MotionCtrl: A Unified and Flexible Motion Controller for Video Generation

https://wzhouxiff.github.io/projects/MotionCtrl/

Zhouxia Wang[1,2*] Ziyang Yuan[1,4*] Xintao Wang[1,3] ✉ Tianshui Chen[6]
Menghan Xia[3] Ping Luo[2,5] ✉ Ying Shan[1,3]

[1] ARC Lab, Tencent PCG [2] The University of Hong Kong [3] Tencent AI Lab
[4] Tsinghua University [5] Shanghai AI Laboratory [6] Guangdong University of Technology

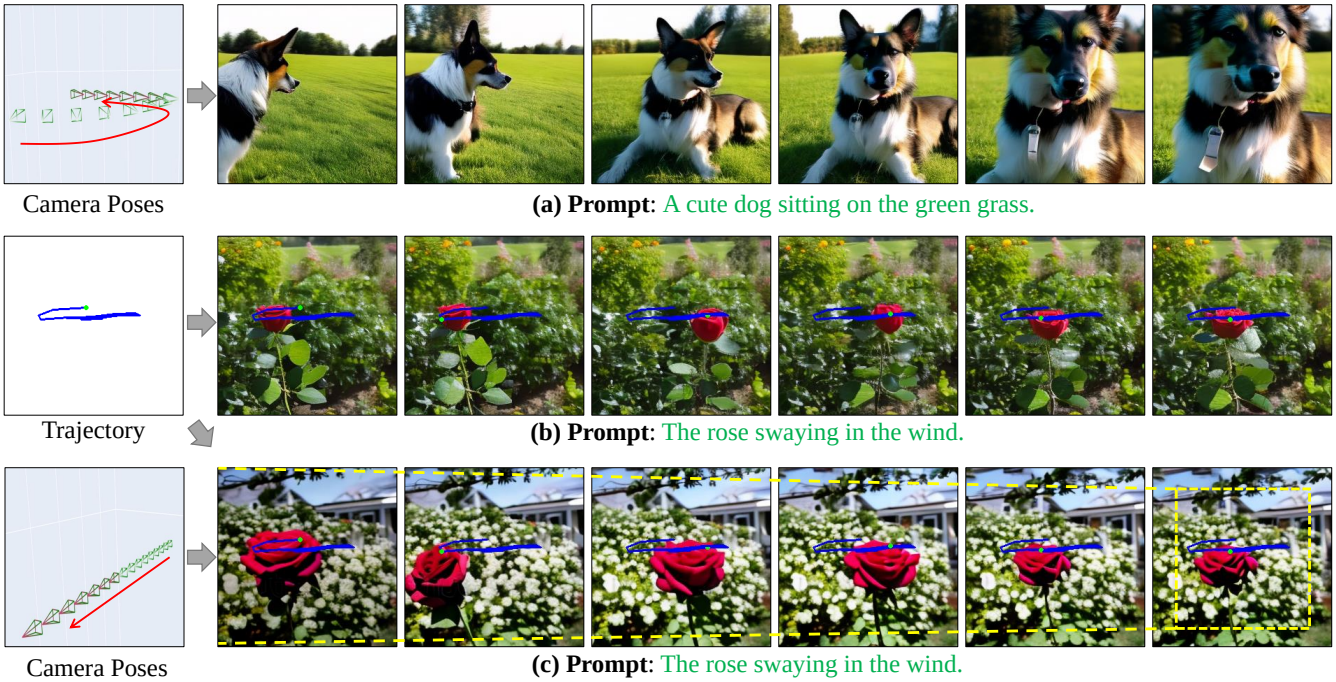


Figure 1. **Control Results of MotionCtrl**. MotionCtrl is capable of controlling both camera motion and object motion in videos produced by a video generation model. It can also simultaneously control both types of motion within the same video. **We highly encourage readers to check our project page for video results, which cannot be well demonstrated by still images.**

Abstract

Motions in a video primarily consist of camera motion, induced by camera movement, and object motion, resulting from object movement. Accurate control of both camera and object motion is essential for video generation. However, existing works either mainly focus on one type of motion or do not clearly distinguish between the two, limiting their control capabilities and diversity. Therefore, this paper presents MotionCtrl, a unified and flexible motion controller for video generation designed to effectively and independently control camera and object motion. The architecture and training strategy of MotionCtrl are carefully devised, taking into account the inherent properties of camera motion, object motion, and imperfect training data. Compared to previous methods, MotionCtrl offers three main advantages: 1) It effectively and independently controls camera motion and object motion, enabling more fine-grained motion control and facilitating flexible and diverse combinations of both types of motion. 2) Its motion conditions are determined by camera poses and trajectories, which are appearance-free and minimally impact the appearance or shape of objects in generated videos. 3) It is a relatively generalizable model that can adapt to a wide array of camera poses and trajectories once trained. Extensive qualitative and quantitative experiments have been conducted to demonstrate the superiority of MotionCtrl over existing methods.

* Interns in ARC Lab, Tencent PCG ✉ Corresponding author

1. Introduction

Video generation, such as text-to-video (T2V) generation [4, 5, 9, 10, 23, 32] aims to produce diverse and high-quality videos that conform to given text prompts. Unlike image generation [6, 17–19, 21, 34], which focuses on generating a single image, video generation necessitates the creation of consistent and fluent motion among a sequence of generated images. Consequently, motion control plays a significantly crucial role in video generation, yet it has received limited attention in recent research.

In a video, there are primarily two types of motion: global motion induced by camera movements and local motion resulting from object movements (examples are referred to the zoom out camera poses and swaying rose in Figure 1 (c)). It should be noted that these two motions will be consistently referred to as **camera motion** and **object motion** throughout the paper, respectively. However, most previous works related to motion control in video generation either primarily focus on one of the motions or lack a clear distinction between these two types of motion. For instance, AnimateDiff [8], Gen-2 [7], and PikaLab [2] mainly execute or trigger camera motion control using independent LoRA [11] models or extra camera parameters (such as "-camera zoom in" in PikaLab [2]). VideoComposer [25] and DragNUWA [28] implement both camera motion and object motion using the same conditions: motion vector in VideoComposer [25] and trajectory in DragNUWA [28]. The lack of clear distinction between these two motions prevents these approaches from achieving fine-grained and diverse motion control in video generation.

In this paper, we introduce MotionCtrl, a unified and flexible motion controller for video generation, designed to independently control camera and object motion with a unified model. This approach enables fine-grained motion control in video generation and facilitates flexible and diverse combinations of both motion types. However, constructing such a unified motion controller presents significant challenges due to the following two factors. First, camera and object motions differ significantly in terms of movement range and pattern. Camera motion refers to the global transformation of the whole scene across the temporal dimension, which is typically represented through a sequence of camera poses over time. In contrast, object motion involves the temporal movement of specific objects within the scene, and it is usually represented as the trajectory of a cluster of pixels associated with the objects. Second, no existing dataset encompasses video clips that are accompanied by a complete set of annotations, including captions, camera poses, and object movement trajectories. Creating such a comprehensive dataset requires a significant amount of effort and resources.

In this paper, we design a specialized architecture and training strategy for MotionCtrl to address the aforementioned challenges. First, we construct MotionCtrl with two modules: the Camera Motion Control Module (CMCM) and Object Motion Control Module (OMCM), specifically tailored for camera motion and object motion according to their properties. Both CMCM and OMCM cooperate with the existing video generation model. Noted that in this work, we adopt VideoCrafter1 [5], an improved version over LVDM [9] as the video generation model, and refer it as LVDM throughout the paper. CMCM temporally fuses a sequence of camera poses into the LVDM [9] via its temporal transformers, enabling the global motion of the generated video to conform to the given camera poses. OMCM spatially fuses the information on object movement into the convolutional layers in LVDM to indicate the spatial position of the object in each generated frame.

In this study, we utilize VideoCrafter[5], a model structurally akin to LVDM[9]. For clarity, we will refer to the video model used in this work as LVDM."

Training the MotionCtrl requires video clips with complete annotations of captions, camera poses, and object movement trajectories, which are currently unavailable and quite difficult to construct. Benefiting from the careful-designed architecture that depends on a large-scale pre-trained video diffusion model equipped with two adapter-like [15, 29] CMCM and OMCM modules, we can train the two modules independently and thus compromise the datasets to one video dataset with the annotations of captions and camera poses and another video dataset with the annotations of captions and object movement trajectories. To this end, we first introduce an augmented-Realestate10k dataset, which exploits Blip2 [13] to generate captions for each sample from Realestate10k [33]. This video dataset with both caption and camera pose annotations is adopted to train the CMCM module. After that, we augment the video from WebVid [3] with object movement trajectories synthesized by a motion segmentation algorithm proposed in ParticleSfM [31]. Thus, we can obtain an augmented WebVid dataset to train the OMCM module. Benefiting from the independent adapter-like training strategy and frozen pre-trained LVDM, we can use one of the camera and object motions or combine both motions to control video generation, enabling fine-grained and flexible motion control.

Through these delicate designs, MotionCtrl demonstrates superiority over previous methods in three aspects: 1) It independently controls camera and object motion, enabling fine-grained adjustments and a variety of motion combinations, as shown in Figure 1. 2) It uses camera poses and trajectories for motion conditions, which do not affect the visual appearance, maintaining the objects' natural look in videos. For instance, our MotionCtrl generates a video with a camera motion that closely reflects the reference video, offering a realistic Eiffel Tower, as seen in Figure 4 (b). In contrast, VideoComposer [25] relies on dense

motion vectors and mistakenly captures a door's shape of the reference video, resulting in an unnatural Eiffel Tower. 3) MotionCtrl can control a variety of camera movements and trajectories, without the need for fine-tuning each individual camera or object motion.

