AC3D: Analyzing and Improving 3D Camera Control in Video Diffusion Transformers

Supplementary Material

We encourage the reader to inspect our visual results, comparisons with other models and additional visualizations in the accompanying website in https://snap-research.github.io/ac3d.

A. Ethics Statement

As with all generative AI technologies, there is the potential for misuse by bad actors. However, we anticipate this technology will advance creative expression, education, and research through:

1. Enhanced creative tools enabling filmmakers and educators to achieve complex camera movements without specialized equipment, democratizing high-quality video production and expanding possibilities for visual storytelling.
2. More realistic synthetic video that better simulates real world camera behaviors, improving applications in training autonomous systems, virtual production, and educational simulations where accurate camera dynamics are crucial.
3. Advancing our technical understanding of how camera motion affects visual perception and generation, contributing to fundamental research in computer vision, graphics, and human visual processing.

B. Limitations

In our work, we substantially advance the quality of 3D camera control of video diffusion models, but our analysis and method are not free from limitations.

OOD trajectories generalization. Both our model and all the baselines struggle to generalize to the camera trajectories, which are far away from the training distribution of RealEstate10K [199]. While, in general, it is an expected behavior, it indicates that the model processes the viewpoint conditioning information in a way that is entangled with the main video representations. Another source of this issue is the pre-training distribution of the base VDiT itself: natural videos typically have simple recording trajectories and rarely exhibit something that looks like 3D scanning. In this way, producing OOD trajectories is not an attempt to control existing knowledge of a video model, but an attempt to induce new knowledge into the model, which should require better and more diverse fine-tuning data.

Motion analysis limitations. As discussed in Appendix E, we estimate the optical flows in the latent space of CogVideoX [168] autoencoder rather than the pixel space, because it's the space the video DiT operates in and early trajectory steps produce disarranged decoder's outputs. Besides (as also observed by [75]), motion spectral volumes are sensitive to the quality of optical flow estimation. While the key conclusions (of the camera motion being low-frequency and kicking in very early in the diffusion trajectory) would hold since the are evident even with a bare eye from inspecting the denoising process visualization, the exact behavioral details of motion spectral volumes might change when an optical flow estimator is swapped or the analysis is moved from the latent to pixel space.

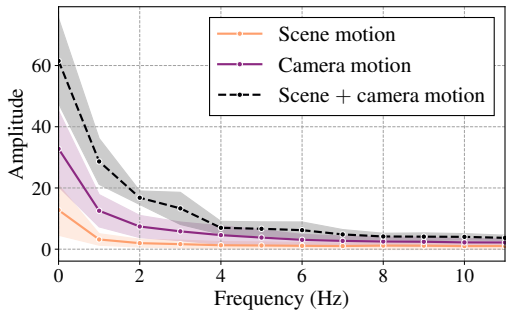
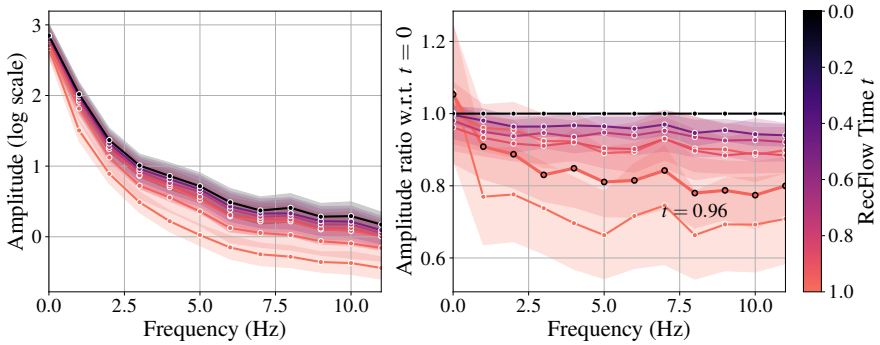


Figure 7. Comparing the average magnitude of motion spectral volumes for scenes with different motion types for CogVideoX [169]. Videos with camera motion (purple) exhibit stronger overall motion than the videos with scene motion (orange), especially for the low-frequency range, suggesting that the motion induced by camera transitions is heavily biased towards low-frequency components.

