Generative Video Motion Editing with 3D Point Tracks

Supplementary Material

In this supplementary material, we elaborate our method details (Sec. A), model analysis (Sec. B), and additional baseline comparisons (Sec. C). We also strongly encourage readers to view our webpage for full video results.

Contents

A. Method Details

A.1. Model and Training Details

We fine-tune our track-conditioned V2V model by using LoRA on the pre-trained DiT blocks and the 3D track-conditioner, which is initialized from scratch. The LoRA is applied to the MLPs in all attention modules of the DiT blocks, including the projection heads for query, key, value, and the feed-forward layers.

For our 3D track conditioner, we use a cross-attention module for sampling, followed by two self-attention Transformer blocks to aggregate temporal information within each track. Finally, the paired splatting branches use a shared-weight cross-attention model to project the processed tokens back to corresponding frame spaces. For both sampling and splatting cross-attention modules, we feed the positional embedding of track coordinates to both MLPs of query and key. The attention value is directly applied by the attention weight from the softmax operation without passing through an MLP beforehand. We use sinusoidal positional encoding to map each track's four input values (xyz coordinates and an existence label $\in \{0, 1\}$, see Sec. A.5) to a 128-dimensional embedding, matching the dimension of a single attention head (12 heads total).

The two-stage fine-tuning takes a total of 60 hours on 16 A100-80GB GPUs with a total batch size of 64. The first stage, using synthetic data, takes 4,000 iterations ($\sim$20 hours), and the second stage, using real data, takes 8,000 iterations ($\sim$40 hours). Our 3D track conditioner comprises 45M parameters, and the LoRA adapters for the DiT blocks account for a total of 87M parameters.

A.2. Synthetic Data Generation

To train our model to learn motion control, we generated a synthetic dataset of video pairs. The core principle of this dataset is that each pair shares the same objects and background scene but features different object actions and camera movements. This design allows the model to isolate motion changes from appearance. For the background scenes, we create backgrounds by applying textures from the Kubric [6] dataset onto a large, dome-shaped mesh that envelops the scene. For the foreground dynamic objects, we exploit the Mixamo human animation asset library [9]. We gathered 25 unique human characters and 64 distinct action animations, enabling a large combinatorial space of motions and appearances.

Our generation process starts by defining a base scene, which consists of a randomly selected background texture and a random number $[1, 4]$ of chosen human characters. For each base scene, we then generate four distinct video clips, each rendered with a different combination of camera movements and human actions. The ground-truth 3D point tracks are then extracted from the mesh vertices [28] of the rendered scenes. Notably, we randomly designate some foreground objects as "background dynamics." These specific objects perform the same action across all four clips, and we intentionally do not extract their point tracks. This strategy aims to train the model to preserve the motion of objects when tracks are not specified. We generated a total of 500 base scenes, resulting in a synthetic dataset of 2,000 video clips (four clips per scene). The training process randomly selects two clips from a base scene as the training input and ground truth each time.

A.3. Real Data Curation

Our training dataset for Stage 2 fine-tuning is built from several sources. The primary component consists of 24K real monocular videos curated from a large internal video dataset. To obtain clean data suitable for training, we first apply a filtering process to remove videos that con-

tain dynamic, cluttered objects, such as crowds on streets, flocks of birds, and fast-moving traffic. These 24K selected videos are then preprocessed to estimate camera pose and 3D tracks, as detailed in Sec. A.6.

We observed that while this internal dataset provides diverse motion of scene dynamics, its camera motion is often smooth and linear. This property limits the model's ability to learn from large and arbitrary viewpoint changes. To complement these videos and address this limitation, we augment the training set with the DL3DV [13] dataset, which specifically contains static-scene videos with large viewpoint changes.

Lastly, we integrate the object-effect-removal training pairs from Gen-Omnimatte [11] (comprising a total of 46 real-video pairs) into our training set. This enhances the model's ability to remove objects while preserving the remaining scene. During training, we set the sampling ratios for the final data mixture to 85% for our dynamic monocular videos, 10% for DL3DV [13], and 5% for Gen-Omnimatte [11].