The main contributions of this work can be summarized as follows:

- We introduce MotionCtrl, a unified and flexible motion controller for video generation, designed to independently and effectively control camera motion and object motion in generated videos, achieving more fine-grained and diverse motion control.
- We carefully tailor the architecture and training strategy of MotionCtrl according to the inherent properties of camera motion, object motion, and imperfect training data, effectively achieving fine-grained motion control in video generation.
- We conduct extensive experiments to demonstrate the superiority of MotionCtrl over previous related methods, both qualitatively and quantitatively.

2. Related Works

Video generation, especially text-to-video (T2V) generation strives to create varied and high-fidelity videos that align with specified textual descriptions. With advancements in diffusion models and the availability of robust computational resources, a plethora of diffusion-based video generation models have emerged[4, 5, 8–10, 23, 25, 32]. Notably, deploying diffusion models in latent space[4, 9, 19] has enhanced the computational efficiency of video generation. Concurrently, there has been a surge in research focused on controlling motion within generated videos. Many existing approaches learn motion by referencing specific or a series of template videos[8, 26, 27, 30]. While effective at motion control, these methods typically require training a new model for each template or set of templates, which can be limiting.

Some efforts aim to achieve more generalized motion control. For instance, VideoComposer [25] introduces motion control via extra provided motion vectors, and DragNUWA [28] suggests video generation conditioned on an initial image, provided trajectories, and text prompts. However, the motion control in these methods is relatively broad and fails to fine-grainedly disentangle the camera and object motion within videos.

Different from these works, we propose MotionCtrl, a unified and flexible motion controller that can use either the camera poses and object trajectories or combine these two kinds of guidance to control the motion of generated videos. It enables a more fine-grained and flexible control for video generation.

3. Methodology

3.1. Preliminary

Latent Video Diffusion Model (LVDM). The Latent Video Diffusion Model (LVDM) [9] aims to generate high-quality and diverse videos guided by text prompts. It employs a denoising diffusion model (U-Net [20]) in the latent space for space and time efficiency. Consequently, it constructs a lightweight 3D autoencoder, comprising an encoder $\mathcal{E}$ and a decoder $\mathcal{D}$, to encode raw videos into the latent space and reconstruct the denoised latent features back into videos, respectively. Its denoising U-Net (denoted as ϵ_θ) is constructed with a sequence of blocks that consist of convolutional layers, spatial transformers, and temporal transformers (shown in Figure 2). It is optimized using a noise-prediction loss:

$$\mathcal{L} = \mathbb{E}_{z_0,c,\epsilon\sim\mathcal{N}(0,I),t}\left[\|\epsilon - \epsilon_\theta(z_t, t, c)\|_2^2\right], \tag{1}$$

where c represents the text prompt, z_0 is the latent code obtained using $\mathcal{E}$, t ($t \in [0, T]$) denotes the time step, and z_t is the noisy latent features acquired by weighted addition of Gaussian noise ϵ to z_0 using the following formula:

$$z_t = \sqrt{\bar{\alpha}_t} z_0 + \sqrt{1-\bar{\alpha}_t}\epsilon,\ \bar{\alpha}_t = \prod_{i=1}^{t} \alpha_t, \tag{2}$$

where α_t is used for scheduling the noise strength based on time step t.

3.2. MotionCtrl

Figure 2 illustrates the framework of MotionCtrl. To achieve disentanglement between camera motion and object motion, and enable independent control of these two types of motion, MotionCtrl comprises two main components: a Camera Motion Control Module (CMCM) and an Object Motion Control Module (OMCM). Taking into account the global property of camera motion and the local property of object motion, CMCM interacts with the temporal transformers in LVDM, while OMCM spatially cooperates with the convolutional layers in LVDM. Furthermore, we employ multiple training steps to adapt MotionCtrl to the absence of training data that contains high-quality video clips accompanied by captions, camera poses, and object movement trajectories. In the following subsections, we will provide a detailed description of CMCM and OMCM along with their corresponding training datasets and training strategies.

3.2.1 Camera Motion Control Module (CMCM)

The CMCM is a lightweight module constructed with several fully connected layers. Since the camera motions are global transformations between frames in a video, CMCM

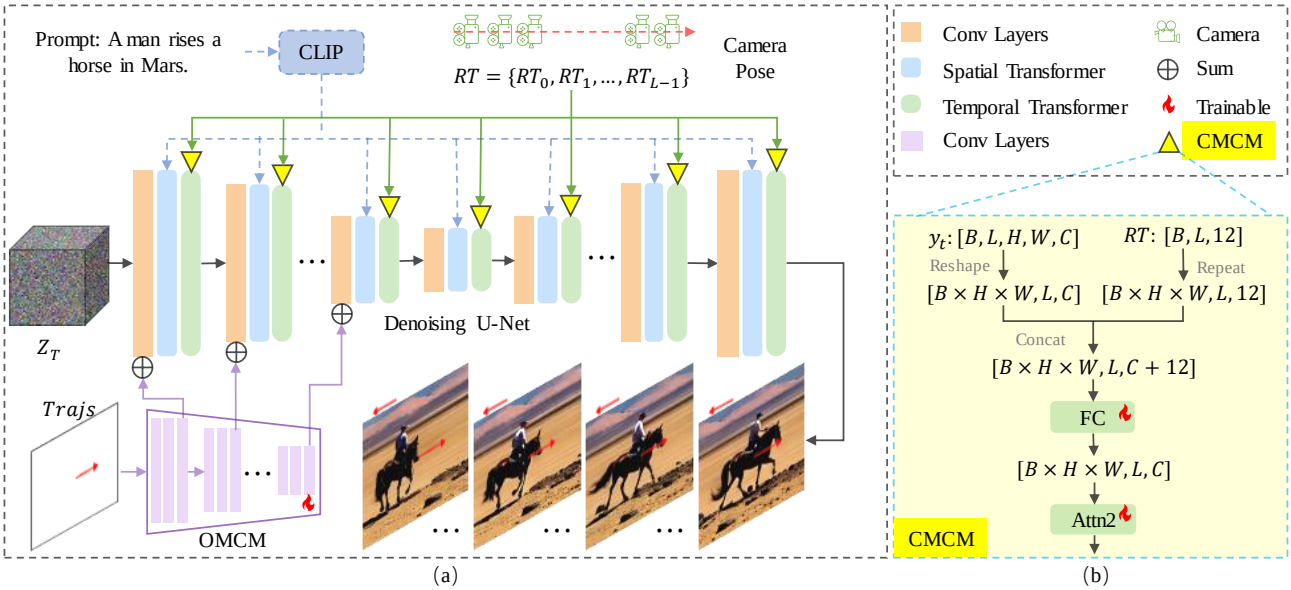


Figure 2. **MotionCtrl Framework**. MotionCtrl extends the Denoising U-Net structure of LVDM with a Camera Motion Control Module (CMCM) and an Object Motion Control Module (OMCM). As illustrated in (b), the CMCM integrates camera pose sequences RT with LVDM's temporal transformers by appending RT to the input of the second self-attention module and applying a tailored and lightweight fully connected layer to extract the camera pose feature for subsequent processing. The OMCM utilizes convolutional layers and downsamplings to derive multi-scale features from $Trajs$, which are spatially incorporated into LVDM's convolutional layers to direct object motion. Further given a text prompt, LVDM generates videos from noise that correspond to the prompt, with background and object movements reflecting the specified camera poses and trajectories. The resulting video demonstrates the horse moving along its trajectory and meanwhile, the background moves left, consistent with the camera's rightward motion.

cooperates with LVDM [9] via its temporal transformers. Typically, the temporal transformers in LVDM comprise two self-attention modules and facilitate temporal information fusion between video frames. To minimize the impact on LVDM's generative performance, CMCM only involves the second self-attention module in the temporal transformers. Specifically, CMCM takes a sequence of camera poses $RT = \{RT_0, RT_1, \ldots, RT_{L-1}\}$ as input. In this paper, the camera pose is represented by its 3×3 rotation matrix and 3×1 translation matrix. Consequently, $RT \in \mathbb{R}^{L \times 12}$, where L denotes the length of the generated video. As depicted in Figure 2 (b), RT is extended to $H \times W \times L \times 12$ before being concatenated with the output of the first self-attention module in the temporal transformer ($y_t \in \mathbb{R}^{H \times W \times L \times C}$) along the last dimension, where H and W represent the latent spatial size of the generated video, and C is the number of channels in y_t. The concatenated results are then projected back to the size of $H \times W \times L \times C$ using a fully connected layer before being fed into the second self-attention module in the temporal transformer.

3.2.2 Object Motion Control Module (OMCM)

As depicted in Figure 2, MotionCtrl controls the object motion of the generated video using trajectories ($Trajs$). Typically, a trajectory can be represented as a sequence of spatial positions $\{(x_0, y_0), (x_1, y_1), \ldots, (x_{L-1}, y_{L-1})\}$, where $(x_i, y_i), i \in [0, T-1]$ indicates that the trajectory passes through the i_{th} frame at the spatial position (x, y). While implemented, we tend transform the trajectory $\{(x_0, y_0), (x_1, y_1), \ldots, (x_{L-1}, y_{L-1})\}$ into $\{(0, 0), (u_1, v_1), \ldots, (u_{L-1}, v_{L-1})\}$, where

$$u_i = x_i - x_{i-1}, v_i = y_i - y_{i-1}, i > 1. \quad (3)$$

Denoted that the other spatial positions that the trajectories do not pass are described as $(0, 0)$ and $Trajs \in \mathbb{R}^{L \times H \times W \times 2}$.