Linear probing limitations. We inspect the presence of disentangled camera information in the video DiT model of a particular architecture and trained on particular data. For it, we observe that the knowledge starts to arise from the 9-th block, but for a different instance of a video model it can be distributed across the blocks differently. Besides, we only evaluate it on RealEstate10K [199] test-set trajectories, and do not explore classical camera estimation datasets. Such analysis was sufficient to draw actionable conclusions and improve our base method, but there is vast room in making it more rigorous. We expect that video diffusion models are capable of revolutionizing the field of camera registration bringing strong priors about feature correspondences under difficult conditions, like scene motion, changing lighting and occlusions.

(a) A generated video at different diffusion timesteps. The camera has already been decided by the model even at $t = 0.9$ (first 10% of the denoising process) and does not change after that.



(b) Motion spectral volumes of VDiT's generated videos for different diffusion timesteps (left) and their ratio w.r.t. the motion spectral volume at $t = 0$ (i.e., a fully denoised video).

Figure 8. **How camera motion is modeled by diffusion (CogVideoX)?** As visualized in Figure 4a and Figure 3, the motion induced by camera transitions is a low-frequency type of motion. We observe that a video DiT creates low-frequency motion very early in the denoising trajectory: Figure 4b (left) shows that even at t=0.96 (first ≈4% of the steps), the low-frequency motion components have already been created, while high frequency ones do not fully unveil even till t=0.5. We found that controlling the camera pose later in the denoising trajectory is not only unnecessary but *detrimental* to both scene motion and overall visual quality. Here, we provide the same analysis conducted for CogVideox [169].

C. CogVideoX Results

We implement our method on top of CogVideoX [169] to show generalizability. Moreover, we conduct the motion analysis of the main paper for CogVideoX and show results in Fig. 7 and Fig. 8. We observe a similar pattern, confirming the generalizability of our findings.

D. Implementation details

This section describes the training and architectural details of the base VDiT, VDiT-CC , and our downstream experiments.

D.1. VDiT implementation details

Architecture details. Our video DiT [100] architecture follows a very similar design to the other contemporary video DiT models (e.g., [10, 32, 102, 168, 197]. As the backbone, we used a transformer-based architecture of 32 DiT blocks [100]. Each DiT block consists on a cross-attention layer to read the text embeddings information, produced by the T5 [108] model; a self-attention layer, and a fully-connected network with a $4\times$ dimensionality expansion. Each attention layer has 32 heads and RMSNorm [183] for queries and keys normalization. To encode positional information, we used 3D RoPE [123] attention, where each axis (temporal, vertical, and horizontal) had a fixed dimensionality allocated for it in each attention head (we split the dimensions in the ratio of 2:1:1 for temporal, vertical, and horizontal axes, respectively). LayerNorm [1] is used to normalize the activations in each DiT block. We used CogVideoX [168] autoencoder which is a causal 3D convolutional autoencoder with $4 \times 8 \times 8$ compression rate and 16 channels for each latent token. The hidden dimensionality of our DiT model is 4,096 and it has 11.5B parameters in total. Similar to DiT [100], we use block modulations to condition the video backbone on the rectified flow timestep information, SiLU [41] activations and 2×2 ViT-like [26] patchification of the input latents to reduce the sequence size.