A.4. Track Perturbation and Data Augmentation

For Stage 2 fine-tuning on real monocular videos, we apply data augmentations to the video frames and 3D tracks to better match the testing distribution. A primary challenge is that the source tracks, $\mathcal{T}_{\text{src}}$, estimated from monocular video are often inaccurate and noisy, particularly in their depth component. These inaccuracies are then propagated and frequently amplified when edited into target tracks, $\mathcal{T}_{\text{tgt}}$. For example, when applying a 3D rotation, a point in $\mathcal{T}_{\text{src}}$ with an incorrect depth value will be transformed to an incorrect 3D position, resulting in significant visual artifacts. To address this, we apply several point track perturbations directly to the target tracks ($\mathcal{T}_{\text{tgt}}$) during fine-tuning to improve the model's robustness to the inevitable noise during 3D track editing.

- Perturbing along epipolar line: To simulate noise from inaccurate depth, we perturb up to 10% of the target tracks. We use the estimated camera poses $(\mathcal{P}_{\text{src}}, \mathcal{P}_{\text{tgt}})$ to transform the target 3D tracks into the source camera's coordinate system. We then apply random jitter along the viewing ray, which ensures the same 2D projection but perturbs the depth component. Finally, these jittered target 3D tracks are reprojected back into the target video frames, which simulates 2D misalignments along the epipolar line caused by depth errors.
- Perturbing by random homography: Since jittering along epipolar lines can be limited in videos with small viewpoint changes, we apply an additional random homography perturbation. We begin by sampling a subset of 3D tracks (up to 10%) to be perturbed. We then randomly designate one frame to serve as an anchor. Following this, a series of per-frame homography matrices is obtained using randomly chosen four tracks within the subset, each matrix defining a random transformation for its corresponding frame relative to the anchor frame. These per-frame homographies are then applied to all tracks in each frame of the selected subset to simulate more complex and noisy tracks.
- Linear motion: To further simulate motion noise, we select up to 10% of the target tracks and apply a linear motion drift. This involves adding a consistent 2D velocity vector to each selected track in the 2D frame space, causing it to move uniformly in a specific direction throughout the video.

These track perturbations enhance the model's robustness to noisy 3D tracks during editing, and we provide a detailed analysis of this effectiveness in Sec. B.2.

In addition to 3D track perturbation, we also perform data augmentation on the two non-contiguous clips sampled from a monocular video.

- Source frame dropout: With simple camera motion in a training pair, the model may learn to over-rely on only the first and last source frames while ignoring intermediate content. To prevent this, we randomly zero out (mask) up to 50% of the source video frames, encouraging the model to utilize the visual context from the entire input video.
- Small clip overlap: While most training pairs are sampled from non-contiguous clips, we ensure that 5% of pairs have some temporal overlap (up to 50% of frames). This small subset of overlapping clips, when combined with our target track jittering, improves the model's ability to robustly preserve input details.
- Horizontal flipping: Since the source and target clips sampled from a single monocular video can be visually similar, we introduce further data variance by horizontally flipping the target clip and its corresponding tracks with a 50% probability.

These data augmentations enhance the model's robustness, ensuring it generalizes from our monocular training data to more complex editing scenarios during inference.

A.5. Existence Label for Object Removal

We introduce an existence label in the track inputs, designed specifically to control the object-removal task. By default, for non-removal tasks, this label is set to 1 for all tracks. To specify removal, we set the label to 0 for the target tracks $\mathcal{T}_{\text{tgt}}$ of an object to be removed. During training, we utilize the object-removal data from Gen-Omnimatte [11], either by setting the existence label to 0 for the target tracks, or by moving the tracks off-screen in the target video. At inference time, removing an object involves both setting the existence label of its associated tracks to 0 and moving those tracks off-screen.