The OMCM spatially incorporates $Trajs$ into the LVDM generation process, as these trajectories represent the spatial positions of the object in each frame. The OMCM's architecture, highlighted in the purple block of Figure 2, consists of multiple convolutional layers combined with downsampling operations. It extracts multi-scale features from the $Trajs$ and corresponding adds them to the input of the LVDM's convolutional layers. Drawing inspiration from previous works such as ControlNet[29] and T2I-Adapter [15], the trajectories are only applied to the encoder of the Denoising U-Net to balance the generated video quality with the ability to control object motion.

3.2.3 Training Strategy and Data Construction

To achieve the control of camera and object motion while generating a video via text prompts, video clips in a training dataset must contain annotations of captions, camera poses, and object movement trajectories. However, a dataset with such comprehensive details is currently unavailable, and

assembling one would require considerable effort and resources. To address this challenge, we introduce a multi-step training strategy and train our proposed camera motion control module (CMCM) and object motion control module (OMCM) with distinct augmented datasets tailored to their specific motion control requirements.

Learning the camera motion control module (CMCM) requires a training dataset that contains video clips with captions, and camera poses, but not object movement trajectories. To this end, we opt to use the Realestate10K dataset [33], which, after removing invalid video links, offers 62,992 video clips accompanied by diverse camera poses. Nonetheless, employing Realestate10K presents two potential challenges: 1) the dataset's limited diversity of scenes, primarily from real estate videos, potentially compromising the quality of the generated video; and 2) it lacks captions needed for T2V models.

Regarding the first challenge, we specifically design the CMCM module to only train several extra MLP layers and the second self-attention module of the temporal transformers in LVDM while freezing all other parameters (as illustrated in Figure 2 (b).). The training of CMCM mainly focuses on learning global motion and seldom affects the content of the generated video. Therefore, the limited scene diversity in the Realestate10K dataset has a negligible effect on the generated quality of LVDM. This is substantiated by quantitative results presented in Table 2, where the FID [22] and FVD [24] metrics indicate that the video quality generated by our MotionCtrl is on par with the LVDM outcomes.

To address the second challenge, we utilize Blip2 [13], an image captioning algorithm, to generate captions for each video clip in Realestate10K. We extract frames at specific intervals—the first, quarter, half, three-quarters, and final frames of the video. We then use Blip2 to predict their captions. These captions are then concatenated to form a comprehensive description for each video clip. With these captions in place, we train the CMCM on Realestate10K, which in turn allows the LVDM to control camera motion effectively.

Learning the object motion control module (OMCM) requires a dataset comprising video clips with corresponding captions and object movement trajectories. Currently, there is an absence of large-scale datasets that combine video-text pairs with object trajectories. To address this, we employ ParticleSfM [31] to generate object movement trajectories using the WebVid dataset [3], which is widely used for T2V generation. ParticleSfM features a trajectory motion segmentation network capable of identifying trajectories associated with moving objects within a video, as depicted in Figure 3 (b), where the trajectories predominantly correspond to a moving person. Despite its effectiveness, ParticleSfM is not time-efficient, requiring approximately 2 minutes to process a 32-frame video. To balance time efficiency and data coverage, we opt to randomly select 32 frames from each WebVid video, with a frame skip interval $s \in [1, 16]$, to precompute the object movement trajectories. This approach yields a total of 243,000 samples that fulfill the training requirements for the OMCM.

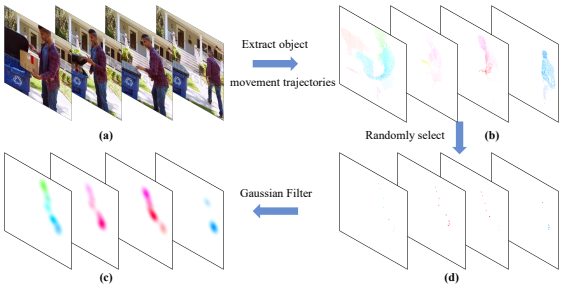


Figure 3. **Trajectories for Object Motion Control**. ParticleSfM [31] is employed to extract object movement trajectories from video clips, effectively disentangling object motion from camera-induced movement. To circumvent the issues of dense trajectories, which can encode object shapes and are challenging to design at inference, we train the OMCM using sparse trajectories sampled from the dense ones. These sparse trajectories, being too scattered for effective learning, are subsequently refined with a Gaussian filter.

To enhance user experience and interaction, we utilize sparse trajectories to direct object motion instead of dense ones. During the training process, we randomly select $n \in [1, N]$ trajectories (where N is the maximum number of trajectories for each video) from the synthesized trajectories. An example of the outcome can be seen in Figure 3 (c). However, these selected sparse trajectories can be too scattered for effective training. To address this issue, we apply a Gaussian filter to the sparse trajectories (Figure 3 (d)), inspired by DragNUWA [28]. During training, we initially train the OMCM using dense trajectories and then proceed to fine-tune it with sparse trajectories.

In this training phase, we adopt the LVDM model that is already fine-tuned with CMCM. We only train the OMCM layers, while the entire base model and CMCM remains frozen. This strategy guarantees that OMCM adds the object motion control capabilities with a limited dataset while minimally impacting LVDM and CMCM. Upon completion of this training phase, giving both camera poses and trajectories allows for flexible customization of camera and object motion in the generated video.

4. Experiments

4.1. Experiment Settings

Implementation Details. Our MotionCtrl is based on the LVDM [9] and the trained weights are provided by VideoCraft1 [5]. It is trained on 16-frame sequences at a resolution of 256×256. We set the maximum number of trajectories N to 8. Both CMCM and OMCM are optimized using the Adam optimizer [12] with a batch size of 128

and a learning rate of 1×10^{-4} on 8 NVIDIA Tesla V100 GPUs. The CMCM requires approximately 50,000 iterations to converge. OMCM is initially trained on dense trajectories for 20,000 iterations, followed by fine-tuning with sparse trajectories for an additional 20,000 iterations.

Evaluation Datasets. We independently evaluate the efficacy of our proposed MotionCtrl on camera motion control and object motion control. These assessments are conducted using their respective evaluation datasets. More details about the two evaluation datasets are provided in the supplementary materials.

Camera motion control evaluation dataset. MotionCtrl's camera motion control is tested on a set of 8 basic camera pose sequences: pan left, pan right, pan up, pan down, zoom in, zoom out, anticlockwise rotation, and clockwise rotation. Additionally, we randomly select 20 complicated camera pose sequences from the Realestate10K test set [33]. For each camera pose sequence, we evaluate 10 different text prompts.

Object motion control evaluation dataset. For object motion control, we create 19 diverse trajectories, including straight lines, curves, shaking lines, and complex combinations of those. For each trajectory, we evaluate 12 different text prompts.

Evaluation Metrics. 1) The quality of the generated videos is evaluated using Fréchet Inception Distance (**FID**)[22], Fréchet Video Distance (**FVD**)[24], and CLIP Similarity (**CLIPSIM**) [16], which measure the visual quality, temporal coherence, and semantic similarity to the input text, respectively. Denoted that the reference videos of FID and FVD are 1000 videos selected from WebVid [3]. 2) The efficacy of the camera motion control and object motion control is quantified by computing the Euclidean distance between the predicted and ground truth camera poses and object trajectories. The camera poses and object trajectories for the predicted videos are extracted using ParticleSfM [31]. We title these two metrics as **CamMC** and **ObjMC**, respectively.

4.2. Comparisons with State-of-the-Art Methods

To validate the effectiveness of our MotionCtrl in controlling both camera and object motion, we compare it with two leading methods: AnimateDiff [8] and VideoComposer [25]. AnimateDiff employs 8 separate LoRA [11] models to control 8 basic camera motions in videos, such as panning and zooming, while VideoComposer manipulates video motion using motion vectors without differentiating between camera and object movements. Although DragNUWA [28] is relevant to our research, its code is not publicly available, precluding a direct comparison. Moreover, DragNUWA only learns motion control with the trajectories extracted from optical flow, which cannot fine-grainedly distinguish the movement between foreground objects and background, limiting its ability to precisely control camera and object motion.

Method	AnimateDiff [8]	VideoComposer [25]	**MotionCtrl**
CamMC ↓ (Basic Poses)	0.0548	-	**0.0289**
CamMC ↓ (Complex Poses)	-	0.1073	**0.0840**
ObjMC ↓	-	34.7778	**25.1198**
CLIPSIM ↑	0.2144	0.2219	**0.2324**
FID ↓	157.73	134.97	**130.29**
FVD ↓	1815.88	1045.82	**934.37**

Table 1. **Quantitative Comparisons with SOTA Methods.** Our MotionCtrl outperforms competing approaches in both camera and object motion control while also excelling at preserving text similarity and the quality of the video generation.

We compare our MotionCtrl with these methods in terms of camera motion and object motion control, and show the capability of our MotionCtrl in flexibly combining the control of camera motion and object motion in video generation. *More comparisons and video comparisons are provided in the supplementary materials.*

Camera Motion Control. We assess camera motion control using basic poses and complex poses from the Realestate10K test set. AnimateDiff [8] is limited to basic camera poses, while VideoComposer [25] handles complex poses by extracting motion vectors from provided videos. The qualitative results are shown in Figure 4. For basic poses, both MotionCtrl and AnimateDiff can produce videos with forward camera movement, but MotionCtrl can generate camera motion with varying speeds, while AnimateDiff is nonadjustable. Regarding complex poses, where the camera first moves left front and then forward, VideoComposer can mimic the reference video's camera motion using extracted motion vectors. However, the dense motion vectors inadvertently capture object shapes, such as a door's outline in the reference video (frame 12), resulting in an unnatural-looking Eiffel Tower. MotionCtrl, guided by rotation and translation matrices, generates more natural-looking videos with camera motion close to the reference.

Quantitative results in Table 1 show MotionCtrl's superiority over AnimateDiff and VideoComposer for both basic and complex poses, as reflected by the CamMC score. Additionally, MotionCtrl achieves better text similarity and quality metrics, as measured by CLIPSIM, FID, and FVD.

