A Model Implementation

In this section, we will provide more details about model implementation.

A.1 SEAL

The attention computation for the entire spacetime patches is essentially a modeling of very long sequences, and therefore, it is a memory-intensive computational process. We utilized the *memory_efficient_attention* function from Xformers to reduce memory consumption. In the case of inputting epipolar attention masks, we modified the function so that the computed attention weights are forcibly assigned a value of negative infinity at the positions where the mask value is 1.

A.2 Camera Embeddings

For raw camera embeddings, we first flatten the rotation matrix and translation vector into a vector of length 12. Then, we transform it into a 96-dimensional vector through positional encoding. For the Plücker embedding, we pre-compute an embedding with the same resolution as the UNet input. For different sizes, we use pixel unshuffle to change its size while preserving as much fine-grained positional information as possible.

A.3 Embedding coordination

In the case of I2V-adapter, we can directly use the epipolar attention masks computed in SEAL to extract effective spatial information from the context frame. Similarly, the precomputed camera embedding can be inserted into the IP-adapter to balance the expression of text and image information.

In leveraging camera embedding to balance the expression of text and image information, the activation function is sigmoid function for fully-connected layers and its output p and $1 - p$ will multiply image and text embedding respectively to ensure mutual exclusive between two conditions.

B Technical Details

B.1 Data

For RealEstate dataset, we use 67,477 scenes for training and 7,289 scenes for testing. For EpicKitchen, we use 611 scenes for training and 88 scenes for testing. It should be noted that the duration of each scene of EpicKitchen is much longer than that in RealEstate. That's why the number of scenes in EpicKitchen is much smaller.

Following the protocol of CameraCtrl, we use BLIP to annotate every frame of both dataset for text prompting. The original captions provided in EpicKitchen are overly simplistic, such as "wash hand" or "open refrigerator." Additionally, they only annotate cooking-related actions, and most egomotion frames do not have captions. Therefore, we have to re-annotate this dataset using BLIP. The following are some examples of BLIP captions: "a person using a brush to clean a sink";"a person holding a bowl in front of a microwave"; "a person standing in front of a TV holding a remote control." Most captions involve interactions like "move" or hand-related actions. Meanwhile, since most scenes are static in RealEstate, its BLIP captions are generally overall descriptions of scenes like "a bedroom with a bed, a bedstand with lamb on it, and a windows". These text prompt mainly provides details of unobserved parts for imaginative purposes. For each training sample, we concatenate the captions of its first and last frame to form a text prompt.

Besides, we found that a large number of videos in EpicKitchen suffer from significant motion blur due to rapid movement, which is detrimental to model learning. Therefore, we applied NAFNet [1] to perform motion deblurring on all frames.

During the training, we use sample stride of 8 for RealEstate and 4 for EpicKitchen. We also used several additional techniques for data augmentation. In RealEstate, since most scenes are static, we can reverse the videos to generate additional motion trajectories. In both datasets, we can randomly increase or decrease the sample stride by one step to obtain video clips with different speeds. As a result, each training sample is consist of a 14-frame video clip, a text prompt and camera poses for all frames.

B.2 Training and Inference

In the case of SVD and AnimateDiff, we keep the original parameters of our base models fixed and only optimize the newly introduced layers and their subsequent layers using the AdamW optimizer with learning rate 2×10^{-4}. All models are trained on 8 NVIDIA A100 GPUs for 300k iterations using an effective batch size 32. We use BF16 precision for training SVD.

We use DDIM scheduler with 1000 steps during training and 25 steps during inference.

B.3 Evaluation

For calculation of TransErr and RotErr, we basically follow the protocol from CameraCtrl. The main difference is that we adopted different camera poses. We sorted the translation and rotation of the real camera poses from two datasets, and then randomly selected poses from the top 20% to 40% and 40% to 60% ranges to classify them into hard and medium groups, respectively. Additionally, we designed simple linear motions and rotations to create the simple group. We combined the hard, medium, and simple groups in a 1:1:1 ratio to adequately test the camera poses. Therefore, our selection process, which filters out poses with larger motion ranges, results in more challenging evaluation. For FVD and SSIM, we follow the common practice.

