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 analysis,
- Additional comparison, and
- Additional visual results.

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 a 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 only modified the timestep sampling strategy of the EDM framework. Inspired by T2I-Adapter [10], 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 [16], an optical flow estimation network [13], and a segmentation network [12] for moving object detection.

Single-image 3D estimation network. We use Depth Anything V2 [16] for the single-image 3D estimation network. Specifically, we use its fine-tuned version for metric depth estimation, which has a ViT-base encoder and is trained using the Hypersim dataset.

Optical flow estimation network. We use RAFT [13] for the optical flow estimation network. Network outputs are iteratively updated 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 objects. 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 [12]. This method takes a text prompt and predicts 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 is more than 50% of the total image pixels.

Temporally-consistent video editing. For temporally-consistent video editing using FloVD, we employ an off-the-shelf optical flow estimator and image editing tool. We use RAFT [13] and InfEdit [15] for the optical flow estimation and first frame editing, respectively.

	Training Data	Timestep Strategy	Pexels-random FVD (↓)	FID (↓)	IS (↑)	Pexels-small (< 20) FVD (↓)	FID (↓)	IS (↑)	Pexels-med. (< 40) FVD (↓)	FID (↓)	IS (↑)	Pexels-large (≥ 40) FVD (↓)	FID (↓)	IS (↑)
Baseline	RE10K	QTS	157.99	39.61	11.19	241.61	**22.01**	11.09	334.06	22.30	11.17	363.04	23.17	12.05
+ OMSM	RE10K	QTS	104.92	36.33	11.51	231.35	22.43	11.44	206.38	**20.53**	11.62	229.05	**20.95**	12.65
+ large-scale data	Internal	QTS	**91.55**	**35.66**	**11.63**	**220.65**	22.49	**11.58**	**183.14**	20.71	**11.68**	**207.39**	21.12	**12.95**
Baseline	RE10K	EDM	148.42	39.92	11.23	238.85	22.16	11.07	309.28	22.05	11.28	335.02	22.91	12.15
+ OMSM	RE10K	EDM	95.31	36.90	11.43	217.24	22.13	11.44	186.21	20.12	**11.81**	201.27	20.45	12.71
+ large-scale data	Internal	EDM	**80.74**	**35.65**	**11.73**	**212.03**	**21.79**	**11.62**	**165.78**	**19.73**	11.684	**177.45**	**20.02**	**12.88**

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

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

2. Experimental Details of Baseline Methods

We compare our method against baseline methods for camera-controllable video synthesis [3, 4, 14, 17]. Among these, MotionCtrl [14] and CameraCtrl [4] support detailed camera control by utilizing camera parameters as input, whereas AnimateDiff [3] and Direct-a-Video [17] support 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 [14]. 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.

	mRotErr (°)	mTransErr	mCamMC
Ours w/ EDM	1.70	0.0983	0.1010
Ours w/ QTS	1.43	0.0869	0.0887

Table S2. Analysis of employing different timestep sampling strategies for camera controllability.

Direct-a-Video We use the official PyTorch implementation of Direct-a-Video [17]. 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 used 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 [18]. 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 the farthest camera in the scene.

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

Figure S2. A visual example of the limitation of FloVD.

# of Data	70K	300K	500K (Original)
FVD (↓)	191.73	188.22	177.45

Table S3. Analysis of the scaling effect using the Pexels-large benchmark dataset and the FVD scores.

	mRotErr (°)	mTransErr	mCamMC
Ours w/ CamParams	1.76	0.1016	0.1039
Ours w/ Flow (final)	1.43	0.0869	0.0887

Table S4. Analysis of training FVSM using camera parameters.

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 Analysis

In the following, we provide additional analysis of our method in terms of comprehensive ablation study, timestep sampling strategy, scaling effect, and training FVSM using camera parameters.

4.1. Additional Quantitative Ablation Study

Tab. S1 reports the comprehensive evaluation of our main components using the whole benchmark datasets for video synthesis quality. Introducing each component mostly enhances evaluation metrics, in both cases using the quadratic timestep sampling strategy (QTS) or EDM framework [5].