Object Motion Control. We compare our MotionCtrl with VideoComposer for object motion control, where VideoComposer utilizes motion vectors extracted from trajectories. The qualitative results are shown in Figure 4 (c). The red curve illustrates the given trajectory, while the green points indicate the expected object locations in the corresponding frame. The visual comparison reveals that MotionCtrl can generate objects whose movements are closer to the given trajectories, whereas VideoComposer's results deviate in certain frames, highlighting MotionCtrl's superior object motion control capability. The quantitative results in Table 1 demonstrate that MotionCtrl achieves better object motion control than VideoComposer, as reflected by

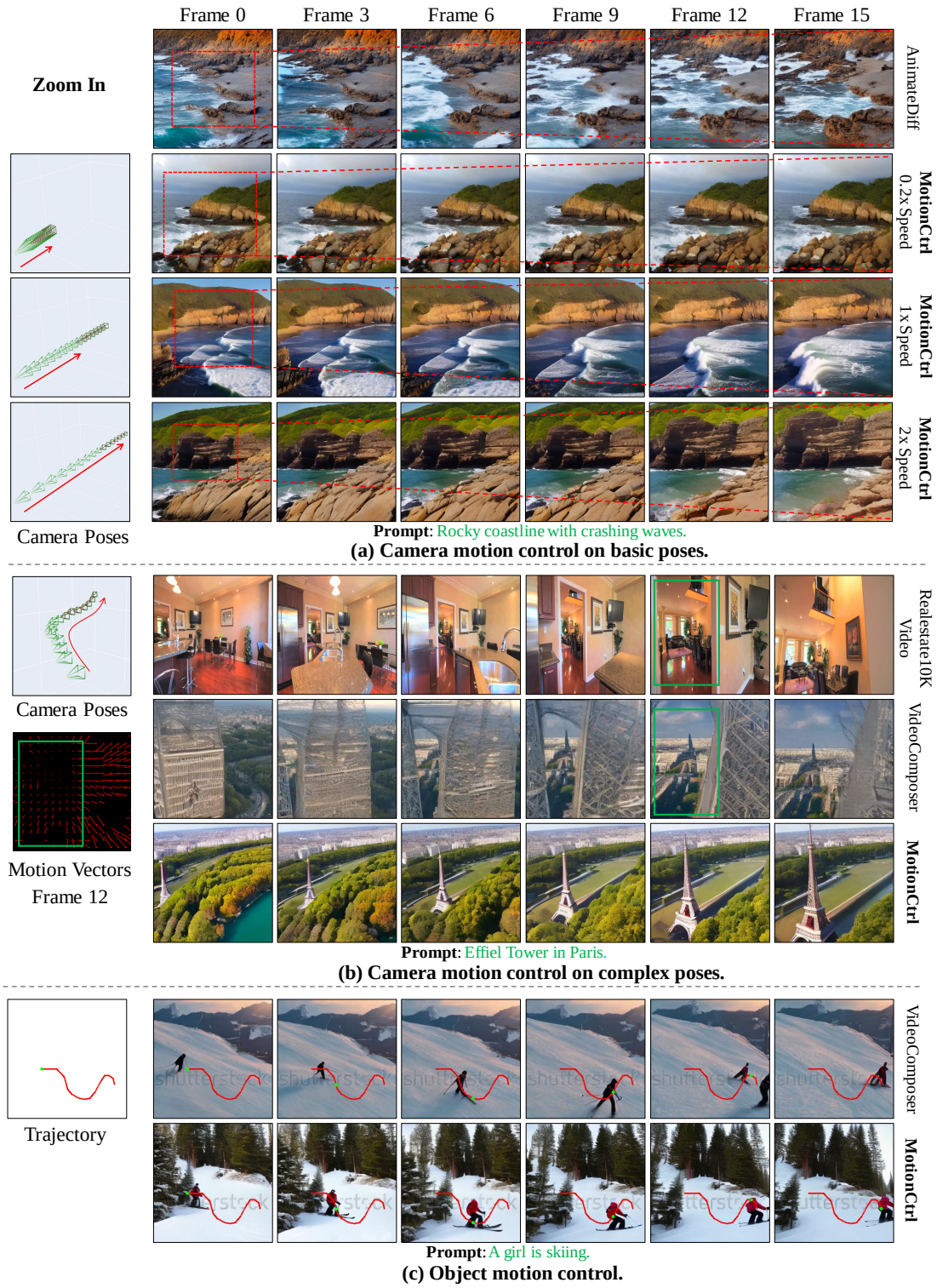


Figure 4. **Qualitative Comparisons on Camera and Object Motion Control.** (a) Basic Poses: MotionCtrl and AnimateDiff[8] effectively execute zooms, but MotionCtrl can adjust to varying camera moving speeds. (b) Complex Poses: VideoComposer[25] uses Realestate10K's raw video for motion vectors, capturing unintended shapes like doors, leading to unnatural results (refer to frame 12). MotionCtrl, however, produces a relatively natural video with motion that closely matches the camera poses. (c) Object Trajectory: Both VideoComposer and MotionCtrl can generate an object moving along a given trajectory (red curve), but MotionCtrl more precisely follows it in each frame, as indicated by green points.

the improved ObjMC score.

Combination of Camera Motion and Object Motion. MotionCtrl can not only control camera and object motion independently within a single video but also perform integrated control of both. As demonstrated in Figure 1 (b) and (c), when MotionCtrl is applied with only a trajectory, it primarily generates a swaying rose that follows this path. By further introducing zoom-out camera poses, both the rose and the background are animated in accordance with the specified trajectory and camera movements. **Additional results can be found in the supplementary materials and on the project page.**

4.3. Ablation Studies

In this section, we present experiments to validate the effectiveness of the delicate designs in MotionCtrl, including the integrated positioning of CMCM and LVDM, the training strategy for OMCM, and the sequence in which CMCM and OMCM are trained.

Integrated Position of Camera Motion Control Module (CMCM). We test implementing camera motion control by combining camera poses with the time embedding, spatial cross-attention, or spatial self-attention module in LVDM. Although such methods have succeeded in other types of controlling [14, 15, 29], such as sketch, depth, and so on, they fail to endow camera control capabilities to LVDM, as evidenced by the CamMC scores in Table 2, being close to LVDM without camera motion control. That is because these components primarily focus on spatial content generation, which is largely insensitive to the camera motion encoded in camera poses. Conversely, incorporating CMCM with LVDM's temporal transformers significantly improves camera motion control, as indicated by a lower CamMC score of 0.0289 in Table 2. Camera motion primarily causes global view transformations over time, and fusing camera poses into LVDM's temporal blocks aligns with this property, enabling effective camera motion control during video generation.

Furthermore, our CMCM has a negligible impact on the quality of spatial content, with FID[22] and FVD[24] scores comparable to LVDM, even though it is trained on the relatively small Realestate10K dataset [33] with low diversity. In contrast, injecting camera poses through time embedding or spatial transformers compromises quality scores, as it disrupts spatial content creation.

The corresponding qualitative results are in the supplementary materials.

Dense Trajectories *v.s.* Sparse Trajectories. We first train OMCM with dense object movement trajectories extracted via ParticleSfM[31] and then fine-tune with sparse trajectories. We evaluate the effectiveness of this approach by comparing it with training OMCM solely on dense or sparse trajectories. Table 3 indicates that training exclusively with dense trajectories yields inferior outcomes. This can be attributed to discrepancies between the training and inference phases (since only sparse trajectories are available during inference). While training solely with sparse trajectories shows improvement over the dense-only approach, it still falls short of the hybrid method. Sparse trajectories alone provide insufficient information for OMCM's learning. In contrast, dense trajectories offer richer information that accelerates learning, and subsequent fine-tuning with sparse trajectories allows OMCM to adjust to the sparsity encountered during inference.

Method	CamMC ↓	CLIPSIM ↑	FID ↓	FVD ↓
LVDM [9]	0.9010	0.2359	**130.62**	1007.63
Time Embedding	0.0887	0.2361	132.74	1461.36
Spatial Cross-Attention	0.0857	0.2357	153.86	1306.78
Spatial Self-Attention	0.0902	**0.2384**	146.37	1303.58
Temporal Transformer	**0.0289**	0.2355	132.36	**1005.24**

Table 2. **Ablation of Camera Motion Control.** Our Camera Motion Control Module (CMCM), incorporated with the temporal transformers of LVDM [9], effectively controls camera motion and maintains LVDM's video quality. Conversely, other variants fail to control the camera motion and may reduce the LVDM's generative quality.

Method	ObjMC ↓	CLIPSIM ↑	FID ↓	FVD ↓
Dense	54.4114	0.2352	175.8622	2227.87
Sparse	34.6937	0.2365	158.5553	2385.39
Dense + Sparse	**25.1198**	**0.2342**	**149.2754**	**2001.57**

Table 3. **Ablation of Object Motion Control**. The Object Motion Control Module (OMCM), when initially trained on dense object movement trajectories and subsequently fine-tuned with sparse trajectories, outperforms versions trained exclusively on either dense or sparse trajectories.

The corresponding qualitative results are in the supplementary materials.

Training Strategy. Given the limitations of the available training dataset, we propose a multi-step training strategy for MotionCtrl, starting with the Camera Motion Control Module (CMCM) using Realestate10K [33], followed by the Object Motion Control Module (OMCM) with synthesized object movement trajectories. To thoroughly assess our approach, we experiment with reversing the order, training OMCM before CMCM. This sequence does not impact camera motion control, as OMCM components do not participate in CMCM training. However, it does lead to a decrease in object motion control performance. The subsequent training of CMCM adjusts parts of LVDM's temporal transformers, disrupting the object motion control adaptation achieved during OMCM's initial training (evidenced by a higher ObjCM score of 25.2712 compared to 25.1198 attained with our proposed training sequence). Thus, our multi-step strategy, though a compromise due to dataset constraints, is deliberately structured to train CMCM before OMCM, ensuring enhanced performance in both cam-

era and object motion control.