We emphasize again that this existence label is separate

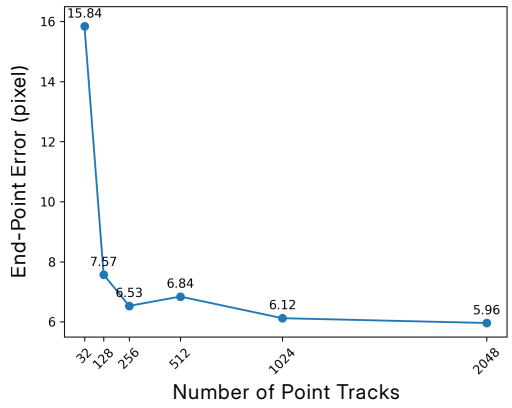

Figure 1. **Analysis on the sparsity of track inputs.** We evaluate End-Point Error (EPE) using 100 in-the-wild videos from MiraData [10] to test performance with different numbers of point tracks. All evaluations are run on the same final model, which was trained once using a random number of tracks between 500 and 1000.

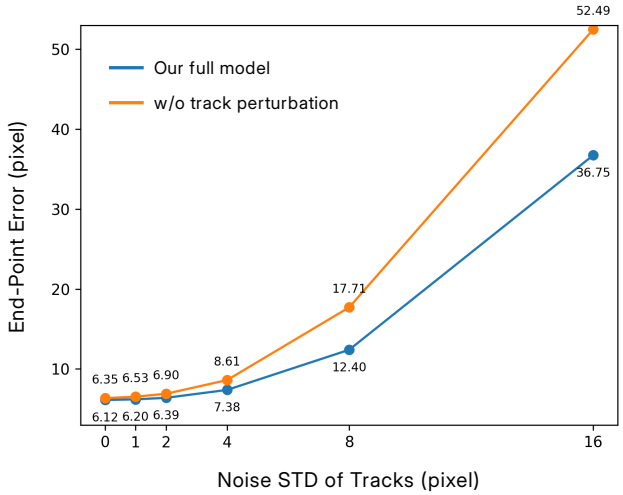

Figure 2. **Robustness to noisy track inputs.** Our final model, trained with track perturbation (Sec. A.4), is compared to an ablated model trained without it. Both are tested with varying levels of Gaussian noise on the target tracks. Our model handles 4-pixel noise with only a 1.26-pixel increase in error, demonstrating its robustness.

from the visibility labels obtained from the 3D track estimation. Our method *does not* use visibility labels for occlusion handling since 3D motion editing can introduce the ambiguity of visibility. Therefore, the model is trained to handle all tracks regardless of their visibility, enabling it to reason about depth order and occlusion for target video edits.

A.6. Video Pose and 3D Point Track Estimation

We leverage the recent advancements in video-depth-pose [8, 12, 20, 21] and 3D tracking methods [12] to estimate the required 3D tracks and camera parameters, including intrinsics and extrinsics, for our framework. While some 3D methods may further apply optimization after feed-forward models to refine the 3D estimates, such as MegaSAM [12] for consistent depth, or SpatialTrackerV2 [22] for 3D tracks, these steps can be time-consuming. To speed up pose and track estimation, we use the fine-tuned VGGT [20] model in SpatialTrackerV2 [22] to estimate video depth and pose, and then employ TAPIP3D [27] to obtain 3D point tracks without optimizations. For a 400-frame training video, we sample 2000 points across 17 uniformly spaced keyframes with the same interval to ensure most video content is tracked. To ensure detailed foreground movements are tracked, we employ a foreground-biased sampling strategy when masks are available. This involves densely sampling points on foreground objects while sparsely sampling the background. The mask generation process is adapted for different phases: during training, we automatically generate masks by applying semantic segmentation to extract likely-dynamic classes; at test time, users interactively provide the foreground masks using SAM2 [15]. This 3D preprocessing takes approximately 4 minutes for a 400-frame video on an A100-80GB GPU.

A.7. Editing and Inference Details

For a given test video, we first run SAM2 [15] to extract foreground object masks before performing video pose and 3D track estimation (Sec. A.6). These masks serve two purposes. First, they allow us to associate point tracks with their corresponding objects (Sec. 3.3, main paper). Second, they are used to densely sample points on the foreground objects during the 3D track estimation, which is crucial for capturing detailed motions, such as those of arms and legs.