4.2. Timestep Sampling Strategy

As stated in Sec. 5.1 of the main paper, we adopt the quadratic timestep sampling strategy (QTS) for better camera controllability, instead of the timestep sampling strategy of the EDM framework [5] used in Stable Video Diffusion [2]. Our model using QTS shows slightly compromised video synthesis quality compared to our model using EDM (Tab. S1). Nevertheless, our model with QTS still demonstrates better performance than the previous methods (Tab. 2 in the main paper). Moreover, QTS enhances the camera controllability of FVSM (Tab. S2). This improved camera controllability results from increased chances of training with highly noised data, allowing our model to effectively leverage flow conditions for structural content generation.

4.3. Scaling effect

By utilizing optical flow during training instead of camera parameters, our method can leverage arbitrary training datasets, allowing for the use of an arbitrary number of training datasets. Thus, we investigate whether increasing the size of the training dataset could enhance the performance of our approach.

To assess this, we constructed two subsets randomly sampled from our full internal dataset, containing 70K and 300K pairs of video frames with corresponding optical flow maps, respectively. Notably, the RealEstate10K dataset [19] includes around 70K videos, fewer compared to the 500K pairs available in our full internal dataset. We trained two variant models of FVSM using these subsets. As illustrated in Tab. S3, the performance of our method on the Pexels-large benchmark dataset consistently improves as the training dataset size increases, indicating the potential for further performance gains with even larger datasets. Additionally, we anticipate that adopting transformer-based architectures could further enhance performance with larger datasets.

4.4. Training FVSM using Camera Parameters

Our proposed training strategy employs optical flow maps rather than camera parameters to eliminate the dataset restriction. Although we may use camera parameters for training when available in the dataset, we found that this does not improve the performance. To show this, we train FVSM using camera flow maps derived from estimated depth and camera parameters, rather than optical flow estimated from the video dataset. As shown in Tab. S4, the results show degraded performance compared to the original one. This performance degradation came from depth estimation errors, as we utilize the off-the-shelf single-image 3D estimation network [16].

4.5. Visual Example on Limitation

Despite using FVSM trained on the large-scale dataset, errors derived from both the object motion synthesis model and the moving object segmentation model can produce unnatural object motions. Fig. S2 shows a visual example on the limitation, particularly in the ring pull of the can (red arrow).

Figure S3. Limited scope of circular camera trajectory.

	Pexels-small	Pexels-med.	Pexels-large
MotionCtrl	0.884	0.881	0.875
CameraCtrl	0.913	0.913	0.910
Ours	**0.917**	**0.916**	**0.911**

Table S5. Quantitative comparison on visual consistency using CLIP-Similarity.

5. Additional Comparison

In the following, we provide additional comparison against previous methods [4, 14] in terms of visual consistency, 3D scene consistency, and challenging camera motion.

5.1. Visual Consistency

To evaluate image-to-video fidelity, we measure CLIP-Similarity [11] between synthesized videos and an input image. Tab. S5 shows quantitative comparison results. MotionCtrl [14] less effectively preserves visual consistency with input image due to visual artifacts, achieving the lowest scores, while ours achieves slightly higher scores than CameraCtrl [4].

5.2. 3D Scene Consistency

Thanks to our 3D-based approach for camera-controllable video generation, FloVD can also provide 3D consistent video synthesis results. To validate this, we evaluate 3D scene consistency by employing a NeRF-based approach [9]. Specifically, we first generate video frames with a forward-facing, circular camera trajectory. We then split these synthesized frames into training and test sets, and reconstruct NeRF using the training set. Lastly, we compute PSNR by comparing the test-set frames against the corresponding images rendered from NeRF.

Fig. S4 shows rendered frames at test-set viewpoints from the reconstructed NeRF. FloVD shows high-quality rendered images compared to those of the previous methods [4, 14], demonstrating more 3D-consistent video synthesis results of our method. Tab. S6 shows quantitative comparison on 3D scene consistency using the PSNR scores. These results also demonstrate that our method achieves comparable or superior 3D scene consistency compared to others.