Training details. We train the model with the AdamW [89] optimizer with the learning rate of 0.0001 and weight decay of 0.01. The model was trained for 750,000 total iterations with the cosine learning rate scheduler [90] in bfloat16. We also incorporate the support of image animation by encoding the first frame with the same CogVideoX encoder, adding random gaussian noise (with the noise level sampled independently from the video noise levels σ_t), projecting with a separate learnable ViT-like [26] patchification layer, repeating sequence-wise to match the video length and summing with the video tokens. During training, we use loss normalization [60]. The model is trained jointly on images and videos of variable resolution (256, 512 and 1024), aspect ratio (16 : 9 and 9 : 16 for videos, and 16 : 9, 9 : 16 and 1 : 1 for images), and video lengths (from 17 frames to 385 frames). The video framerate was set to 24 frames per second and we did not use variable-FPS training as contemporary works [93, 102] since we found it to decrease the performance for a target framerate (at least, without fine-tuning).

Inference details. During inference, we use the standard rectified flow without any stochasticity. We found that 40 steps gives a good trade-off between quality and sampling

speed. We follows the same time shifting strategy as Lumina-T2X for higher resolutions and longer video generation [32] with time shifting of $\sqrt{32}$ for the 1024 resolution.

D.2. VDiT-CC implementation details

As being said in Section 3.2, VDiT-CC is a simple ControlNet-like [71, 188] fine-tuning of VDiT for camera control. We use smaller versions of the base VDiT blocks with only a 128 hidden dimensionality and 4 attention heads. Besides, we do not use cross-attention over the context information since we found it to severely decrease the visual quality and camera control precision. For the Plücker encoding computation, we replicate the pipeline of VD3D [6]. Our linear layer that processes them contains 4096 hidden features and SiLU [41] non-linearity.

D.3. AC3D implementation details

We train our complete setup for 6K iterations on the joint dataset of 65K videos from RealEstate10K [199] and 20K dynamic videos with static cameras (this dataset is described in Section 3.5 and Appendix G). We train with the learning rate of 0.0001 using the AdamW [89] optimizer with weight decay of 0.01 and cosine learning schedule [90].

As described in Section 3.2, since the camera motion is a low-frequency type of signal, we propose to use truncated and biased noise schedule: both at train and inference time. In Figure 9, we visualize three distributions: 1) the standard one used by SD3 [28], our base VDiT, and "w/o biasing noise" experiment (orange); 2) the biased but non-truncated schedule used in the VDiT-CC "w/o noise truncation" ablation experiment (purple), which has some unnecessary and detrimental probability mass in the high-frequency range; and 3) our final schedule (red).

Each ablation experiment from Section 4.3 was trained for 6K iterations on 32 NVIDIA A100 80GB GPUs.

During training, to plug in the conditioning camera pose for our static dataset, we were randomly sampling extrinsics and intrinsics parameters from a RealEstate10K dataset.

1D temporal camera encoder. VDiT-CC processes videos in the autoencoder's latent space with $4\times$ temporal compression [168], raising the question of how to incorporate full-resolution camera parameters. While camera poses could be naively downsized (e.g., subsampled or reshaped) to match the latent resolution, this would force small `DiT-XS` blocks to process compressed camera information. Instead, we implement a sequence of causal 1D convolutions that transform a $F\times 6$ sequence of Plücker coordinates for each pixel into a $(F//4)\times 32$ representation.

E. Motion analysis details

As discussed in Section 3.3, we perform camera motion analysis of generated videos at different time steps by inspecting their spectral volumes.

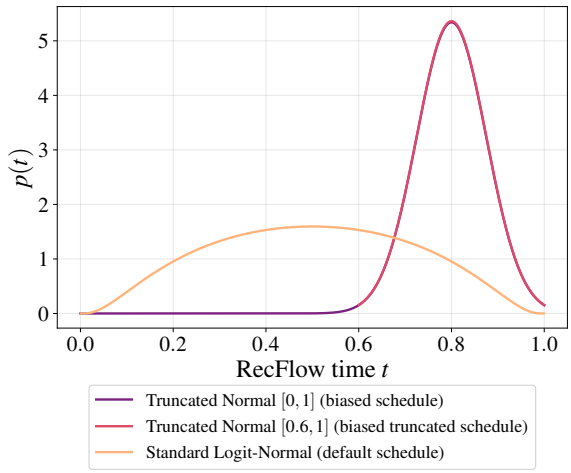


Figure 9. Comparing rectified flow noise schedules: (orange) vanilla standard logit-normal noise schedule proposed by [28] and used for baseline experiments; (purple) biased but non-truncated noise schedule; (pink) biased and truncated noise schedule.