5. Conclusion

This paper proposes MotionCtrl, a unified and flexible controller that can independently or combinably control the camera and object motion in a video attained with a video generation model. To achieve this end, MotionCtrl carefully tailors a camera motion control module and object motion control module to adapt to the specific properties of camera motion and object motion and deploys a multi-step training strategy to train these two modules with delicately augmented datasets. Comprehensive experiments, including qualitative and quantitative evaluations, showcase the superiority of our proposed MotionCtrl in both camera and object motion control.

Acknowledgements: We express our gratitude to Yaowei Li for his adaptation of our proposed MotionCtrl to AnimateDiff [8].

References

[1] Civitai: https://civitai.com/. 1

[2] https://www.pika.art/. 2

[3] Max Bain, Arsha Nagrani, Gül Varol, and Andrew Zisserman. Frozen in time: A joint video and image encoder for end-to-end retrieval. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1728–1738, 2021. 2, 5, 6

[4] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22563–22575, 2023. 2, 3

[5] Haoxin Chen, Menghan Xia, Yingqing He, Yong Zhang, Xiaodong Cun, Shaoshu Yang, Jinbo Xing, Yaofang Liu, Qifeng Chen, Xintao Wang, et al. Videocrafter1: Open diffusion models for high-quality video generation. *arXiv preprint arXiv:2310.19512*, 2023. 2, 3, 5

[6] Ming Ding, Zhuoyi Yang, Wenyi Hong, Wendi Zheng, Chang Zhou, Da Yin, Junyang Lin, Xu Zou, Zhou Shao, Hongxia Yang, et al. Cogview: Mastering text-to-image generation via transformers. *NeurIPS*, 2021. 2

[7] Patrick Esser, Johnathan Chiu, Parmida Atighehchian, Jonathan Granskog, and Anastasis Germanidis. Structure and content-guided video synthesis with diffusion models. *arXiv preprint arXiv:2302.03011*, 2023. 2

[8] Yuwei Guo, Ceyuan Yang, Anyi Rao, Yaohui Wang, Yu Qiao, Dahua Lin, and Bo Dai. Animatediff: Animate your personalized text-to-image diffusion models without specific tuning. *arXiv preprint arXiv:2307.04725*, 2023. 2, 3, 6, 7, 9, 1, 8

[9] Yingqing He, Tianyu Yang, Yong Zhang, Ying Shan, and Qifeng Chen. Latent video diffusion models for high-fidelity long video generation. *arXiv preprint arXiv:2211.13221*, 2023. 2, 3, 4, 5, 8, 1, 6, 7, 10

[10] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022. 2, 3

[11] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. 2, 6, 1

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[13] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023. 2, 5

[14] Yuheng Li, Haotian Liu, Qingyang Wu, Fangzhou Mu, Jianwei Yang, Jianfeng Gao, Chunyuan Li, and Yong Jae Lee. Gligen: Open-set grounded text-to-image generation. In *CVPR*, 2023. 8

[15] Chong Mou, Xintao Wang, Liangbin Xie, Jian Zhang, Zhongang Qi, Ying Shan, and Xiaohu Qie. T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models. *arXiv preprint arXiv:2302.08453*, 2023. 2, 4, 8

[16] 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 *ICML*, 2021. 6

[17] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *ICML*, 2021. 2

[18] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[19] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 2, 3

[20] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 3

[21] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022. 2

[22] Maximilian Seitzer. pytorch-fid: FID Score for PyTorch. https://github.com/mseitzer/pytorch-fid, 2020. 5, 6, 8

[23] Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, et al. Make-a-video: Text-to-video generation without text-video data. *arXiv preprint arXiv:2209.14792*, 2022. 2, 3

[24] Thomas Unterthiner, Sjoerd Van Steenkiste, Karol Kurach, Raphael Marinier, Marcin Michalski, and Sylvain Gelly. Towards accurate generative models of video: A new metric & challenges. *arXiv preprint arXiv:1812.01717*, 2018. 5, 6, 8

[25] Xiang Wang, Hangjie Yuan, Shiwei Zhang, Dayou Chen, Jiuniu Wang, Yingya Zhang, Yujun Shen, Deli Zhao, and Jingren Zhou. Videocomposer: Compositional video synthesis with motion controllability. *arXiv preprint arXiv:2306.02018*, 2023. 2, 3, 6, 7

[26] Jay Zhangjie Wu, Yixiao Ge, Xintao Wang, Weixian Lei, Yuchao Gu, Wynne Hsu, Ying Shan, Xiaohu Qie, and Mike Zheng Shou. Tune-a-video: One-shot tuning of image diffusion models for text-to-video generation. *arXiv preprint arXiv:2212.11565*, 2022. 3, 2

[27] Ruiqi Wu, Liangyu Chen, Tong Yang, Chunle Guo, Chongyi Li, and Xiangyu Zhang. Lamp: Learn a motion pattern for few-shot-based video generation. *arXiv preprint arXiv:2310.10769*, 2023. 3, 2

[28] Shengming Yin, Chenfei Wu, Jian Liang, Jie Shi, Houqiang Li, Gong Ming, and Nan Duan. Dragnuwa: Fine-grained control in video generation by integrating text, image, and trajectory. *arXiv preprint arXiv:2308.08089*, 2023. 2, 3, 5, 6

[29] Lvmin Zhang and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. *arXiv preprint arXiv:2302.05543*, 2023. 2, 4, 8

[30] Rui Zhao, Yuchao Gu, Jay Zhangjie Wu, David Junhao Zhang, Jiawei Liu, Weijia Wu, Jussi Keppo, and Mike Zheng Shou. Motiondirector: Motion customization of text-to-video diffusion models. *arXiv preprint arXiv:2310.08465*, 2023. 3, 2

[31] Wang Zhao, Shaohui Liu, Hengkai Guo, Wenping Wang, and Yong-Jin Liu. Particlesfm: Exploiting dense point trajectories for localizing moving cameras in the wild. In *ECCV*, 2022. 2, 5, 6, 8

[32] Daquan Zhou, Weimin Wang, Hanshu Yan, Weiwei Lv, Yizhe Zhu, and Jiashi Feng. Magicvideo: Efficient video generation with latent diffusion models. *arXiv preprint arXiv:2211.11018*, 2022. 2, 3

[33] 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. 2, 5, 6, 8

[34] Yufan Zhou, Ruiyi Zhang, Changyou Chen, Chunyuan Li, Chris Tensmeyer, Tong Yu, Jiuxiang Gu, Jinhui Xu, and Tong Sun. Lafite: Towards language-free training for text-to-image generation. *arXiv preprint arXiv:2111.13792*, 2021. 2

MotionCtrl: A Unified and Flexible Motion Controller for Video Generation

https://wzhouxiff.github.io/projects/MotionCtrl/

Supplementary Material

The supplementary materials provide additional results achieved with our proposed MotionCtrl, along with in-depth analyses. **For a more visual understanding, we strongly recommend readers visit our project page for the video results**. The structure of the supplementary materials is as follows:

- More results of MotionCtrl based on LVDM [9].(Section 6)
- Results of MotionCtrl when extended to AnimateDiff [8] framework. (Section 7)
- Visualized results and analyses of ablation studies. (Section 8)
- More discussions about previous related works. (Section 9).
- Details of evaluation datasets. (Section 10)

6. More Results of MotionCtrl Deployed on LVDM [9]

In this section, we present additional results of our proposed MotionCtrl model deployed on LVDM [9], focusing on camera motion control, object motion control, and combined motion control. **Notably, all results are obtained using the same trained MotionCtrl model, without the need for extra fine-tuning for different camera poses or trajectories.**

Specifically, Figure 5 illustrates the outcomes of camera motion control of MotionCtrl guided by 8 basic camera poses, including pan up, pan down, pan left, pan right, zoom in, zoom out, anticlockwise rotation, and clockwise rotation. These poses are visualized in Figure 15 (a). This demonstrates the capability of our MotionCtrl model to integrate multiple basic camera motion controls in a unified model, contrasting with the AnimateDiff model [8] which requires a distinct LoRA model [11] for each camera motion.

Figure 6 showcases the results of camera motion control using MotionCtrl, which is guided by relatively complex camera poses. These complex camera poses are distinct from basic camera poses, as they include elements of camera turning or self-rotation within the same camera pose sequence. The results demonstrate that, given a sequence of camera poses, our MotionCtrl can generate natural videos. The content of these videos aligns with the text prompts, and the camera motion corresponds to the provided complex camera poses. For additional results, please visit our project page.

Figure 7 presents the results of object motion control using MotionCtrl, guided by specific trajectories. When given the same trajectories and different text prompts, MotionCtrl can generate videos featuring different objects, but with identical object motion, as demonstrated in the first four samples of Figure 7. Furthermore, when provided with multiple trajectories within the same video, MotionCtrl is capable of controlling the motion of several objects simultaneously in the same generated video, as shown in the last samples of Figure 7.

Figure 8 provides the results of combining both the camera motion control and object motion control. With the same trajectory but different camera poses, the horse in the generated videos has a different performance.

7. Results of MotionCtrl Deployed on AnimateDiff [8]

We also deploy our MotionCtrl on AnimateDiff [8]. Therefore, we can control the motion of the video generated with our fine-tuned AnimateDiff cooperating with various LoRA [11] models in the committee. The results in Figure 11, 12, 9, and 10 are generated resutls of our MontionCtrl cooperating with different LoRA models provided by in CIVITAI [1]. They demonstrate that our MotionCtrl also works well in camera motion control and object motion control. Related results are also provided on the project page.