5.3. Challenging Camera Motion

We provide a qualitative comparison of video synthesis capability for extreme camera motions. Extreme camera poses (e.g., circular camera rotations) are still challenging in camera-controlled video synthesis, and all existing methods struggle with this issue. Nonetheless, our approach demonstrates higher robustness to extreme camera poses than others, due to our 3D-structure-based camera flow map. Fig. S3 shows that MotionCtrl [14] completely fails to follow the input camera parameters. In contrast, both CameraCtrl [4] and our method generate results that reflect input camera parameters up to 72°. While both methods generate distorted images at 72°, our results exhibit fewer distortions.

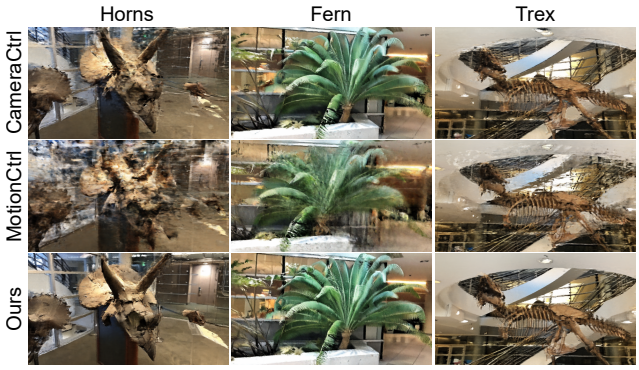

Figure S4. Qualitative comparison on 3D scene consistency using NeRF [9] reconstruction.

	Horns	Fern	Trex
MotionCtrl	15.56	17.87	18.12
CameraCtrl	19.84	**21.42**	20.71
Ours	**21.55**	20.93	**20.02**

Table S6. Quantitative comparison on 3D scene consistency using the PSNR scores.

6. Additional Visual Results

In this section, we provide additional visual results. First, we qualitatively compare our method with AnimateDiff [3] and Direct-a-Video [17] using basic camera trajectories such as zoom and translation, as these methods are limited to those basic camera movements. Next, we qualitatively compare our method with MotionCtrl [14] and CameraCtrl [4], which support more detailed camera control during video generation. Lastly, we provide additional visual examples of the applications regarding temporally-consistent video editing and cinematic camera control.

Figure S5. Additional visual example of the dolly-zoom effect.

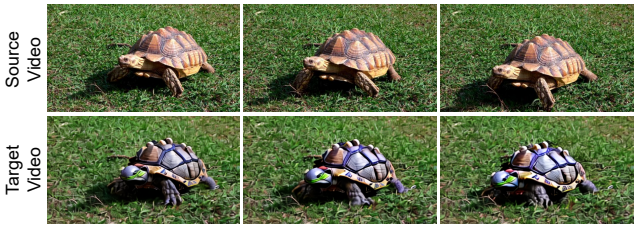

Figure S6. Additional visual example of the temporally-consistent video editing.

6.1. Using Basic Camera Trajectory

Fig. S8 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 [17] 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. S8, our method synthesizes high-quality video frames with accurate camera control, while previous methods produce video frames with quality degradation.

6.2. Using Detailed Camera Trajectory

Fig. S7 and Fig. S9 provide a visual comparison of synthesized video frames with detailed camera trajectory. MotionCtrl [14] often fails to accurately follow the input camera trajectory, whereas both CameraCtrl [4] and our method demonstrate accurate camera control performance.

Fig. S10 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 [14] often produces inconsistent foreground and background, as shown in Fig. S10. Additional visual examples can be found in Fig. S11.

6.3. Application

In Sec. 5.5 of the main paper, we provide two applications using our framework: temporally-consistent video editing and cinematic camera control. We provide additional visual examples of these applications. Fig. S5 and Fig. S5 show additional examples of the dolly-zoom effect and temporally-consistent video editing result, respectively.

Figure S7. Additional qualitative comparison of our method against MotionCtrl [14] and CameraCtrl [4] using detailed camera trajectories. MotionCtrl often fails to follow the input camera parameters during video generation.