For all our demonstrated examples, the editing process exploits a Python script to apply 3D transformations to the 3D tracks and camera poses. For more specific tasks, such as *Shape deformation for body parts* or applying *Partial track inputs* (*e.g.*, removing legs' tracks to avoid detailed leg motion specification), we select a keyframe and use a 2D bounding box in the 2D frame space to select the corresponding subset of 3D points for editing. For basic 3D track edits, we can also optionally generate a preview video by editing the depth-unprojected per-frame point clouds and warping them to the target viewpoints before running the full model. Note that the preview video *will not* be input to our model. The 3D track and viewpoint editing process itself is very fast, typically taking less than 1 second for an 81-frame video without preview warping. The time increases to around 30 to 60 seconds if a preview video is rendered, as that process additionally requires editing the per-frame point clouds. In the future, we plan to develop a 3D GUI editor to make viewpoint and 3D object motion editing more accessible to general users.

Once the poses and tracks are edited, we input a random

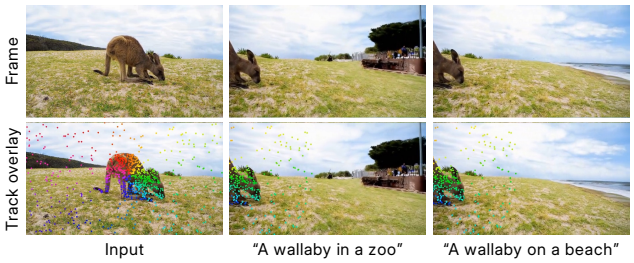

Figure 3. **Effects of text prompts.** The text prompt serves as a supplementary context to help generate unseen regions revealed in the novel viewpoints. As shown in the example, the right-half regions without tracks overlayed are the unseen regions.

Figure 4. **Effects of different random seeds.** Random seeds introduce generation variations, especially in newly revealed areas (red arrows). Please note that all other examples and evaluations utilize the same fixed seed (0) for consistency.

sample of 1,000 tracks to the track-conditioned V2V model. The model uses these tracks and a text prompt describing the target video to edit the video with the specified motion. For consistency, we use a fixed random seed of 0 for all demonstrated examples and evaluations.

B. Model Analysis

B.1. Robustness to Sparse Point Tracks

Our model is trained using a random number of point tracks, ranging from 500 to 1000. We tested the trained model's robustness to track sparsity during inference by measuring the End-Point Error (EPE) on 100 in-the-wild videos from MiraData [10], with results shown in Fig. 1. While the model fails with extremely sparse inputs (*e.g.*, $N = 32$), due to few correspondences, it may still achieve reasonable performance with $\sim$256 tracks. Visual results are available in the webpage.

B.2. Robustness to Noisy Point Tracks

To account for potential noise and inaccuracies in estimated 3D tracks and camera poses, we test the robustness of our model to perturbed track inputs. This experiment involves adding varying amounts of Gaussian noise to the target point tracks. We evaluate the End-Point Error (EPE) on 100 MiraData [10] videos, with results shown in Fig. 2. Our model, trained with track perturbation augmentation (Sec. A.4), is robust to handle approximately 4 pixels of noise while incurring only a 1.26-pixel increase in error. In contrast, an ablated model trained without this augmentation is less robust, and its End-Point Error (EPE) increases more rapidly as noise levels rise. We provide video results in the webpage, demonstrating the performance of our final model under various noise levels.

B.3. Effects of Text Prompts

Our model utilizes 3D tracks for precise motion control, while text prompts provide supplementary context, helping to generate unseen regions from novel viewpoints (Fig. 3) or specific, motion-dependent effects. Please see the videos in our project webpage.

B.4. Effects of Random Seeds

Different random seeds produce slight variations in the generated videos, particularly in areas that are unseen from the input and revealed by the target motion (Fig. 4). Note that for consistency, all our results and evaluations use a fixed seed $= 0$, except for Fig. 4.

B.5. Failure Cases

While our model demonstrates strong capability in joint camera and object motion control, it still exhibits a few limitations (Fig. 5). First, applying large motion changes (*e.g.*, 270° front-flipping) to small objects that are tracked by noisy, densely-clustered points can be challenging. In such cases, the model may struggle to accurately transfer visual context or precisely apply the motion condition, leading to distortion artifacts (Fig. 5a).

