FloVD: Optical Flow Meets Video Diffusion Model for Enhanced Camera-Controlled Video Synthesis

Supplementary Material

In this supplemental document, we provide:

- Additional implementation details of FloVD,
- Experimental details of baseline methods,
- Additional evaluation details,
- Additional quantitative ablation, and
- Additional qualitative comparison.

1. Additional Implementation Details of FloVD

1.1. Network Architecture

The VAE encoders and decoders, and denoising U-Nets of the object motion synthesis model (OMSM) and of the flow-conditioned video synthesis model (FVSM) adopt the network architectures of Stable Video Diffusion [2]. For the flow encoder of FVSM, we adopt the CNN encoder from CameraCtrl [4], modifying it to process two-channel optical flow maps instead of six-channel Plücker embeddings. The flow encoder produces multi-level flow embeddings $\{\zeta_{(t,l)}\}_{t=1,l=1}^{T,L}$ for the t-th frame at level l, where $T = 14$ and $L = 4$. To incorporate the optical flow condition into the denoising U-Net, the multi-level flow embeddings are added to the intermediate feature maps within the U-Net's encoder. Each intermediate feature map matches the resolution and channel size of the corresponding flow embedding at the encoder's depth level l.

1.2. Experimental Details

For training OMSM and FVSM, we use learning rate of 0.00003 with the AdamW optimizer [8]. FVSM is trained over approximately two days using 16 A100 GPUs. OMSM is trained on the entire dataset for about three days using 8 A100 GPUs, followed by fine-tuning on the curated dataset for an additional 1.5 days. As explained in the main paper, we apply adaptive normalization for optical flow maps, following Li et al. [6]. Specifically, we use different scale factors for the normalization of x- and y-directional optical flow vectors. The scale factors of (18, 12) and (45, 24) are used to normalize flow-map data for OMSM and FVSM, respectively.

Following Stable Video Diffusion [2], we adopt the EDM framework [5] for both training and inference, and apply linearly increasing classifier-free guidance during inference. For training FVSM, we modify only the timestep sampling strategy of the EDM framework. Inspired by T2I-Adapter [9], we introduce a quadratic timestep sampling strategy to enable FVSM to more effectively leverage the input flow condition for structural content generation. Specifically, FVSM is trained primarily on highly noised data with large timesteps.

To achieve this, timesteps are uniformly sampled within the range [0, 1], squared, and subtracted from 1. The resulting values are then scaled to match the range of (-3.66, 3.66), which roughly aligns with the timestep range used in the EDM framework. This approach enables the denoising U-Net in FVSM to better learn structural content generation by leveraging the flow map condition, thereby enhancing its capability for effective camera control.

1.3. Off-the-shelf Models Used in FloVD

We employ several off-the-shelf models in our framework: a single-image 3D estimation network [13], an optical flow estimation network [11], and a segmentation network [10] for moving object detection.

Single-image 3D estimation network. We use Depth Anything V2 [13] for the single-image 3D estimation network. Specifically, we use its fine-tuned version for metric depth estimation, which has ViT-base encoder and is trained using the Hypersim dataset.

Optical flow estimation network. We use RAFT [11] for the optical flow estimation network. Network outputs are iteratively updated for 20 times to obtain the final optical flow map.

Moving object segmentation network. In the flow integration stage, we use a binary mask for moving object. To obtain the mask, we use an open-set segmentation method, Grounded-SAM 2, which integrates an open-set object detection model [7] and a foundation segmentation model [10]. This method takes a text prompt and predict masks indicating subjects related to the input text prompt. To obtain masks for moving objects, we use "moving object." as the input text prompt. We do not use the obtained mask if the number of pixels in the mask was more than 50% of the total image pixels.

