CAT4D: Create Anything in 4D with Multi-View Video Diffusion Models

Supplementary Material

A. Method Details

Diffusion Model We initialize our model from CAT3D [17], with the additional MLP layers in the new timestamp embedding randomly initialized. All timestamps $t \in \mathrm{T}^{\text{cond}} \cup \mathrm{T}^{\text{tgt}}$ (of each set of $M+N$ frames) are normalized within range $[0, 1]$ and are relative to the first timestamp $\mathrm{T}_0^{\text{cond}}$. We fine-tune the full latent diffusion model (*i.e.*, the denoising U-Net) with $M = 3$ input views and $N = 13$ target views for 2.0M iterations with a batch size of 128 and a learning rate of 5×10^{-5}. For the 4D reconstruction application, in order to condition on more input frames, we further fine-tune the model with $M = 9$ input views and $N = 8$ target views for 20K iterations.

Sampling For all of our experiments, we use DDIM [64] with 25 sampling steps and classifier-free guidance weights $s_1 = 3.0, s_2 = 4.5$. Our alternating sampling strategy takes about 1 minute to generate all $K' = 128$ views for each timestamp, when executed in parallel on 16 A100 GPUs. We note that CAT3D originally use 50 DDIM steps, yet for our model we found that 25 steps work just as well—and using fewer steps reduces the runtime of our sampling strategy.

Camera Trajectory Selection The choice of camera trajectories where we generate novel views has a large impact on the quality of 4D creation. In principle, the camera trajectories should cover the viewpoints where we want to render the scene after reconstruction. We design the novel-view camera trajectory based on the camera trajectory of the input video:

- For input videos with sufficient view coverage (*e.g.*, videos from DyCheck [16] whose cameras are centered around a focus point), we simply sample views on the input camera trajectory.
- For input videos with a forward-moving camera trajectory, we sample novel views from a spiral path around the input camera trajectory.
- For input videos with little or no camera movement, we sample novel views from either a spiral path that moves into and out of the scene or a orbit path that spins around the central object. For each example, we run both, and select the one which is most appropriate for the given scene.

See Fig. 10 for an illustration of different types of camera trajectories.

Sparse-View Bullet-Time 3D Reconstruction The conditional times T^{cond} should in principal be the actual timestamps of the input frames, but they may not be known for unstructured in-the-wild datasets. We found that in practice the model works well if we just set the timestamps for the bullet-time frame as 0 and other frames as 1.