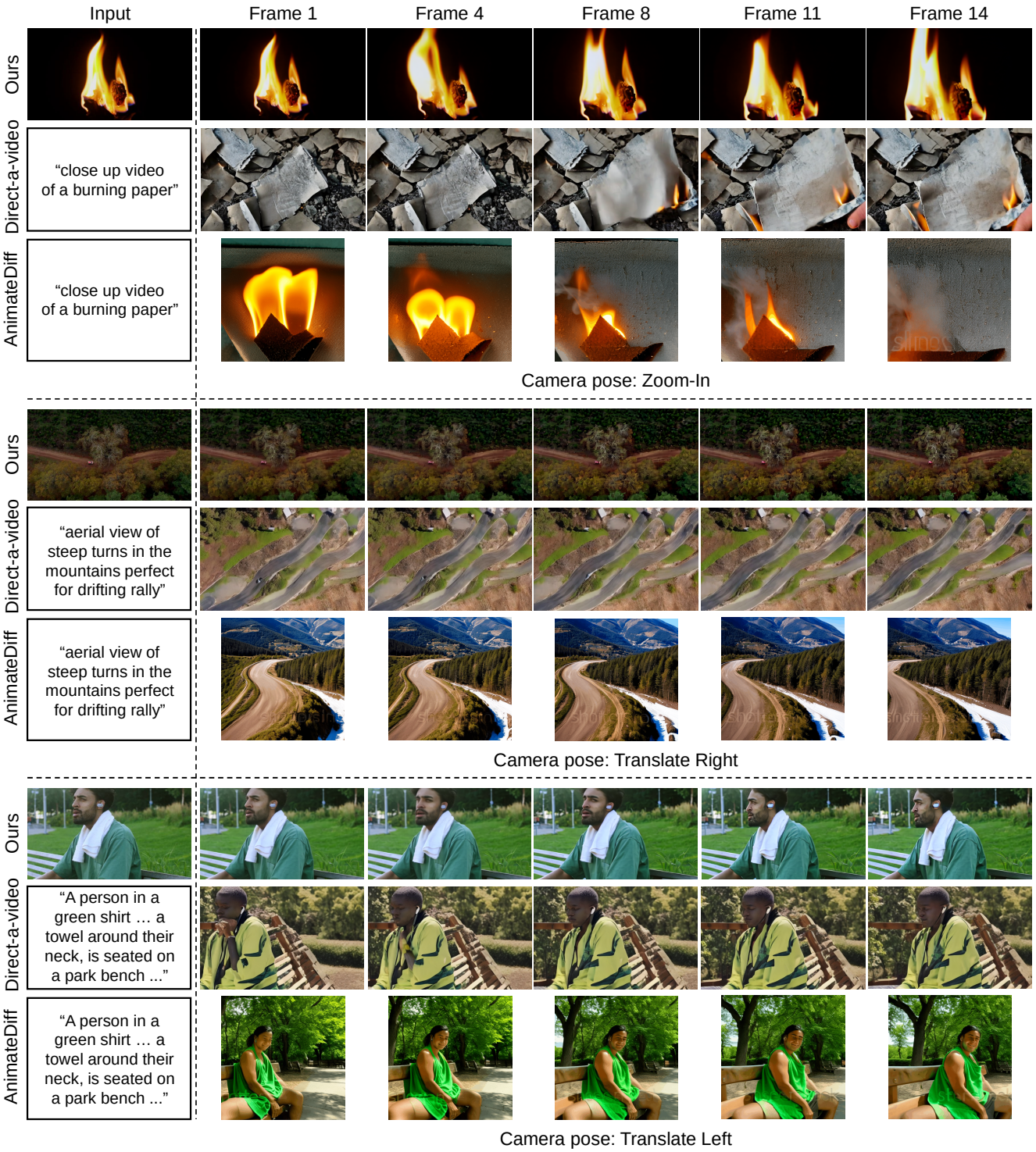

Figure S8. Additional qualitative comparison of our method against Direct-a-video [17] and AnimateDiff [3] using basic camera trajectories.