2. Experimental Details of Baseline Methods

We compare our method against baseline methods for camera-controllable video synthesis [3, 4, 12, 14]. Among these, MotionCtrl [12] and CameraCtrl [4] support detailed camera control by utilizing camera parameters as input, whereas AnimateDiff [3] and Direct-a-Video [14] support

	Training Data	Timestep Strategy	Pexels-small (<20)			Pexels-med. (<40)			Pexels-large (≥ 40)		
			FVD (↓)	FID (↓)	IS (↑)	FVD (↓)	FID (↓)	IS (↑)	FVD (↓)	FID (↓)	IS (↑)
Baseline	RE10K	QTS	241.61	**22.01**	11.09	334.06	22.30	11.17	363.04	23.17	12.05
+ OMSM	RE10K	QTS	231.35	22.43	11.44	206.38	**20.53**	11.62	229.05	**20.95**	12.65
+ large-scale data	Internal	QTS	**220.65**	22.49	**11.58**	**183.14**	20.71	**11.68**	**207.39**	21.12	**12.95**
Baseline	RE10K	EDM	238.85	22.16	11.07	309.28	22.05	11.28	335.02	22.91	12.15
+ OMSM	RE10K	EDM	217.24	22.13	11.44	186.21	20.12	**11.81**	201.27	20.45	12.71
+ large-scale data	Internal	EDM	**212.03**	**21.79**	**11.62**	**165.78**	**19.73**	11.684	**177.45**	**20.02**	**12.88**

Table S1. Additional quantitative ablation study of our main components.

basic camera control operations, such as translation and zoom. For all the baseline methods, we used the official checkpoints and inference code provided in their respective GitHub repositories.

MotionCtrl We use the official PyTorch implementation of MotionCtrl [12]. To ensure a fair comparison with our method, which is based on Stable Video Diffusion [2], we utilize the official variant of MotionCtrl that employs Stable Video Diffusion (https://github.com/TencentARC/MotionCtrl).

CameraCtrl We use the official PyTorch implementation of CameraCtrl [4]. To ensure a fair comparison with our method, which is based on Stable Video Diffusion [2], we utilize the official variant of CameraCtrl that employs Stable Video Diffusion (https://github.com/hehao13/CameraCtrl).

AnimateDiff We use the official PyTorch implementation of AnimateDiff [3]. The official codes can be found in (https://github.com/guoyww/AnimateDiff). To control predefined camera trajectories, such as zoom and pan, AnimateDiff provides fine-tuned models tailored for each camera trajectory. Thus, we utilize these fine-tuned models for camera control during video generation. Additionally, we use the model from Realistic Vision as a backbone for AnimateDiff, as it is most closely aligned with generating natural images.

Direct-a-Video We use the official PyTorch implementation of Direct-a-Video [14]. The official codes can be found in (https://github.com/ysy31415/direct_a_video). Direct-a-Video controls basic camera motions using camera parameters such as x-pan ratio, y-pan ratio, and zoom ratio. For our experiments, we set the pan ratio to 0.3 and use scales of 0.8 and 1.2 for zoom-in and zoom-out ratios, respectively.

3. Additional Details for Evaluation Protocol

3.1. Camera Controllability

To obtain each video clip in the evaluation dataset, we first select a middle frame within the whole video frames as the first frame, and then choose 13 additional frames at intervals of four frames starting from the first frame, resulting in a total of 14 frames. Then, we obtain camera parameters corresponding to these selected frames to serve as the ground-truth camera parameters for the evaluation set. For the camera parameters estimated from synthesized videos, we normalize the translation vectors using a scale factor to account for scene-specific scale variations, following CamI2V [15]. Specifically, the translation vectors are divided by a scene-specific scale factor. This scale factor is determined based on the $\mathcal{L}_2$ distance between the locations of the first camera and farthest camera in the scene.

Figure S1. Visual examples of each benchmark dataset, categorized according to the degree of object motion.

3.2. Video Synthesis Quality

As explained in the main paper, we provide three benchmark datasets of real-world videos categorized by small, medium, and large object motions to evaluate the object motion synthesis quality. To obtain videos with minimal camera motions, we first obtain optical flow maps from the Pexels dataset [1], and then filter out videos whose average magnitude of the optical flow vectors of the background is larger than 1.0. Fig. S1 shows several visual examples from each benchmark dataset. The first set primarily features landscapes or objects with minimal motions, while the third set typically consists of objects with notable motions.