8. More Results and Analyses of Ablation Studies

The manuscript includes numerous ablation studies and detailed analyses. This section provides qualitative results of the ablation studies concerning the integrated position of the Camera Motion Control Module (CMCM) and the training strategy of the Object Motion Control Module (OMCM).

Specifically, Figure 13 demonstrates the generated results of LVDM[9] and our MotionCtrl deployed on LVDM, implemented by integrating CMCM with different components in LVDM. These components include the time embedding block, the cross-attention or self-attention module in the spatial transformers, and temporal transformers (accepted in our final MotionCtrl). The results indicate that MotionCtrl fails to control the camera motion of the generated video, except when integrating CMCM with the temporal transformers. This is primarily because camera motion mainly leads to global and temporal movement in the video. The camera poses integrated into the temporal transformer

can effectively fuse the camera movement information into the T2V generation model and yield camera motion.

Figure 14 illustrates the qualitative outcomes of the ablation study "Dense Trajectories vs. Sparse Trajectories". The results reveal that the Object Motion Control Module (OMCM) in MotionCtrl, when trained with dense trajectories, fails to control the object motion in the generated video. This is primarily due to the disparity between the dense trajectories in the training phase and sparse trajectories in the inference phase. Furthermore, the model trained on dense trajectories, followed by a fine-tuning phase on sparse trajectories, demonstrates superior precision in object motion control compared to the model trained exclusively on sparse trajectories. This is largely because sparse trajectories are too scattered to be effectively learned, and a model with dense trajectories can provide a more optimal starting point for the learning of sparse trajectories.

9. More Discussions about the Related Works

To further illustrate the advantages of our proposed MotionCtrl, we've conducted a comparative analysis with previous related works. The comparisons are detailed in Table 4. Models such as AnimateDiff[8] (refers to the motion control LoRA models provided by AnimateDiff), Tune-a-video[26], LAMP[27], and MotionDirector[30] implement motion control by extracting motion from one or multiple template videos. This approach necessitates the training of distinct models for each template video or template video set. Moreover, the motions these methods learned are solely determined by the template video(s), and they fail to differentiate between camera motion and object motion. Similarly, MotionDirector[30] and VideoComposer[25], despite achieving motion control with a unified model guided by motion vectors and trajectories respectively, do not distinguish between camera motion and object motion. In contrast, our proposed MotionCtrl, utilizing a unified model, can independently and flexibly control a wide range of camera and object motions in the generated videos. This is achieved by guiding the model with camera poses and trajectories respectively, offering a more fine-grained control over the video generation process.

10. Details of Evaluation Datasets

In this paper, we construct two evaluation datasets to independently evaluate the efficacy of our proposed MotionCtrl on camera motion control and object motion control.

Camera motion control evaluation dataset. This dataset contains a total of **280** samples. It consists of 8 basic camera poses, 20 complex camera poses, and 10 text prompts. The basic camera poses include pan left, pan right, pan up, pan down, zoom in, zoom out, anticlockwise rotation, and clockwise rotation. The complex camera poses are derived from the testset of Realestate10K [33]. Visualizations of these poses are provided in Figure 15, demonstrating the dataset's diverse range of camera poses. Besides, The text prompts are listed below:

- A train travels along a railway through a valley.
- Rocky coastline with crashing waves.
- Effiel Tower in Paris.
- City of Venice, with buildings, river, and boats.
- A pyramid in the desert.
- A castle in the forest.
- A villa in a garden.
- A fish is swimming in the aquarium tank.
- A zebra next to a river.
- A massive, multi-tiered elven palace adorned with flowing waterfalls, its cascades forming staircases between ethereal realms.

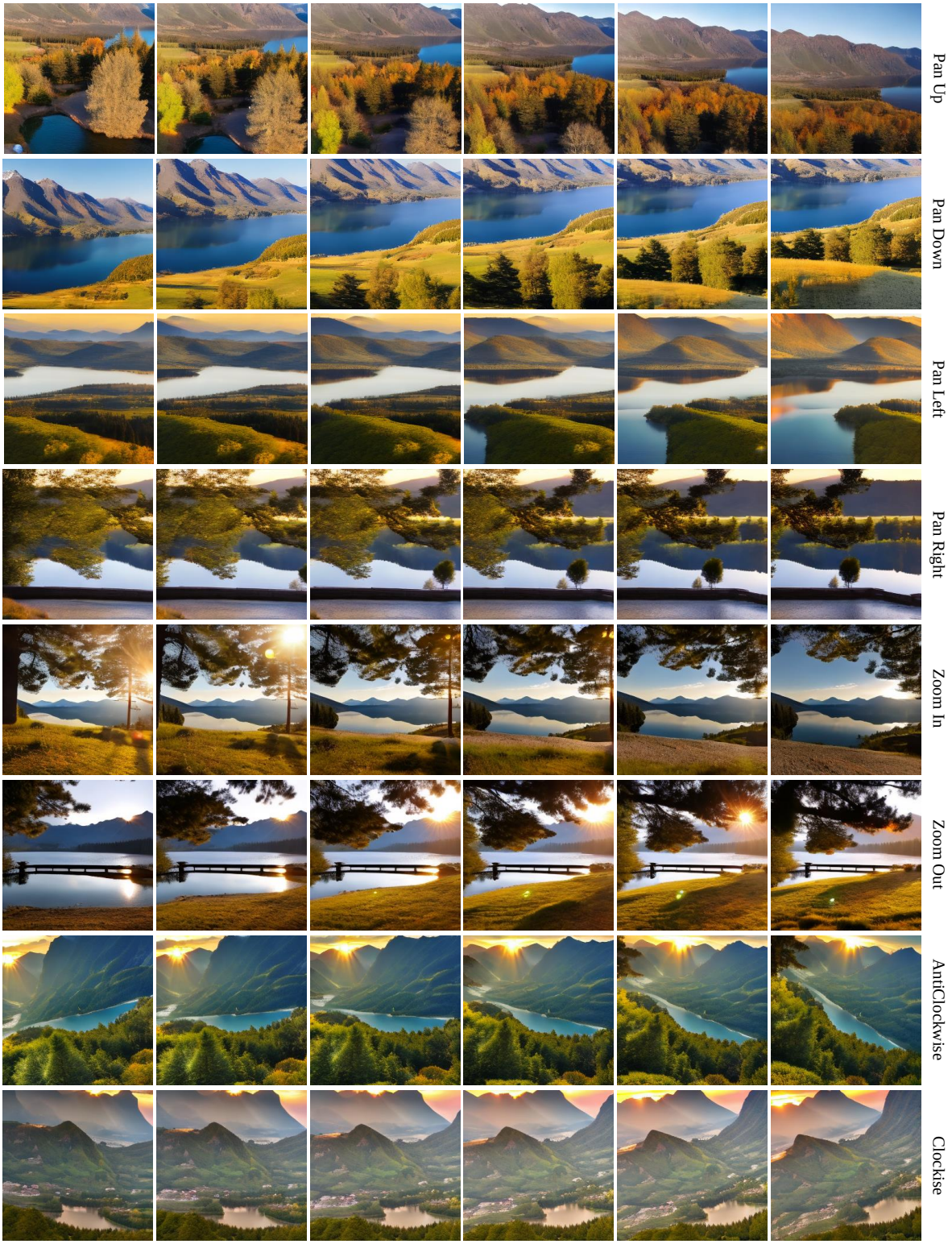
Object Motion Control Evaluation Dataset. This evaluation dataset consists of 19 diverse trajectories and 12 text prompts. The trajectories, which include straight lines, curves, shaking lines, and combinations thereof, are illustrated in Figure 16. The text prompts are provided below:

- A man skateboarding
- A man surfing.
- A girl skiing.
- The rose swaying in the wind.
- The sunflower swaying in the wind.
- Two zebras walking on the grass.
- Two horses walking on the grass.
- A train passing by the valley.
- A car running on Mars.
- A horse running on Mars.
- A cute dog sitting on the green grass.
- A feather floating in the air.

Please note that the evaluation datasets we have constructed are primarily used for quantitatively assessing the performance of our proposed MotionCtrl in both camera and object motion control in video generation. Our MotionCtrl is capable of handling a wider variety of camera poses and trajectories that are not included in the evaluation datasets.

Method	Require Fine-tuning	Motion sources	Distinguish Camera & Object Motion
AnimateDiff [8]	✓	template videos	✗
Tune-a-video [26]	✓	template video	✗
LAMP [27]	✓	template videos	✗
MotionDirector [30]	✓	template videos	✗
VideoComposer [25]	✗	motion vectors	✗
DragNUWA [28]	✗	trajectories	✗
MotionCtrl (Ours)	✗	camera poses & trajectories	✓

Table 4. **Differences between our proposed MotionCtrl and related works**. Unlike AnimateDiff [8] (which refers to the motion control LoRA model provided by AnimateDiff), Tune-a-video [26], LAMP [27], and MotionDirector [30] that implement motion control by extracting motion from one or a series of template videos and require different models for different template videos, our proposed MotionCtrl uses a unified model. Besides, the motions learned by these methods are determined by the template video(s) and they do not distinguish between camera motion and object motion. On the other hand, although MotionDirector [30] and VideoComposer [25] achieve motion control with a unified model guided by motion vectors and trajectories, respectively, they also do not distinguish between camera motion and object motion. In contrast, our proposed MotionCtrl, with a unified model, can independently and flexibly control the camera motion and object motion of the generated video, guided by camera poses and trajectories, respectively.