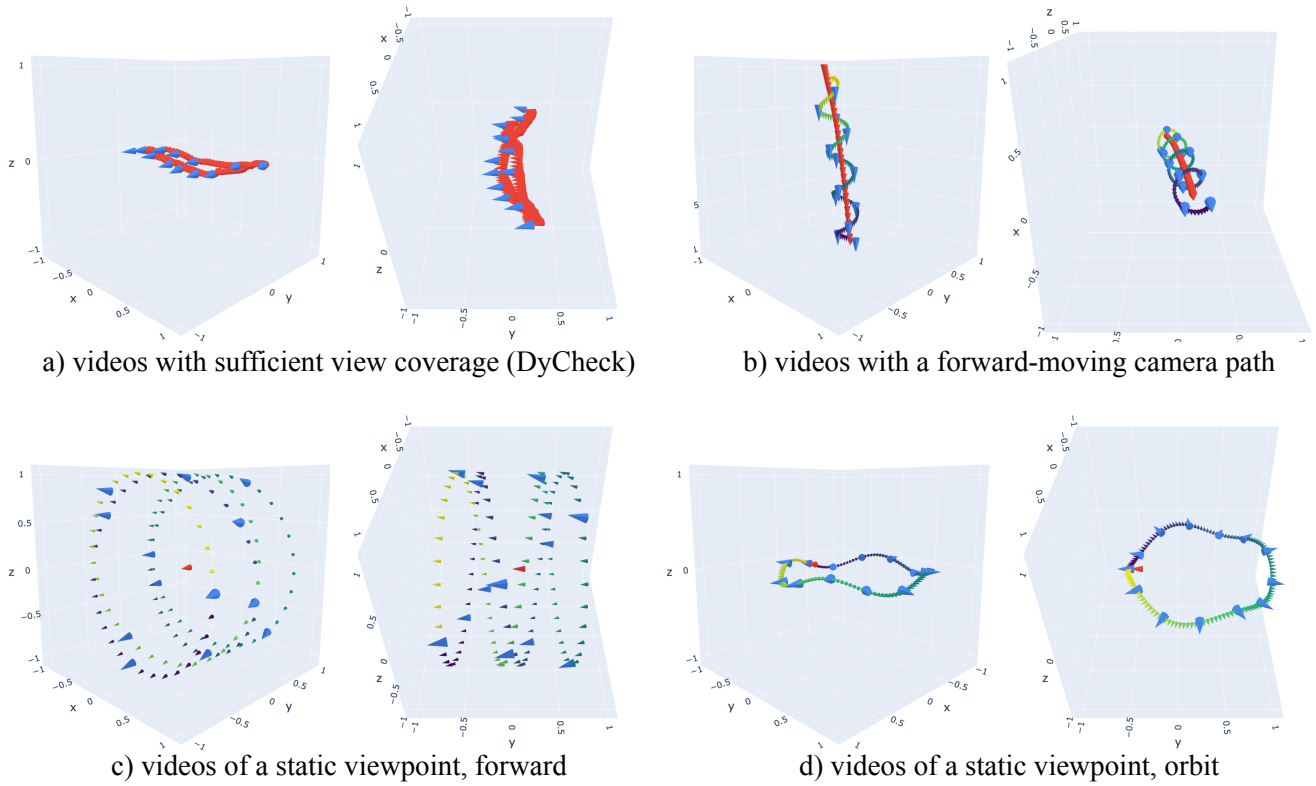
4D Reconstruction We build our reconstruction pipeline on top of 4D-GS [72] with several extensions. We use a combination of L1, DSSIM and LPIPS for the photometric reconstruction loss, with weighting factors 0.8, 0.2 and 0.4 respectively, and keep all regularization terms from [72] as is. We set the 3D Gaussians' densification threshold (magnitude of view-space position gradients) to 0.0004, and use a batch size of 4 (images). We initialize the 3D Gaussians with points from SfM [63] or MonST3R [85]. When SfM or MonST3R points are not available, *e.g.*, input videos of a static viewpoint, we use uniformly random points for initialization. We first optimize only the canonical-space 3D Gaussians with all generated images at $t = 0$ for 2000 iterations, then jointly optimize both the 3D Gaussians and the deformation field with images at all timestamps for 18000 iterations. After the first 2000 iterations, we linearly anneal the multiplier of reconstruction loss for our generated images from 1.0 to 0.5 while keeping the multiplier for real input images fixed to 1.0. The optimization takes about 25 minutes on a single A100 GPU.

B. Datasets Details

For Objaverse [12], we use only the animated assets filtered by [38] (around 42k in total). We render each asset under 4 different lighting conditions (randomly sampled environment maps). For each lighting condition, we render synchronized videos of 8 frames at 8 evenly spaced viewpoints on the 360° orbit path. For Kubric [19], we randomly generate 4k scenes using their generator. For each scene, we render synchronized videos of 8 frames at 8 viewpoints evenly spaced on a smooth camera path that is randomly sampled on the upper hemisphere. When drawing samples from these two synthetic 4D datasets, we sample the input and target views according to different combinations in Table 1 with equal probability.

For data samples from all multi-view image datasets [35, 54, 82, 94], we set all timestamps $\mathrm{T}^{\text{cond}} \cup \mathrm{T}^{\text{tgt}}$ to zero.