4. Additional Quantitative Ablation Study

Tab. S1 reports additional evaluation of our main components using the three benchmark datasets for object motion synthesis quality. As shown in Tab.3 of the main paper, introducing each component usually enhances evalua-

tion metrics, in both cases using the quadratic timestep sampling strategy (QTS) or EDM framework [5].

5. Additional Qualitative Comparison

We additionally provide visual comparison against previous methods. First, we compare our method with AnimateDiff [3] and Direct-a-Video [14] using basic camera trajectories such as zoom and translation, as these methods are limited to those basic camera movements. Next, we compare our method with MotionCtrl [12] and CameraCtrl [4], which support more detailed camera control during video generation.

5.1. Using Basic Camera Trajectory

Fig. S2 presents a visual comparison using basic camera trajectories such as zoom-in and translation to left and right. Unlike our method, which uses a single image as input, Direct-a-Video [14] and AnimateDiff [3] require text prompts as input. Thus, we use text prompts of videos from the Pexels dataset [1] as input for these methods, while the first frame of the same video serves as input for our method. As shown in Fig. S2, our method synthesizes high-quality video frames with accurate camera control, while previous methods produce video frames with quality degradation.

5.2. Using Detailed Camera Trajectory

Fig. S3 and Fig. S4 provide a visual comparison of synthesized video frames with detailed camera trajectory. MotionCtrl [12] often fails to accurately follow the input camera trajectory, whereas both CameraCtrl [4] and our method demonstrate accurate camera control performance.

Fig. S5 provides additional visual comparison across all the methods. Our method generates realistic object motion in the synthesized video frames, while previous methods often produce unnatural videos with artifacts. In particular, CameraCtrl [4] often synthesizes video frames without object motion, and MotionCtrl [12] often produces inconsistent foreground and background, as shown in Fig. S5. Additional visual examples can be found in Fig. S6.