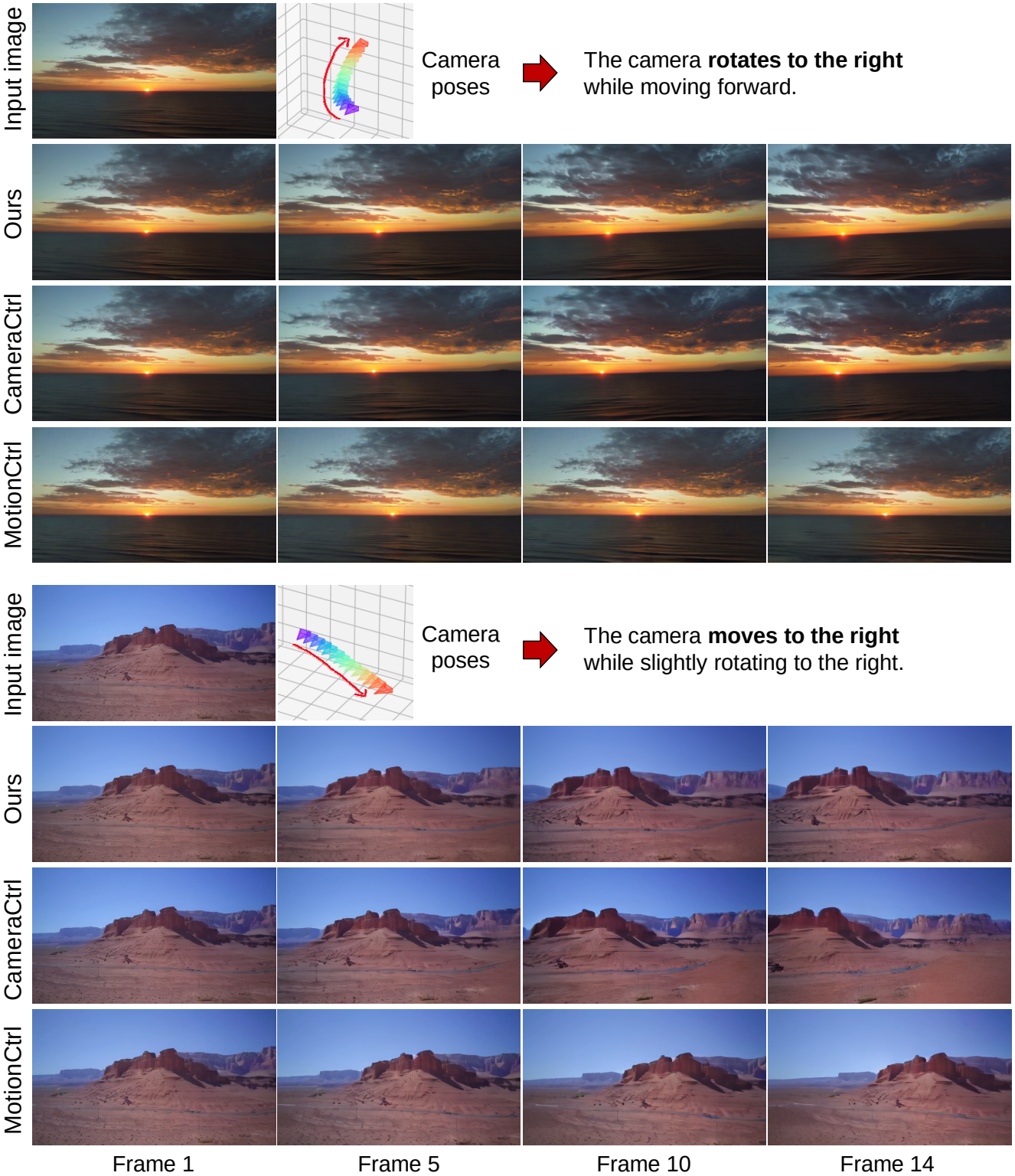

Figure S9. Additional qualitative comparison of our method against MotionCtrl [14] and CameraCtrl [4] using detailed camera trajectories. MotionCtrl often fails to follow the input camera parameters during video generation.