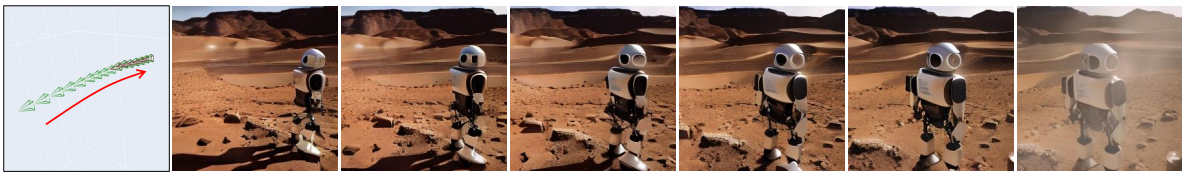
Prompt: A landscape with mountains and lake at sunrise.

Figure 5. The results of our proposed MotionCtrl deployed on LVDM [9], guided by 8 basic camera poses: pan up, pan down, pan left, pan right, zoom in, zoom out, anticlockwise rotation, and clockwise rotation (The visualization of these camera poses can be seen in Figure 15 (a)). **It's important to note that all results are achieved using the same MotionCtrl model, without the need for extra fine-tuning for different camera poses.**

Prompt: A cute cat lying on the floor.

Prompt: A temple on the mountain.

Prompt: A temple on the mountain.



Prompt: A human robot standing on Mars.

Figure 6. The results of our proposed MotionCtrl deployed on LVDM [9], guided by relatively complex camera poses. **Unlike basic camera poses, which only involve simple directional movements, these complex camera poses incorporate elements of camera turning or self-rotation within the same camera pose sequence.** The camera motion in the generated videos closely follows the guided camera poses, while the generated content aligns with the text prompts. For additional results, please visit our project page.

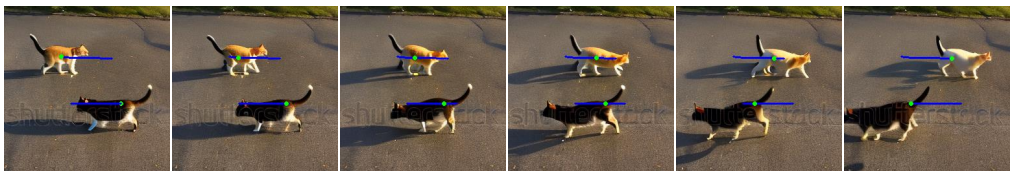
Prompt: A chime in the wind.

Prompt: A sunflower in the wind.

Prompt: A paper plane floating in the sky.

Prompt: A leaf floating in the sky.

Prompt: Two zebras.



Prompt: Two cats.

Figure 7. The result of our proposed MotionCtrl deployed on LVDM [9], guided with trajectories. The green points in the trajectories indicate the starting points. Given the same trajectories, our model can generate different objects in accordance with the text prompts, maintaining the same object motion. When multiple trajectories are present in the same video, our model is capable of simultaneously controlling the motion of different objects within the same generated video.

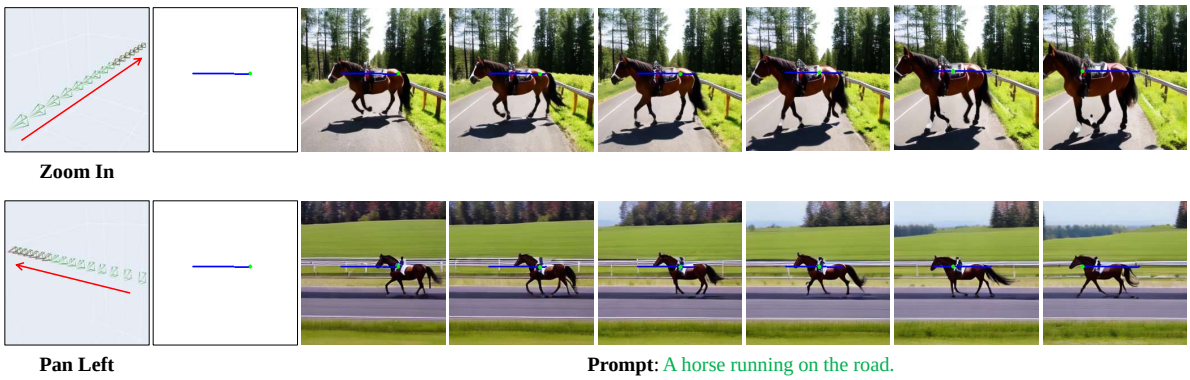


Figure 8. The result of combining camera motion and object motion control of MotionCtrl deployed on LVDM [9]. With the same trajectory but different camera poses, the horse in the generated videos has a different performance.

Figure 9. The camera motion control results of MotionCtrl deployed on AnimateDiff [8]. They are guided with relatively complex camera poses.

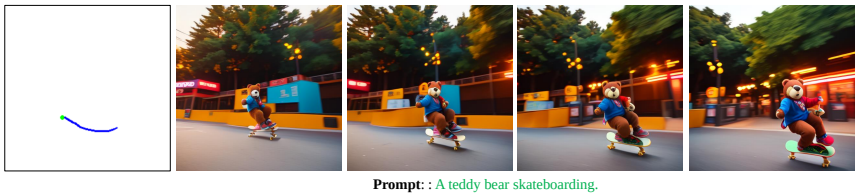


Figure 10. The object motion control results of MotionCtrl deployed on AnimateDiff [8].

Prompt: A teddy bear in the supermarket.

Figure 11. The camera motion control results of MotionCtrl deployed on AnimateDiff [8]. They are guided with 8 basic camera poses.

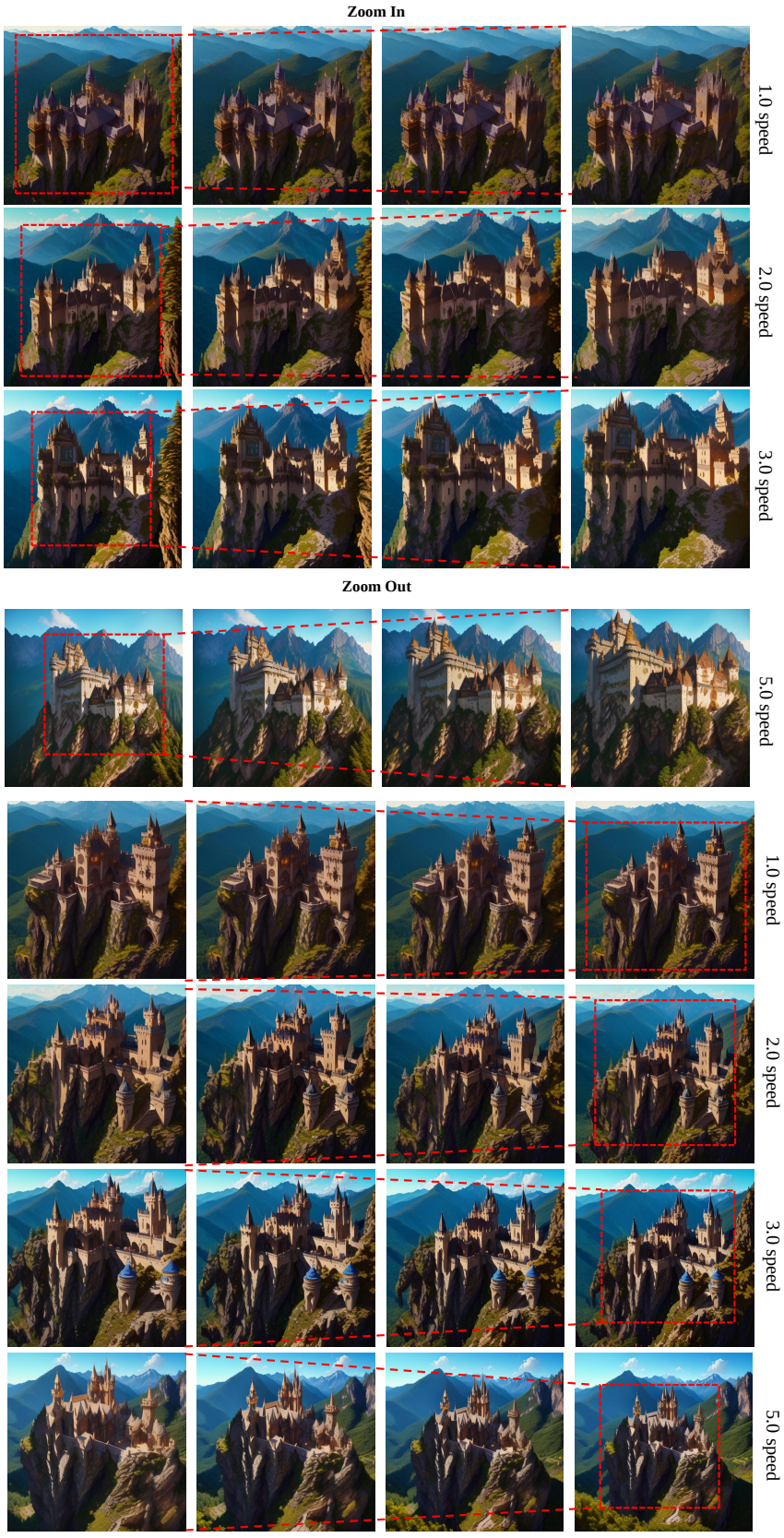
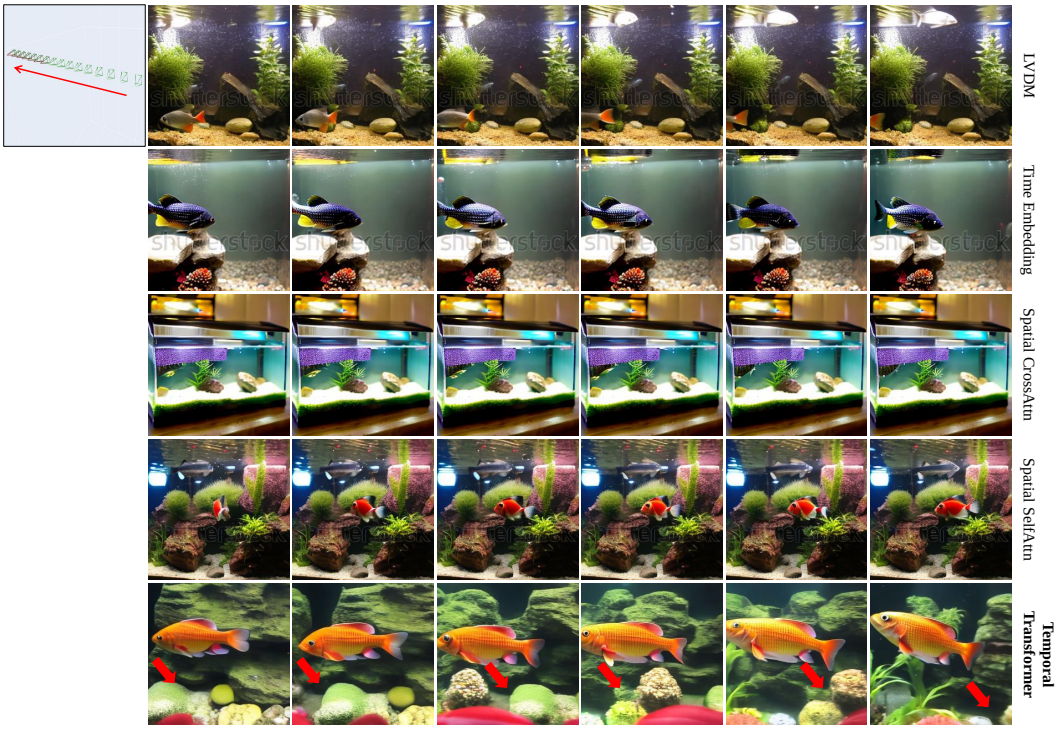