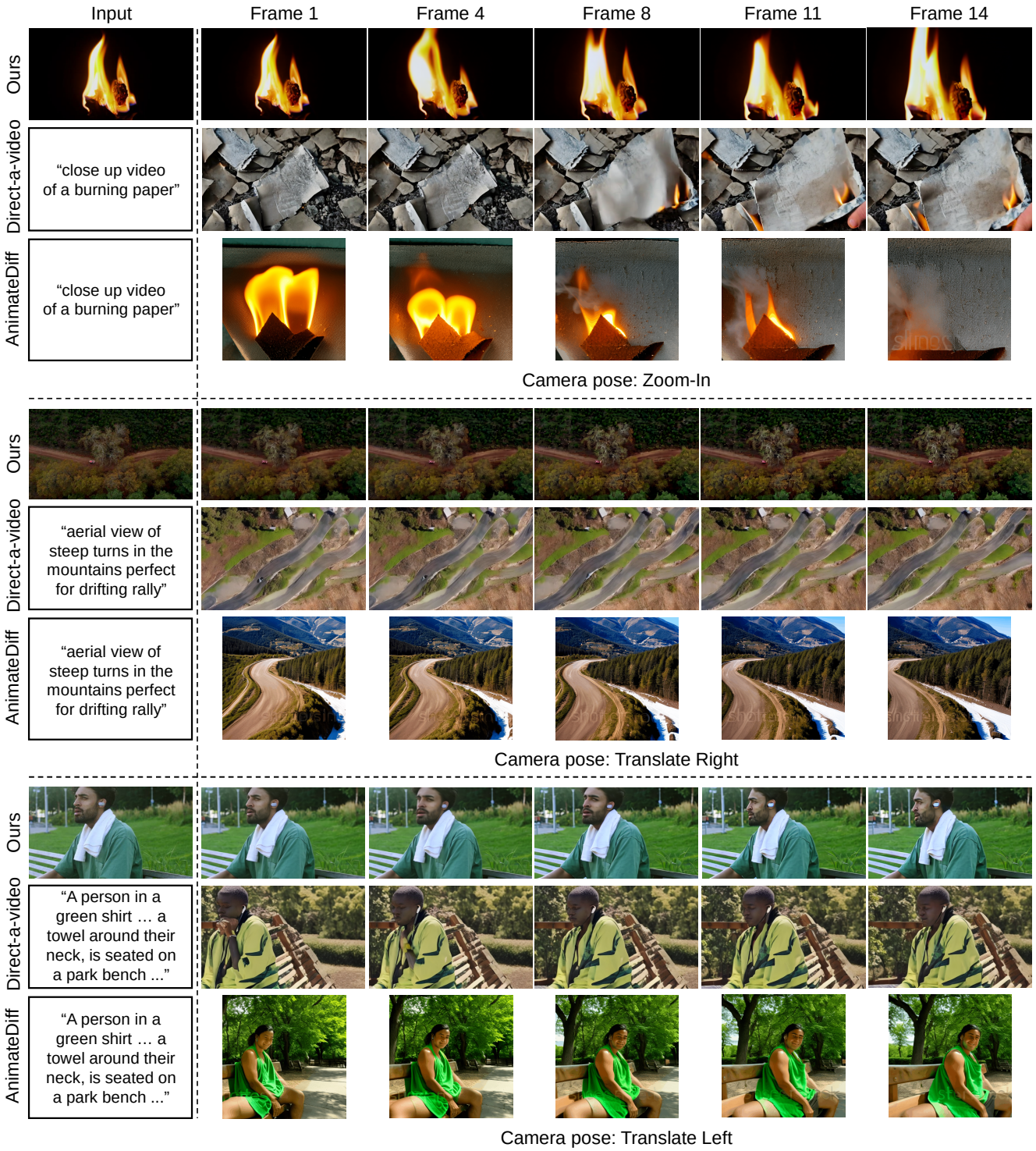

Figure S2. Additional qualitative comparison of our method against Direct-a-video [14] and AnimateDiff [3] using basic camera trajectories.

Figure S3. Additional qualitative comparison of our method against MotionCtrl [12] and CameraCtrl [4] using detailed camera trajectories. MotionCtrl often fails to follow the input camera parameters during video generation.