For our video dataset, we filter it to contain only videos of at static viewpoint. We perform this filtering by checking if the four corner patches (size 10×10) of each video are nearly constant over time. Concretely, we compute for each corner patch the L2 distance between consecutive frames (averaged over time), and then check if the maximum of the four is smaller than 0.05. While this simple strategy sometimes yields false positives, it's sufficiently effective and can be run on the fly. For samples from this dataset, we ran-



a) videos with sufficient view coverage (DyCheck)

b) videos with a forward-moving camera path

c) videos of a static viewpoint, forward

d) videos of a static viewpoint, orbit

Figure 10. Camera trajectories (where we generate novel views) for different types of input videos. Within each panel, we show the trajectories from two different viewpoints. The input views are colored red, and the anchoring sample views are colored blue with the remaining sample views are colored by their index. For videos with sufficient view coverage (a), we only generate anchor views picked from the input camera trajectory.

domly shuffle the order of frames and set all camera parameters $\mathrm{P}^{\text{cond}} \cup \mathrm{P}^{\text{tgt}}$ to be the same with a central principal point and a random focal length sampled from $[0.8 \cdot 512, 1.2 \cdot 512]$.

We use Lumiere [7] to augment the CO3D dataset [54] as follows (4-th row in Table 1). For each sampled sequence $(\mathrm{I}^{\text{cond}}, \mathrm{P}^{\text{cond}}, \mathrm{T}^{\text{cond}}, \mathrm{I}^{\text{tgt}}, \mathrm{P}^{\text{tgt}}, \mathrm{T}^{\text{tgt}})$ from CO3D, we animate each of the M input images (except the first one) using Lumiere [7], resulting in $M-1$ videos $\{V_L^i\}_{i=1}^{M-1}$ of length L. Then we randomly sample one frame (index k_i) from each video, and treat them as pseudo ground truth of original input images at another timestamps, *i.e.* $\mathrm{I}^{\text{cond}} \leftarrow \{\mathrm{I}_0^{\text{cond}}\} \cup \{V_{k_i}^i\}_{i=1}^{M-1}$ and $\mathrm{T}^{\text{cond}} \leftarrow \{\mathrm{T}_0^{\text{cond}}\} \cup \{\frac{k_i}{L-1}\}_{i=1}^{M-1}$. We obtain around 24k sequences in total with this augmentation.

We use CAT3D [17] to augment our static-view video dataset as follows (5-th row in Table 1). For each sampled sequence $(\mathrm{I}^{\text{cond}}, \mathrm{P}^{\text{cond}}, \mathrm{T}^{\text{cond}}, \mathrm{I}^{\text{tgt}}, \mathrm{P}^{\text{tgt}}, \mathrm{T}^{\text{tgt}})$ from the video dataset, we use CAT3D to generate 7 novel views for each of the M input images (except the first one), resulting in $M-1$ image sets $\{V_L^i\}_{i=1}^{M-1} (L=7)$ at viewpoints $\{\tilde{P}^i\}_{i=1}^{M-1}$. Then we randomly sample one frame (index k_i) from each image set, and treat them as pseudo ground truth of original input images at another viewpoints, *i.e.* $\mathrm{I}^{\text{cond}} \leftarrow \{\mathrm{I}_0^{\text{cond}}\} \cup \{V_{k_i}^i\}_{i=1}^{M-1}$ and $\mathrm{P}^{\text{cond}} \leftarrow \{\mathrm{P}_0^{\text{cond}}\} \cup \{\tilde{P}_{k_i}^i\}_{i=1}^{M-1}$. We obtain around 160k sequences in total with this augmentation.

We mix all the datasets (Objaverse [12], Kubric [19], Re10K [94], MVImgNet [82], CO3D [54], MQ4K [35], static-view video data, augmented CO3D and augmented static-view video data) with weights 2.5, 2.5, 1.0, 1.0, 1.0, 1.0, 5.0, 1.0 and 1.0, respectively.