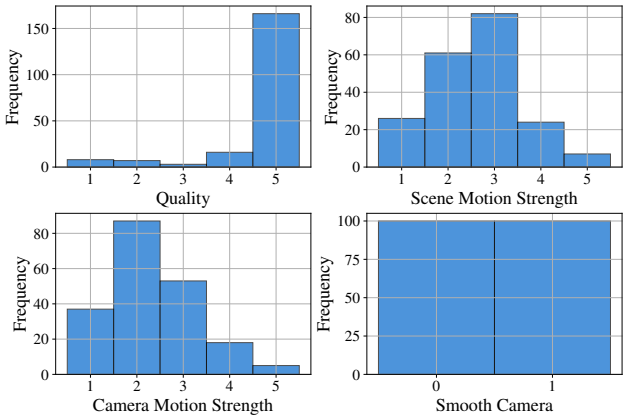


Figure 10. Our annotations collected for 200 randomly generated videos from VDiT and used in our camera motion analysis in Section 3.3.

For our analysis, we generate 200 random 121-frames videos without time shifting [32] in the 288×512 resolution with VDiT and manually annotate them for the following labels: quality (a score from 1 to 5), scene motion strength (a score from 1 to 5, where a score of 1 corresponds to a completely still scene), camera motion strength (a score from 1 to 5, where a score of 1 corresponds to a completely stationary viewpoint), whether the camera is smooth or shaky (a binary flag). We visualize the obtained scores in Fig. 10. Next we discard the videos with the quality score of less than 4, discarding 18 out of 200 videos: "broken" samples (e.g., an artificial animation or a blank black canvas) indicate a complete generation failure which we should exclude from the analysis of the camera motion.

Figure 11. Frames of the generated videos by the VDiT model (upper row) and the corresponding PCA projections of their latents (lower row).

To analyze the spectral volumes differences between scene and camera motion for Fig. 3, we extract three categories of videos from our dataset: 1) scene motion (videos with scene motion, but with no camera motion); 2) camera motion (videos with camera motion, but with no scene motion); and 3) scene and camera motion (videos with scene motion and non-artifactory camera motion). The first category (scene motion) was obtained by selecting the videos with the camera motion strength of 1 (i.e., no camera movement), and the scene motion strength of more or equal than 3; the second category (camera motion) was obtained by selecting the videos with the camera motion strength of more or equal than 3, and the scene motion strength of 1 (i.e., no scene motion); the third category (scene and camera motion) was obtained by selecting the videos with the camera and scene motion strengths of more or equal than 3 and the smooth camera flag being true (to exclude shaky camera movements). To analyze the spectral volumes for Fig. 4, we took the videos which have scene or camera motion strength of more or equal than 3.

To obtain spectral volumes, we need to obtain per-pixel optical flow information. Since our VDiT is following the latent diffusion (LDM) paradigm [111], we opt for performing optical flow estimation in the latent space of the autoencoder. There are two reasons for that: 1) we noticed that it provides more robust flow estimation at earlier denoising timesteps (since the decoder part of CogVideoX [168] autoencoder does not need to operate at out-of-distribution inputs); and 2) the video model operates in this space. In this way, we used the raw generated latents to estimate the optical flow. Since most of the optical flow algorithms operate in a 3-dimensional RGB space, and our latents are 16-channels, we projected them into 3-dimensional inputs via a PCA, computing it independently for each latent. We found that these representations maintain very strong spatiotemporal resemblance to the original videos, as visualized in Fig. 11.