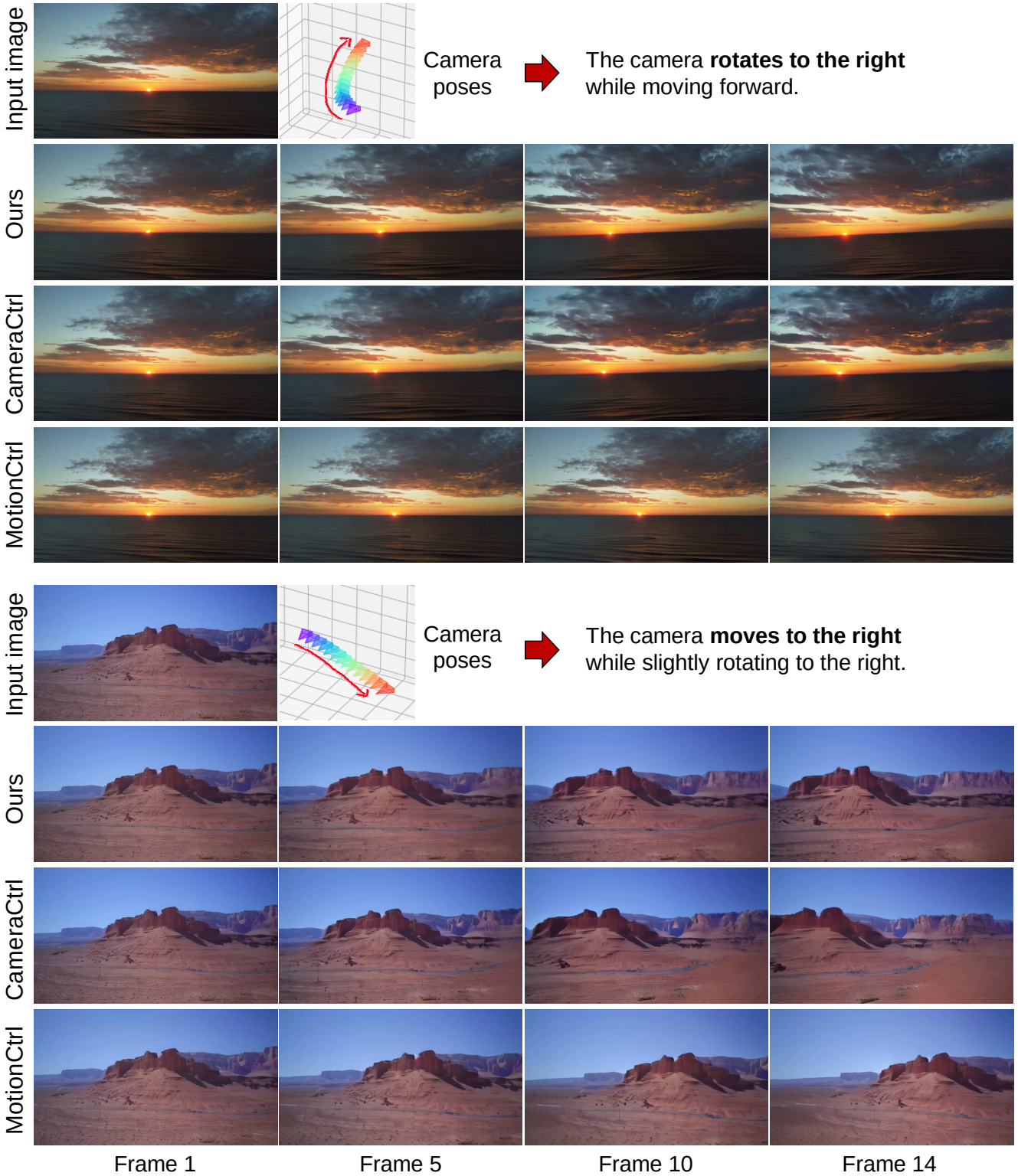

Figure S4. Additional qualitative comparison of our method against MotionCtrl [12] and CameraCtrl [4] using detailed camera trajectories. MotionCtrl often fails to follow the input camera parameters during video generation.