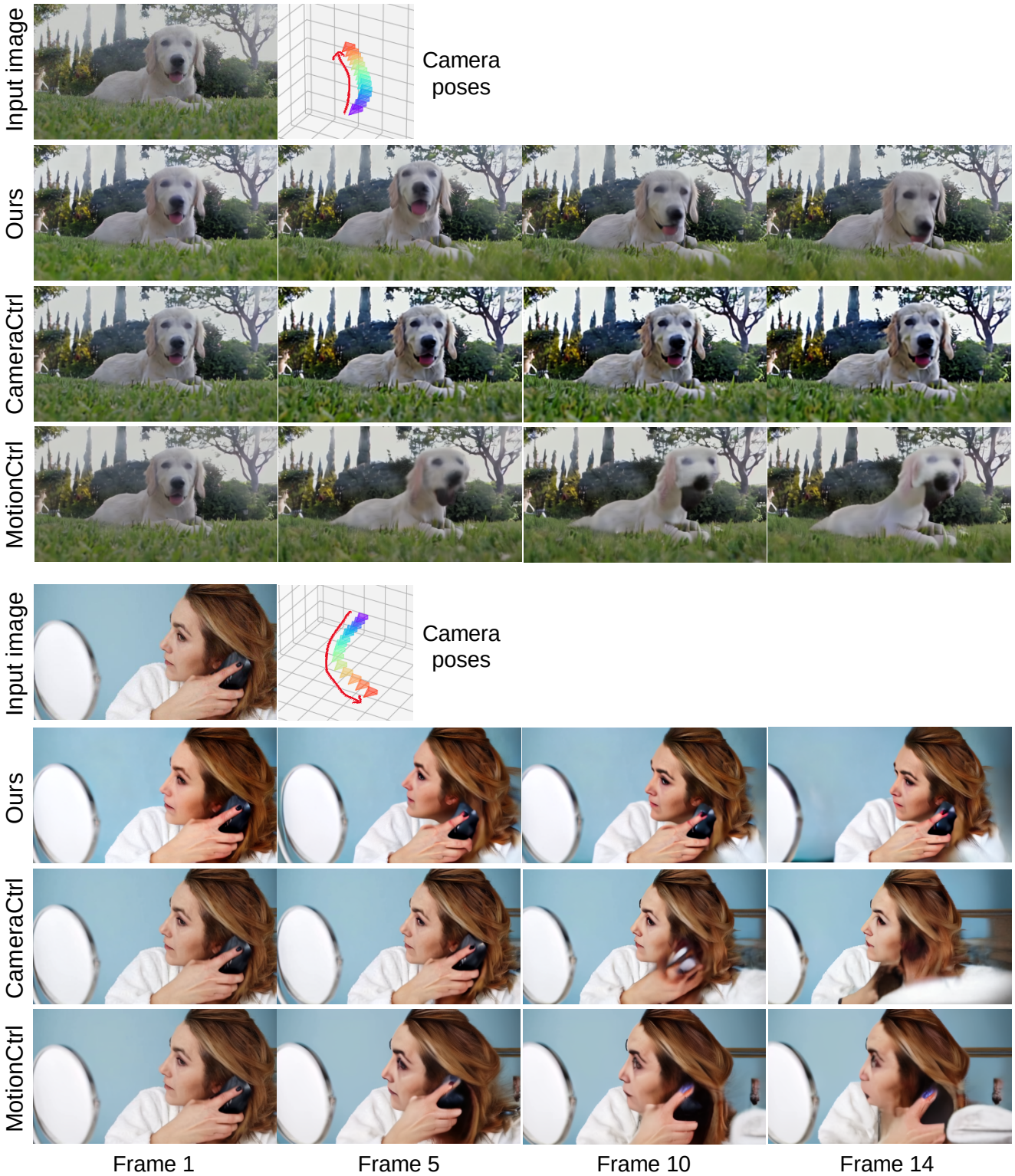

Figure S10. Additional qualitative comparison of our method against MotionCtrl [14] and CameraCtrl [4] using detailed camera trajectories. Our method produces more natural object motion, while CameraCtrl produces a foreground object without motions, and MotionCtrl produces artifacts.