Figure 12. The camera motion control results of MotionCtrl deployed on AnimateDiff [8]. Our MotionCtrl can not only control the camera motion of the generated videos but also their motion speed.

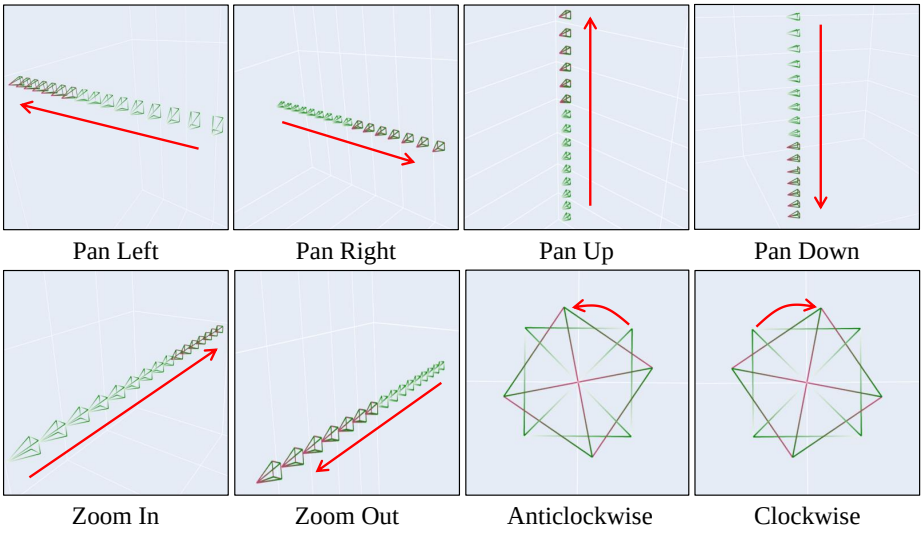


Prompt: A fish is swimming in the aquarium tank.

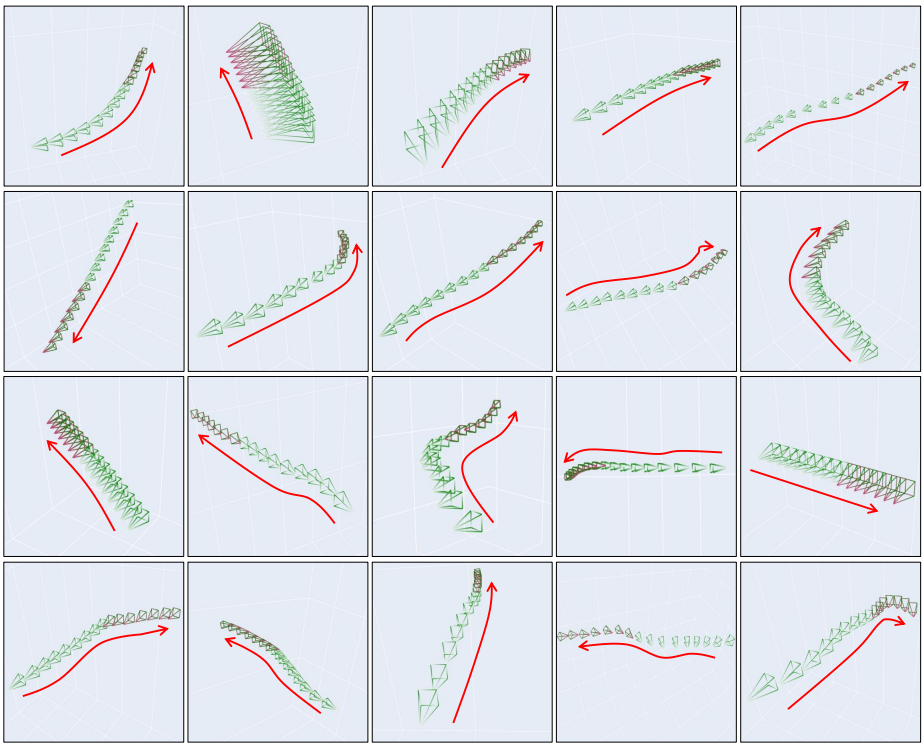
Figure 13. **The qualitative results of ablation study regarding the integrated position of the Camera Motion Control Module (CMCM) with LVDM [9].** Integrating CMCM of MotionCtrl with the temporal transformers in LVDM significantly improves camera motion control compared to other setups.

Prompt: A man is surfing.

Figure 14. **The qualitative results of ablation study "Dense Trajectories v.s. Sparse Trajectories"**. The model trained with dense trajectories fails to control the object motion in the generated video. Conversely, the model trained on dense trajectories, followed by fine-tuning on sparse trajectories, exhibits superior precision in object motion control compared to the model trained solely on sparse trajectories.



(a) 8 Basic Camera Poses



(b) 20 Complex Camera Poses from Realestate10K Testset

Figure 15. The **Camera Motion Control Evaluation Dataset** consists of 8 basic camera poses and 20 complex camera poses, with the complex poses being derived from the test set of Realestate10K. This dataset is utilized to quantitatively assess the effectiveness of our proposed MotionCtrl in controlling a wide range of diverse camera motions in videos generated.

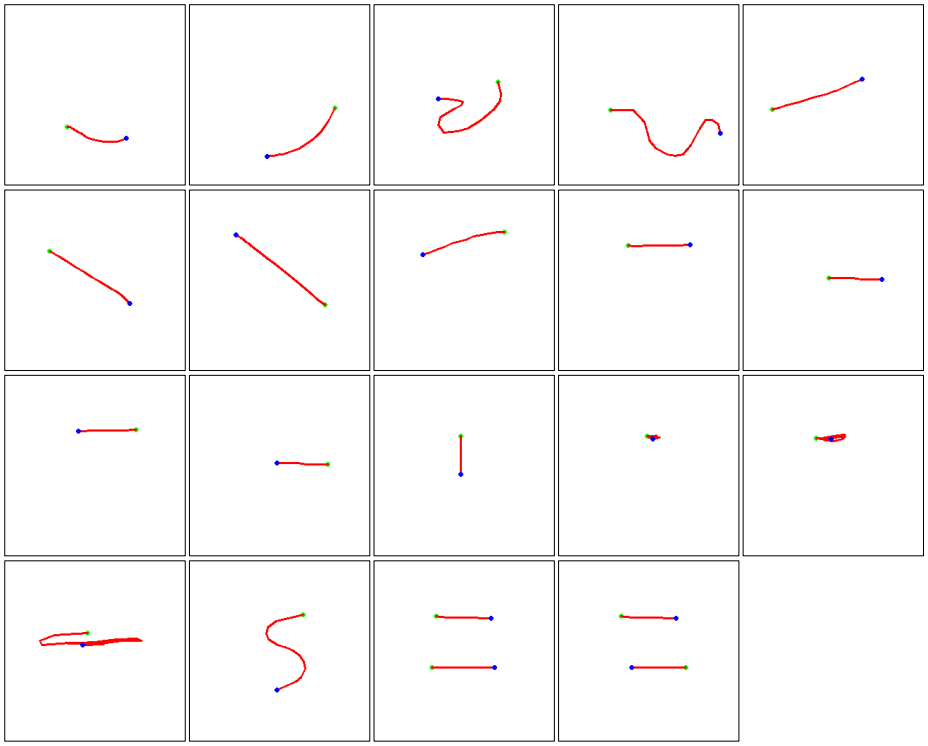


Figure 16. The **Object Motion Control Evaluation Dataset** encompasses 19 trajectories, where the green and blue points respectively represent the starting and ending points of each trajectory. This dataset is used to quantitatively evaluate the effectiveness of the proposed MotionCtrl in controlling object movements in videos generated.