C. Baselines Details

For the evaluation of sparse-view bullet-time 3D reconstruction, we run CAT3D baselines with one or three input images. For CAT3D-3cond, we use the same camera trajectory as ours (Fig. 10 (c)) for generating novel views. For CAT3D-1cond, we use their default camera trajectory (a forward-facing spiral path similar to Fig. 10 (c)) for generating novel views, and manually adjust the global scene scale such that it roughly matches the actual scene scale of the dataset.

For the ablation study of different sampling strategies, we evaluate the quality of the generated multi-view videos on the NSFF dataset [36]. As in the prior work [36, 79], we

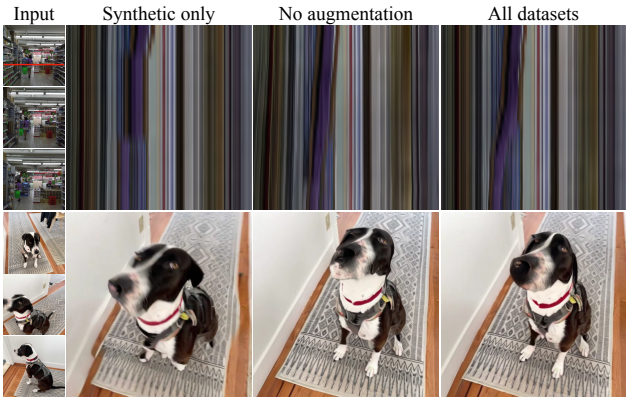


Figure 11. A comparison of models trained with different datasets on in-the-wild input images. The three input images are shown on the left-most column. Top: space-time slices of generated videos of "fixed viewpoint, varying time". Pixels of static background should be straight vertical lines on the slices and pixels of dynamic object should be smooth curves on the slices. Bottom: one frame of generated videos of "varying viewpoint, fixed time".

simulate a moving monocular camera by extracting images from each of the 12 camera viewpoints at different timestamps (24 in total) and compare the generated multi-view videos at all 12 viewpoints to the ground truth.

D. Ablation Study of Training Data

Training Data	Fixed Viewpoint Varying Time			Varying Viewpoint Fixed Time			Varying Viewpoint Varying Time		
	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
Synthetic only	22.19	0.745	0.123	21.41	0.547	0.123	19.50	0.523	0.173
No augmentation	20.84	0.596	0.135	22.03	0.602	0.104	19.41	0.519	0.160
All datasets	22.49	0.749	0.110	21.86	0.599	0.105	19.74	0.546	0.152

Table 6. A ablation study of training data, evaluated on the NSFF dataset [36]. All datasets: using all of our training datasets. No augmentation: dropping the two augmented datasets (CO3D augmented with Lumiere and static-view video data augmented with CAT3D). Synthetic only: dropping all real-world datasets and using only synthetic 4D data (Kubric and Objaverse).

We also perform an ablation study of our training datasets. We train our model with 1) all datasets listed in Table 1, 2) all datasets except the two augmented datasets (CO3D augmented with Lumiere and static-view video data augmented with CAT3D), 3) synthetic 4D datasets only (Kubric and Objaverse). For all three versions, we train the model for 60k iterations, and evaluate its ability for separate camera and time control on the NSFF dataset [36].

The quantitative results are presented in Table 6. While the numbers themselves do not show a large gap (as most of the evaluated pixels are static background), we observe more clear visual differences on in-the-wild data (see Fig. 11). For the model trained only on synthetic 4D datasets, it already gives surprisingly good control over camera and time, but the generated scene motions are often unnatural and the generated novel views usually look worse. This is likely a generalization issue. For the model trained without the two augmented datasets, its main failure mode is "fixed viewpoint, varying time" — in many cases, the camera still moves even when the model is instructed to only change the scene dynamics. This is potentially caused by our imperfect filtering of the video data, where some videos of non-static viewpoints are treated as static-view videos.