Following [75], we use PyFlow [99] coarse-to-fine optical flow estimation to obtain more robust results. We attempted to use Farneback [29] optical flow estimation, but observed that it is less accurate and does align less with our visual evaluation of the results. 121-frames 288×512-resolution videos correspond to 31 latent frames of 36×64 resolution. For each 6-th latent frame, we estimate its flow with respect to each of the 24 subsequent frames. Next, we perform Fast Fourier transform for x and y spatial coordinates independently, compute the amplitudes and average them spatially for each video.

F. Linear probing details

In Section 3.4, we perform linear probing of VDiT for camera pose knowledge. For this, we use 1,000 videos of 49 frames and the 144×256 resolution from the test set of RealEstate10K [199] and extract their internal representations of our VDiT model under various noise levels. We use the noise levels σ_t of $[\frac{1}{8}, \frac{2}{8}, \frac{3}{8}, ..., \frac{7}{8}, 1]$. The hidden dimensionality of our VDiT model is $4,096$ and to reduce the memory requirements and speed up linear probing we project each video representations into 512 dimensionality using PCA. This results in latent representations of $512 \times 13 \times 18 \times 32$ for each block, each video and each timestep. To construct the training features for the linear regression model, we extract the (spatially) middle vector of shape 512×13 and perform spatial average pooling to obtain a context representation of 512×13. We then unroll and concatenate them to obtain the final training representation of dimensionality $13,312$. We split our 1,000 videos into training and test sets as 900 and 100 and then train a ridge linear regression with the regularization weight of 25,000. Our target variable covers the extrinsics parameters of the viewpoints, which are provided by RE10K. We extract rotation angles and translations from the extrinsic matrices and normalize them with respect to the first frame. Then, we compute the rotation and translation errors on the held-out set of 100 videos using the evaluation pipeline of CameraCtrl [39]. Since we have 8 noise values and 32 VDiT blocks, this resulted into 256 linear regression models in total. It was taking $\approx$5 minutes of CPU time to train each model, and their training was parallelizable across different cores.

G. Dataset construction details

As describe in Section 3.5, we construct a dataset of videos with scene motion but recorded from stationary cameras. One might attempt to build such a dataset in an automated way by estimating the optical flow and checking whether there is non-zero motion in the middle region of the frame, and no motion on the borders. However, this would not work well for a myriad of corner cases, which is why we opted for manual construction. For this, we annotated 110K internal videos for the presence of camera and the presence of scene motion. Each video has a duration of $5-30$ seconds and covers various diverse categories of scenes: humans, animals, landscapes, food and other types. Out of these videos, 20K videos turned out to be the necessary ones: with scene motion, but completely static scenes. We proceeded to use them as our training data without further processing. As being discussed in Appendix D, we augmented their

camera information artifically by using random extrinsics from RealEstate10K [199].

H. Scaling bias

Classical feature-based camera estimation [114, 115] outputs camera trajectories with arbitrary scale, as it does not possess any priors about the absolute size of objects in the scene. For example, a house might appear identical to a small object like an apple when viewed from the appropriate distance, making it impossible to determine absolute scale from visual features alone. This lack of scale awareness complicates precise user control over camera trajectories: while the model can determine the camera's direction, it remains unaware of the magnitude of each movement, as demonstrated in Sec. 4. Ideally, all camera trajectories should be aligned to a consistent reference scale. A natural reference would be a metric scale, now achievable due to recent advances in metric depth estimation [172]. We propose a method to rescale camera trajectories across all data using the following steps:

1. Obtain 3D points from the COLMAP output and render the COLMAP depth D_c^f from these points for each frame.
2. Estimate the metric depth D_m^f for each frame using a pre-trained zero-shot metric depth estimator [172].
3. Calculate the re-scaling parameter $\hat{\lambda}$ by solving the optimization problem $\hat{\lambda} = \arg\min_\lambda \mathbb{E}_{f \sim F} \left| \lambda D_c^f - D_m^f \right|$, where F is the total number of frames.
4. Set the camera translation vector $\hat{\boldsymbol{t}}_c$ to $\hat{\lambda} \boldsymbol{t}_c$.