Second, although the model can synthesize plausible secondary effects associated with edited objects—such as water splashes (Fig. 2, main paper) or shadows (Fig. 9, main paper), it may fail to handle more complex physical phenomena (*e.g.*, liquid dynamics) that arise from the target motion. For instance, in Fig. 5b, the model fails to synthesize the mixing of coffee and milk when the pouring action is redirected into the other milk bottle.

C. Additional Comparisons

In this section, we present additional quantitative comparisons with existing methods. We also report the details for inference runtime and output resolutions in Table 1. We highly encourage our readers to view our project webpage for full video comparisons of motion editing on in-the-wild videos.

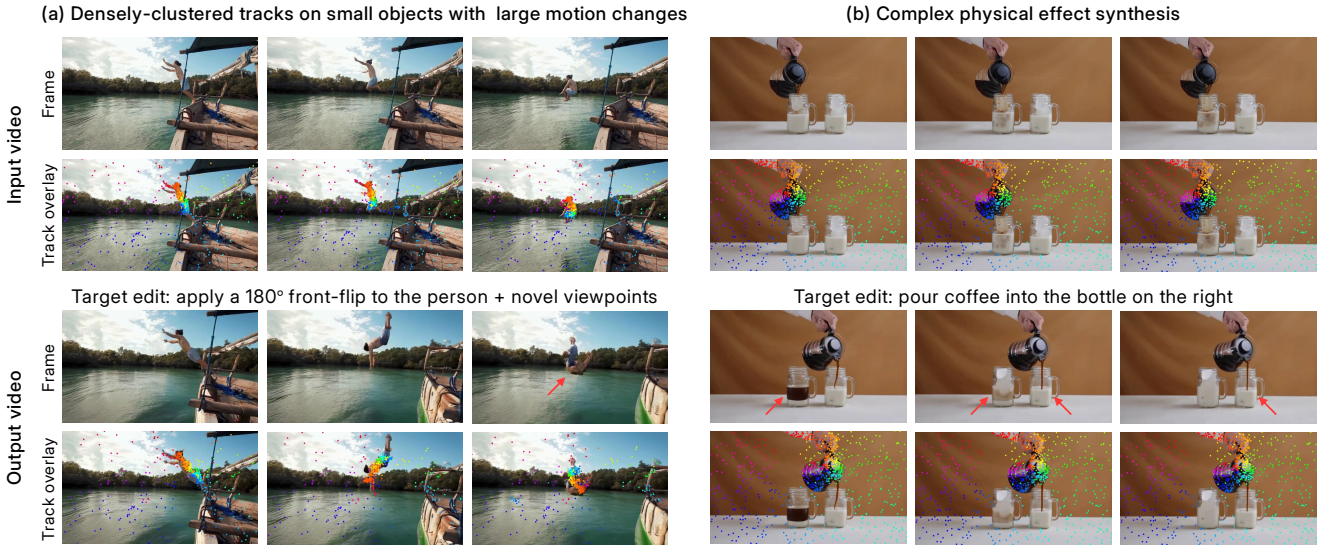

Figure 5. **Limitations.** Our model shows limitations in two main areas: (a) small objects with large motion changes (*e.g.*, a 270° front-flip) can suffer from distortion when their tracks are densely-clustered and noisy; and (b) complex, motion-dependent physical effects (*e.g.*, liquid dynamics) are not synthesized correctly, such as the failure to blend coffee and milk.

Table 1. **Inference runtime and resolution.** The table reports runtime on a single A100-80GB GPU, the number of parameters in the base models, and the inference resolutions for our method and all baselines.

