] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *NeurIPS*, 2022. 1

[7] Diederik P Kingma and Jimmy Ba. Adam: A method for

Figure 9. **Comparing to L4GM with non-zero elevation video input.** Compared to the L4GM, SV4D 2.0 generalizes much better on non-zero elevation inputs. L4GM was trained with only 0°elevation videos.

stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2

[8] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 1

[9] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 1

[10] Vikram Voleti, Chun-Han Yao, Mark Boss, Adam Letts, David Pankratz, Dmitrii Tochilkin, Christian Laforte, Robin Rombach, and Varun Jampani. SV3D: Novel multi-view synthesis and 3D generation from a single image using latent

video diffusion. In *ECCV*, 2024. 1, 2, 3, 4

[11] Yiming Xie, Chun-Han Yao, Vikram Voleti, Huaizu Jiang, and Varun Jampani. SV4D: Dynamic 3D content generation with multi-frame and multi-view consistency. In *ICLR*, 2025. 1, 2, 3, 4, 5

[12] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, pages 586–595, 2018. 2