In Sec. 4.3, we show that training with properly scaled cameras does not lead to visual quality degradation (even improving it slightly), and, as we demonstrate in our supplementary visuals, makes the camera control more predictable and less frustrating for a user by allowing to control the magnitudes of camera transitions.

I. Additional Related Work

Due to space constraints, we summarize related 3D and an extended list of 4D works in the appendix.

3D generation. Early efforts in 3D generation focused on training models for single object categories, extending GANs to 3D by incorporating neural renderers as an inductive bias [3, 13, 24, 97, 116]. As the field progressed, CLIP-based supervision [107] enabled more flexible and diverse 3D asset generation, supporting both text-based generation and editing [19, 35, 53, 55, 113, 142]. Recent advances in diffusion models further enhance generation quality by replacing CLIP with Score Distillation Sampling (SDS) for supervision [20, 40, 61, 66, 74, 77, 78, 86, 103, 124, 143, 148, 170, 178]. To improve the structural coherence of 3D scenes, several approaches generate multiple views of a scene for consistency [30, 34, 47, 63, 80, 83, 84, 117, 133, 140]. Alternatively, iterative inpainting has been explored as a technique for scene generation [46, 118]. Recent works also focus on lifting 2D images to 3D representations, employing methods like NeRF [95], 3D Gaussian Splatting [62], or meshes in combination with diffusion models [14, 36, 87, 88, 105, 128, 131, 134, 141, 174]. Other studies explore fast, feed-forward 3D generation techniques that directly predict 3D models from input images or text [38, 48, 56, 70, 104, 129, 130, 132, 135, 156, 162, 163, 184]. These methods, however, are limited to synthesizing static scenes, in contrast to our approach.

4D generation. There has been significant progress in 4D generation, i.e., dynamic 3D scene generation. These works often rely on input text prompts or images to guide the generation. Since the early advancements in large-scale generative models for this task [120], significant strides have been made in improving both the visual and motion quality of generated scenes [5, 33, 57, 58, 69, 81, 94, 109, 165, 179, 190, 196]. While many of these methods are conditioned on text input, other approaches focus on converting 2D images or videos into dynamic 3D scenes [12, 23, 31, 33, 67, 72, 73, 76, 81, 98, 109, 110, 126, 137, 138, 147, 154, 157, 167, 173, 182, 187, 193, 196]. Recently, several works [52, 79, 189] investigate physics priors in 4D generation pipelines. Other works [15, 125, 186] enhance motion controllability with template-based methods. Another line of work [4, 11, 159, 175, 181, 200] focuses on compositional and interactive 4D generation. Another strand of research extends 3D GANs into the 4D domain by training on 2D video data [2, 164]. However, the quality of these methods is often constrained by the limited nature of the datasets, which typically focus on single object categories. Moreover, the majority of these approaches tackle object-centric generation. As a result, they typically neglect background elements, and their visual fidelity falls short when compared to the high photorealism achieved by state-of-the-art video generation models, such as those employed in our approach.

Motion-controlled video generation. Orthogonal to camera-controlled video generation methods, another line of work investigates object trajectory control [144, 146, 153, 171, 191, 198]. More recently, several methods [54, 91, 96, 106, 166, 176] focus on object trajectory control without relying on additional external data or additional model fine-tuning.

J. User Study

In the user study, we engage 10 professional labelers, each of whom evaluates 100 different video pairs. The labelers are asked to choose between two videos based on several preference metrics:

- *Camera Alignment (CA):* How well the camera trajectory follows the reference video.
- *Motion Quality (MQ):* Which video has larger and more natural motion.

- *Text Alignment (TA):* Which video better aligns with the provided reference text prompt.
- *Visual Quality (VQ):* Which video has a higher overall visual quality.
- *Overall Preference (Overall):* Which generated video the user would prefer for this task.