Method	Base model	# params	Temporal length	Spatial resolution $W \times H$	Runtime (min)
ReVideo [14]	SVD [3]	1.5B	14	1344×768	1.7
TrajAttn [23]	SVD [3]	1.5B	25	1024×576	2.0
TrajAttn [23]+ [25]	SVD [3]	1.5B	25	1024×576	6.5
DaS [7]	CogVX [24]	5B	49	720×480	4.4
PaC [4]	SD1.5 [17]	0.9B	16	768×512	1.4
ATI [19]	Wan [18]	14B	81	832×480	14.0
GEN3C [16]	Cosmos [1]	7B	121	1280×704	12.7
TrajCrafter [26]	CogVX [24]	5B	49	672×384	3.2
Ours	Wan [18]	1.3B	81	672×384	4.5

C.1. Evaluation on DyCheck

DyCheck [5] provides synchronized multi-view videos with depth and camera-pose annotations, captured by one moving camera and two stationary cameras. The multi-view videos in a dynamic scene enable us to validate manipulations of camera and object motion, either jointly or independently. We derive three distinct evaluation scenarios:

- **Joint camera and object motion**: evaluated by extracting two non-overlapping clips from each moving-camera video, which naturally contains both motion types (12 scenes).
- **Camera motion only**: evaluated using a synchronized pair, where the moving-camera video serves as input and the corresponding fixed-view video as ground truth (5 scenes).
- **Object motion only**: evaluated by extracting two non-overlapping clips from a fixed-view video (4 scenes).

For the joint-motion and object-motion-only scenarios, we require training pairs consisting of two non-contiguous clips (81 frames each) from the same video. Because the provided multi-view dataset contains videos of varying durations, we select only those with sufficient length to extract such pairs. The resulting clips have an average temporal gap of 99 frames (minimum 17, median 77, maximum 222).

For fair evaluation, we first crop all videos to a landscape orientation. All methods are evaluated at a resolution of 672×384. To accommodate baselines that generate

Table 2. **Quantitative comparison on camera motion only.** We evaluate camera motion only control using the synchronized multi-view video pairs in the DyCheck Dataset [5].

Method	PSNR ↑	SSIM ↑	LPIPS ↓	mPSNR ↑	mSSIM ↑	mLPIPS ↓
GEN3C [16]	13.03	.307	.544	14.64	.723	.382
TrajCrafter [26]	13.41	.304	.551	14.97	.729	.358
Ours	13.75	.302	.481	15.00	.721	.316

Table 3. **Quantitative comparison on object motion only.** We compare our method with track-conditioned I2V methods on the object motion only task (with a static camera) on DyCheck [5]. Note that the baseline I2V methods directly use the ground-truth first frame as input, while our method operates without this privileged input.

Method	PSNR ↑	SSIM ↑	LPIPS ↓	mPSNR ↑	mSSIM ↑	mLPIPS ↓
ReVideo [14]	16.22	.595	.395	18.86	.768	.268
TrajAttn [23]	15.01	.415	.370	17.13	.594	.244
DaS [7]	17.15	.600	.265	20.20	.766	.137
PaC [4]	16.63	.559	.328	19.42	.729	.218
ATI [19]	16.79	.652	.263	20.11	.826	.128
Ours	18.11	.629	.261	21.03	.808	.149

Figure 6. **Visual comparison of joint motion editing on DyCheck [5].** Our method handles large, joint camera and object motion changes, aligning with the (pseudo) ground-truth video. Methods marked with * use the flow estimation between input and ground-truth videos to warp the input. The results of TrajAttn* are generated with the extension of NVS-Solver [25] to take the warped video input.

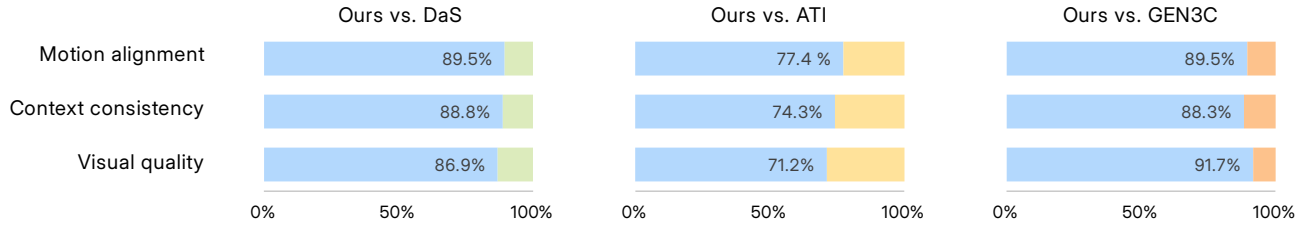

Figure 7. **Human perceptual evaluation.** We present the preference percentages from 42 subjects in comparison with representative methods, including track-conditioned I2V, DaS [7] and ATI [19], and inpainting-based V2V, GEN3C [16].