C Ablation study

In this section, we evaluate the contribution of each module to the model's improvement through extensive ablation experiments. As shown in Table 1, each modification contributes to either the improvement of the video quality or better accuracy of camera control.

Method	RealEstate-I2V				EpicKitchen			
	TransErr	RotErr	FVD	SSIM	TransErr	RotErr	FVD	SSIM
AnimateDiff+I2V+MotionCtrl	14.10	1.71	488.3	0.797	16.31	1.70	1223.5	0.727
+ Plucker embedding	13.96	1.68	466.5	0.808	16.08	1.77	1213.0	0.728
+ SEAL	8.12	1.01	366.7	0.824	13.81	1.41	786.2	0.777
+ Concatenation	7.59	0.90	321.8	0.877	12.98	1.33	706.3	0.814
+ Camera coordination	**6.75**	**0.77**	**293.7**	**0.903**	**12.41**	**1.27**	**663.2**	**0.839**

Table 1: Ablation study. Note that for TransErr, RotErr and FVD, lower number indicates better performance while higher SSIM means better.

D Related Work

The exploration of egocentric 3D generation through direct camera control has undergone several paradigm shifts. InfiniteNature [4] stands out as an early influential work in this field. This project primarily utilizes traditional computer vision techniques. It takes a paired RGB image and a disparity map to construct a textured mesh and then renders from new perspectives by warping the textures to adjust disparities. To complete the process, a refinement network addresses and corrects any gaps or missing parts in the final output. This method does not fully leverage the learning capabilities of neural networks.

After the rise of generating images using transformer-based autoregressive models, GeoGPT [6] posits that all camera motion control conditions can be transformed into tokens or directly summed embeddings. They experimented with a wide variety of ways for incorporating control conditions to assess their impact on the generated results. However, their fusion methods only included concatenation and addition. Therefore, GeoGPT encounters the same issues as CameraCtrl and MotionCtrl, specifically limited ranges of generated motion. Therefore, subsequent studies primarily focused on how to introduce more explicit geometrical control. "Look Outside the Room" [5] proposed using Camera-Aware Bias by computing a bias with a Multi-Layer Perceptron (MLP), which takes the relative camera position as input during the calculation of the attention score. The paper [2] utilized epipolar attention to aggregate information from patches under different viewpoints to generate the final render. These efforts have achieved significant progress.

Subsequently, the era of diffusion models arrived. Researchers [7] introduced epipolar attention into the use of diffusion models for 3D generation. SceneScape [3] returned to the framework of InfinitNature, but with the aid of a more powerful diffusion model. It integrated text control capabilities and achieved better results.

References

[1] Xiaojie Chu, Liangyu Chen, and Wenqing Yu. Nafssr: Stereo image super-resolution using nafnet. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 1239–1248, June 2022.

[2] Yilun Du, Cameron Smith, Ayush Tewari, and Vincent Sitzmann. Learning to render novel views from wide-baseline stereo pairs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4970–4980, 2023.

[3] Rafail Fridman, Amit Abecasis, Yoni Kasten, and Tali Dekel. Scenescape: Text-driven consistent scene generation. *Advances in Neural Information Processing Systems*, 36, 2024.

[4] Andrew Liu, Richard Tucker, Varun Jampani, Ameesh Makadia, Noah Snavely, and Angjoo Kanazawa. Infinite nature: Perpetual view generation of natural scenes from a single image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14458–14467, 2021.

[5] Xuanchi Ren and Xiaolong Wang. Look outside the room: Synthesizing a consistent long-term 3d scene video from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3563–3573, 2022.

[6] Robin Rombach, Patrick Esser, and Björn Ommer. Geometry-free view synthesis: Transformers and no 3d priors. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14356–14366, 2021.

[7] Hung-Yu Tseng, Qinbo Li, Changil Kim, Suhib Alsisan, Jia-Bin Huang, and Johannes Kopf. Consistent view synthesis with pose-guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16773–16783, 2023.