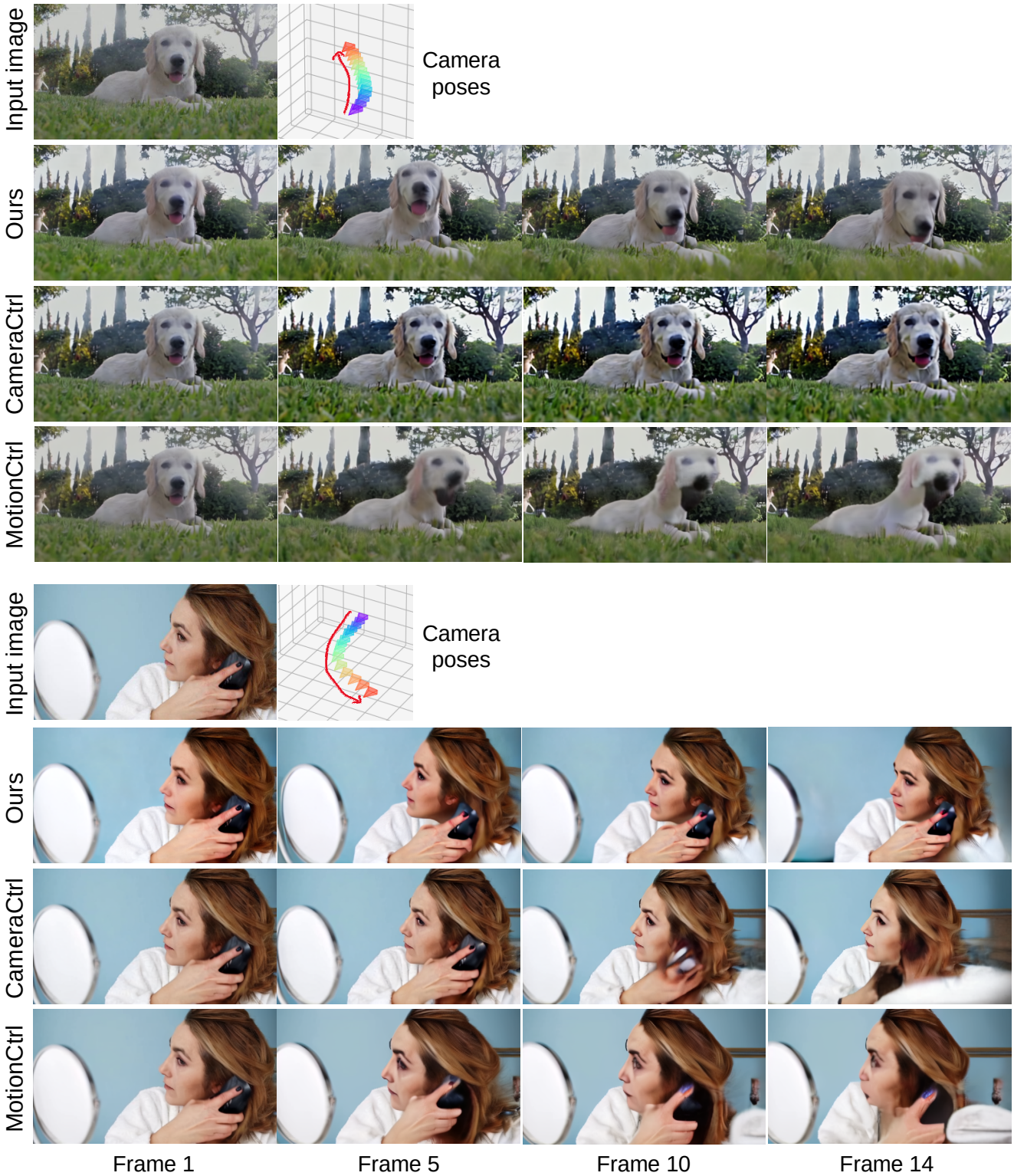

Figure S5. Additional qualitative comparison of our method against MotionCtrl [12] and CameraCtrl [4] using detailed camera trajectories. Our method produces more natural object motion, while CameraCtrl produces a foreground object without motions, and MotionCtrl produces artifacts.