fewer than 81 frames, we apply a temporal stride (2 or 3) to the input video, ensuring access to the full video content in a single pass. We then compute error metrics only on the corresponding subsampled output frames. Note that ReVideo [14] is an exception because it outputs only 14 frames. For this method, we provide the 81-frame video subsampled with a temporal stride of 3 (yielding a 27-frame input). We then run the model twice on consecutive segments of this input and concatenate the results to obtain a 27-frame output video for evaluation.

For our primary task of joint camera and object motion control, we present the quantitative comparisons in the main paper, and the visual comparison in Fig. 6. Table 2 presents the quantitative comparisons on the camera-motion-only task. Our approach performs comparably to dedicated camera-controlled V2V methods [16, 26], which rely on dense depth warping. Furthermore, in the object-motion-only task (Table 3), while tracked-conditioned methods [4, 7, 14, 19, 23] use the ground-truth first frame, our approach achieves overall better scores due to its effective use of context from the input video.

C.2. User Study on Video Editing

We conduct a human perceptual evaluation with 42 subjects using the Two-Alternative Forced Choice (2AFC) method to assess 10 real-world motion editing cases, including camera and/or object motion editing. Subjects assessed output quality based on three critical aspects: (i) alignment with desired motion, (ii) preservation of input context, and (iii) perceived visual quality. As reported in Fig. 7, our method consistently shows a higher preference over the representative baselines, DaS [7], ATI [19], and GEN3C [16] across all three key aspects of the video motion editing task.

C.3. Comparison with ReCamMaster

ReCamMaster [2] is a camera-controlled video-to-video diffusion model that uses target camera extrinsics to edit viewpoints. While it shares the same Wan2.1-T2V-1.3B base model as our method, its design is fundamentally different. First, ReCamMaster conditions only on the target extrinsics and ignores the source camera parameters of the input video. This design choice relies on a key assumption: the input and target output videos must share the same first frame. In contrast, our method *does not* have this same-first-frame requirement. Our training pairs are constructed from two non-contiguous clips from a monocular video, where the first frames of the clips are different by design. Second, although directly inputting camera extrinsics is straightforward, this approach can suffer from scale ambiguity, which limits accurate camera-motion control. Our

method avoids this issue by representing camera motion using screen-projected point tracks, enabling more flexible and precise control.

References

[1] Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025. 5

[2] Jianhong Bai, Menghan Xia, Xiao Fu, Xintao Wang, Lianrui Mu, Jinwen Cao, Zuozhu Liu, Haoji Hu, Xiang Bai, Pengfei Wan, et al. Recammaster: Camera-controlled generative rendering from a single video. In *ICCV*, 2025. 6

[3] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023. 5

[4] Yingjie Chen, Yifang Men, Yuan Yao, Miaomiao Cui, and Liefeng Bo. Perception-as-control: Fine-grained controllable image animation with 3d-aware motion representation. In *ICCV*, 2025. 5, 6

[5] Hang Gao, Ruilong Li, Shubham Tulsiani, Bryan Russell, and Angjoo Kanazawa. Monocular dynamic view synthesis: A reality check. In *NeurIPS*, 2022. 5, 6

[6] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, et al. Kubric: A scalable dataset generator. In *CVPR*, 2022. 1

[7] Zekai Gu, Rui Yan, Jiahao Lu, Peng Li, Zhiyang Dou, Chenyang Si, Zhen Dong, Qifeng Liu, Cheng Lin, Ziwei Liu, et al. Diffusion as shader: 3d-aware video diffusion for versatile video generation control. In *SIGGRAPH*, 2025. 5, 6

[8] Jiahui Huang, Qunjie Zhou, Hesam Rabeti, Aleksandr Korovko, Huan Ling, Xuanchi Ren, Tianchang Shen, Jun Gao, Dmitry Slepichev, Chen-Hsuan Lin, et al. Vipe: Video pose engine for 3d geometric perception. *arXiv preprint arXiv:2508.10934*, 2025. 3