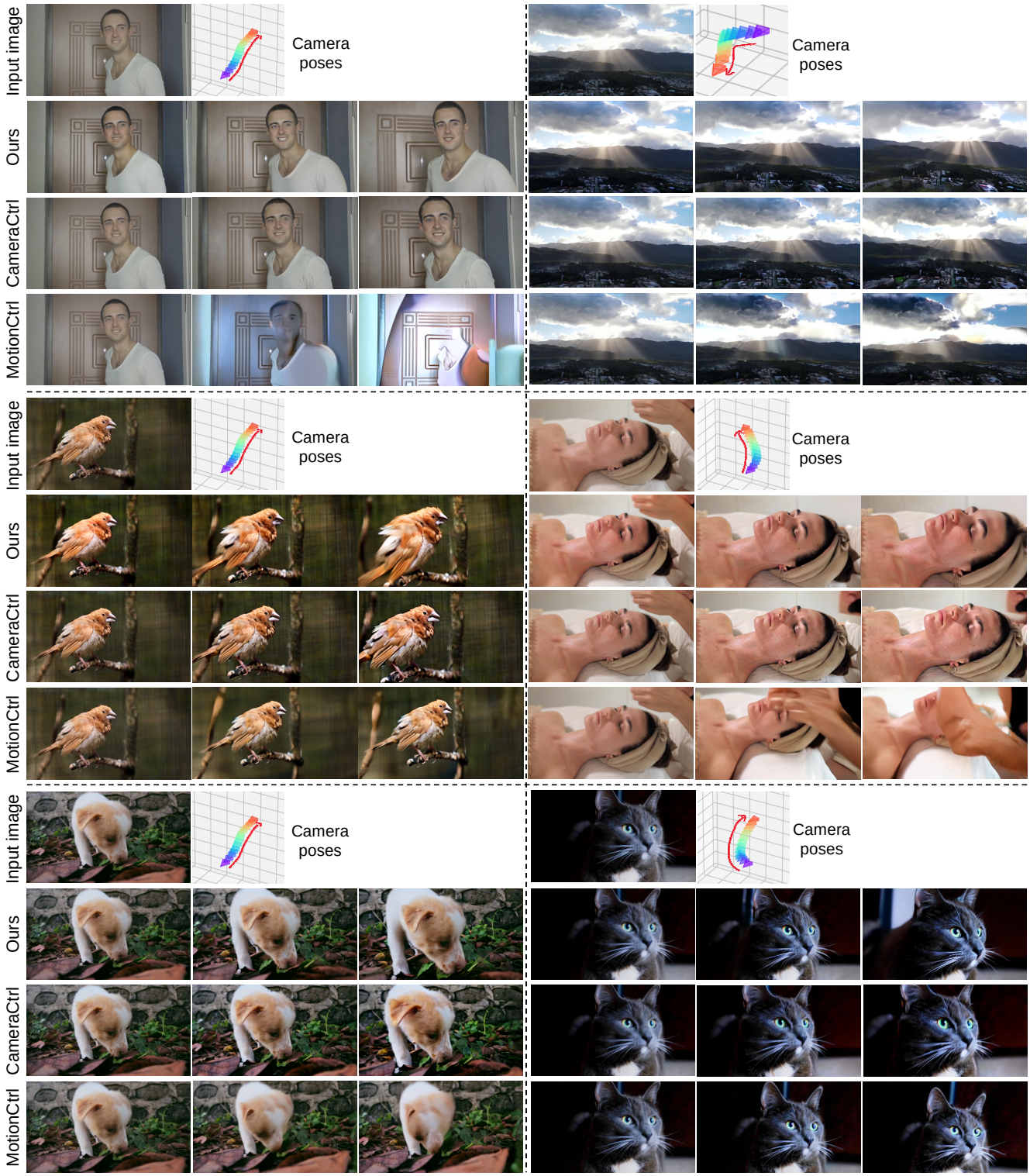

Figure S11. Additional qualitative comparison of our method against MotionCtrl [14] and CameraCtrl [4] using detailed camera trajectories.

References

[1] Pexels, royalty-free stock footage website. https://www.pexels.com. Accessed: 2024-09-30. 2, 5

[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

[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. 2, 4, 5, 7

[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, 4, 5, 6, 8, 9, 10

[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] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 4

[10] 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

[11] 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 *International conference on machine learning*, pages 8748–8763. PmLR, 2021. 4

[12] 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

[13] 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

[14] 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. 2, 4, 5, 6, 8, 9, 10

[15] Sihan Xu, Yidong Huang, Jiayi Pan, Ziqiao Ma, and Joyce Chai. Inversion-free image editing with natural language. *arXiv preprint arXiv:2312.04965*, 2023. 1

[16] 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, 3

[17] 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. 2, 4, 5, 7

[18] 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

[19] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817*, 2018. 3