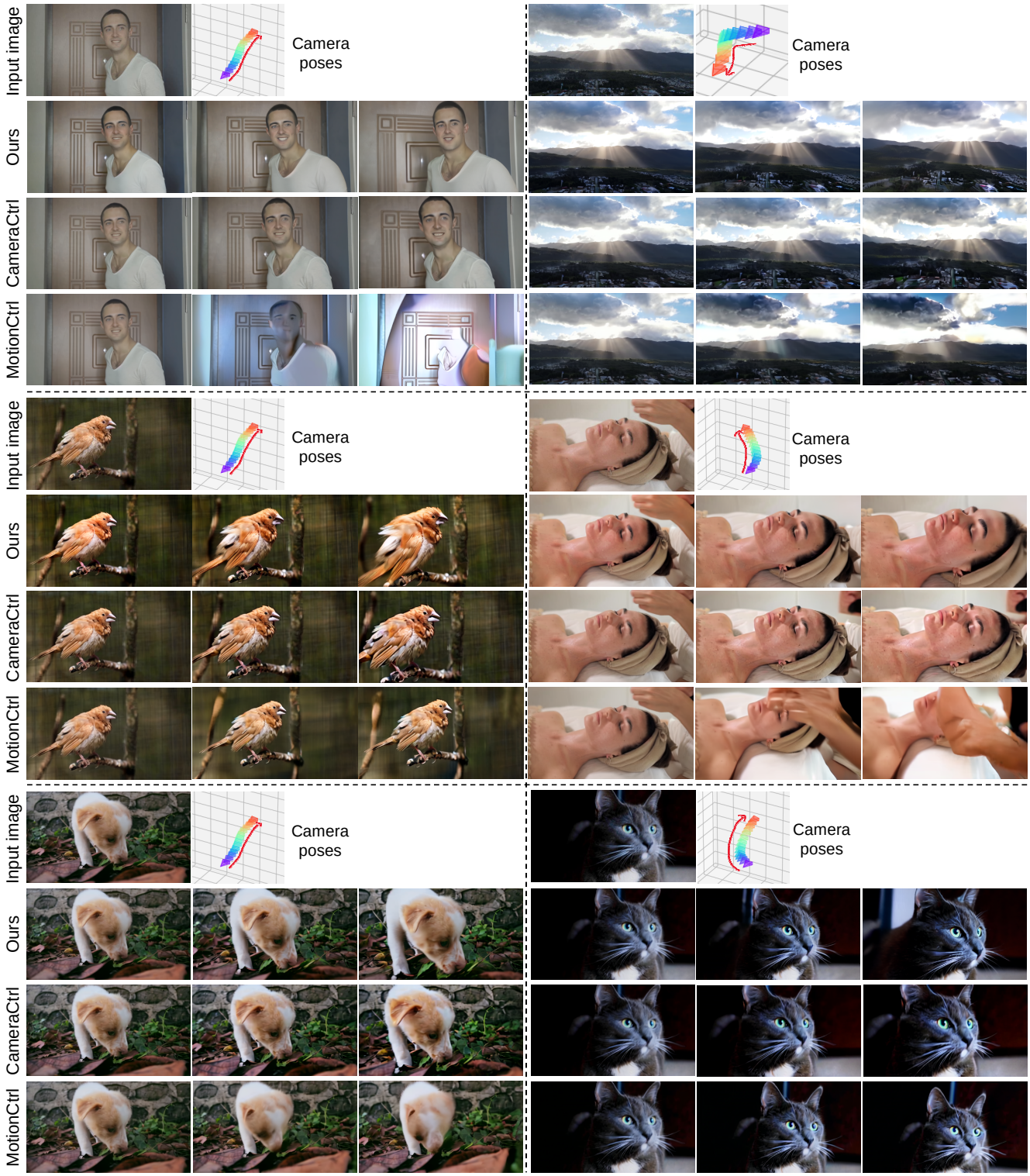

Figure S6. Additional qualitative comparison of our method against MotionCtrl [12] and CameraCtrl [4] using detailed camera trajectories.

References

[1] Pexels, royalty-free stock footage website. https://www.pexels.com. Accessed: 2024-09-30. 2, 3

[2] 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. 1, 2

[3] Yuwei Guo, Ceyuan Yang, Anyi Rao, Zhengyang Liang, Yaohui Wang, Yu Qiao, Maneesh Agrawala, Dahua Lin, and Bo Dai. Animatediff: Animate your personalized text-to-image diffusion models without specific tuning. *arXiv preprint arXiv:2307.04725*, 2023. 1, 2, 3, 4

[4] Hao He, Yinghao Xu, Yuwei Guo, Gordon Wetzstein, Bo Dai, Hongsheng Li, and Ceyuan Yang. Cameractrl: Enabling camera control for text-to-video generation. *arXiv preprint arXiv:2404.02101*, 2024. 1, 2, 3, 5, 6, 7, 8

[5] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022. 1, 3

[6] Zhengqi Li, Richard Tucker, Noah Snavely, and Aleksander Holynski. Generative image dynamics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24142–24153, 2024. 1

[7] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023. 1

[8] I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 1

[9] Chong Mou, Xintao Wang, Liangbin Xie, Yanze Wu, Jian Zhang, Zhongang Qi, and Ying Shan. T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 4296–4304, 2024. 1

[10] 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. 1

[11] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 402–419. Springer, 2020. 1

[12] Zhouxia Wang, Ziyang Yuan, Xintao Wang, Yaowei Li, Tianshui Chen, Menghan Xia, Ping Luo, and Ying Shan. Motionctrl: A unified and flexible motion controller for video generation. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024. 1, 2, 3, 5, 6, 7, 8

[13] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *arXiv preprint arXiv:2406.09414*, 2024. 1

[14] Shiyuan Yang, Liang Hou, Haibin Huang, Chongyang Ma, Pengfei Wan, Di Zhang, Xiaodong Chen, and Jing Liao. Direct-a-video: Customized video generation with user-directed camera movement and object motion. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–12, 2024. 1, 2, 3, 4

[15] Guangcong Zheng, Teng Li, Rui Jiang, Yehao Lu, Tao Wu, and Xi Li. Cami2v: Camera-controlled image-to-video diffusion model. *arXiv preprint arXiv:2410.15957*, 2024. 2