[9] Adobe Systems Inc. Mixamo. https://www.mixamo.com/. Accessed: 2025-10-28. 1

[10] Xuan Ju, Yiming Gao, Zhaoyang Zhang, Ziyang Yuan, Xintao Wang, Ailing Zeng, Yu Xiong, Qiang Xu, and Ying Shan. Miradata: A large-scale video dataset with long durations and structured captions. In *NeurIPS*, 2024. 3, 4

[11] Yao-Chih Lee, Erika Lu, Sarah Rumbley, Michal Geyer, Jia-Bin Huang, Tali Dekel, and Forrester Cole. Generative omnimatte: Learning to decompose video into layers. In *CVPR*, 2025. 2

[12] Zhengqi Li, Richard Tucker, Forrester Cole, Qianqian Wang, Linyi Jin, Vickie Ye, Angjoo Kanazawa, Aleksander Holynski, and Noah Snavely. Megasam: Accurate, fast and robust structure and motion from casual dynamic videos. In *CVPR*, 2025. 3

[13] Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, et al. Dl3dv-10k: A large-scale scene dataset for deep learning-based 3d vision. In *CVPR*, 2024. 2

[14] Chong Mou, Mingdeng Cao, Xintao Wang, Zhaoyang Zhang, Ying Shan, and Jian Zhang. Revideo: Remake a video with motion and content control. In *NeurIPS*, 2024. 5, 6

[15] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024. 3

[16] Xuanchi Ren, Tianchang Shen, Jiahui Huang, Huan Ling, Yifan Lu, Merlin Nimier-David, Thomas Müller, Alexander Keller, Sanja Fidler, and Jun Gao. Gen3c: 3d-informed world-consistent video generation with precise camera control. In *CVPR*, 2025. 5, 6

[17] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 5

[18] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025. 5

[19] Angtian Wang, Haibin Huang, Jacob Zhiyuan Fang, Yiding Yang, and Chongyang Ma. Ati: Any trajectory instruction for controllable video generation. *arXiv preprint arXiv:2505.22944*, 2025. 5, 6

[20] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *CVPR*, 2025. 3

[21] Yifan Wang, Jianjun Zhou, Haoyi Zhu, Wenzheng Chang, Yang Zhou, Zizun Li, Junyi Chen, Jiangmiao Pang, Chunhua Shen, and Tong He. Pi3: Permutation-equivariant visual geometry learning. *arXiv preprint arXiv:2507.13347*, 2025. 3

[22] Yuxi Xiao, Jianyuan Wang, Nan Xue, Nikita Karaev, Yuri Makarov, Bingyi Kang, Xing Zhu, Hujun Bao, Yujun Shen, and Xiaowei Zhou. Spatialtrackerv2: 3d point tracking made easy. In *ICCV*, 2025. 3

[23] Zeqi Xiao, Wenqi Ouyang, Yifan Zhou, Shuai Yang, Lei Yang, Jianlou Si, and Xingang Pan. Trajectory attention for fine-grained video motion control. In *ICLR*, 2025. 5, 6

[24] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer. *arXiv preprint arXiv:2408.06072*, 2024. 5

[25] Meng You, Zhiyu Zhu, Hui Liu, and Junhui Hou. Nvs-solver: Video diffusion model as zero-shot novel view synthesizer. In *ICLR*, 2024. 5, 6

[26] Mark YU, Wenbo Hu, Jinbo Xing, and Ying Shan. Trajectorycrafter: Redirecting camera trajectory for monocular videos via diffusion models. In *ICCV*, 2025. 5, 6

[27] Bowei Zhang, Lei Ke, Adam W Harley, and Katerina Fragkiadaki. Tapip3d: Tracking any point in persistent 3d geometry. In *NeurIPS*, 2025. 3

[28] Yang Zheng, Adam W Harley, Bokui Shen, Gordon Wetzstein, and Leonidas J Guibas. Pointodyssey: A large-scale synthetic dataset for long-term point tracking. In *ICCV*, 2023. 1