A Rendering Training Videos Using Unity

In this work, all training videos are synthesized using Unity. As introduced in Sec. 4.2, for each camera motion synthesis, we first prepare the scene and then render the video with and without camera motion.

Scene creation. Each scene consists of a background, a floor texture, and objects. These elements are randomly determined during scene creation. Specifically, we first randomly select the categories of the background and floor texture. The background options include {"sky", "far mountains", "closer mountains", "both mountains"}. The "sky" refers to the default background in Unity. The "far mountains" and "closer mountains" are publicly available background assets that depict mountains at different distances, respectively. The "both mountains" option includes both of the previous mountain backgrounds. The floor texture options include "brick and stone floor", "black sand ground", "green grassland", "brown ground", "yellow grassland", "light green grassland". These textures are also publicly available assets in Unity. For the objects, we randomly place both static and moving objects in the scene. The static objects include "tree", "bush", "grass", while the moving objects include "sphere", "cube", "polygon", "cylinder". Notably, all objects are created using basic geometric shapes and do not require specific human effort for design. Please refer to the example training videos on our anonymous website `https://anonymousers196.github.io/VividCamDemo/` for a better understanding of their visual appearance.

Video rendering. After creating the scene, we render the videos both with and without camera motion. The videos without camera motion are generated by randomly determining the camera's coordinates and pose, then fixing the camera in place while recording the video. For the videos with camera motion, we first define the camera movements using a short script (typically no more than 10 lines of code). Based on this script, the rendered video incorporates the specified camera motions. We note that the camera motion script can be effectively written by GPT given natural language instructions (*e.g.,* "I want to write a Unity C# code depicting a camera first push forward, then truck left.")

B Details of Text Prompts

Our training and inference processes rely on different categories of text prompts. In this section, we provide a detailed discussion of the prompts used. Generally, two categories of prompts are employed: (1) scene-only prompts $\boldsymbol{c}$, used for appearance LoRA learning, and (2) composite prompts $(\boldsymbol{c}_m \oplus \boldsymbol{c})$, which combine camera instructions with scene descriptions for learning camera control in the text-based setting. Additionally, we provide examples of prompts used during inference.

Scene-Only Prompts for Appearance LoRA Training. During appearance LoRA training, we constrain the LoRA to learn only the appearance style. Therefore, the training prompt at this stage includes only a description of the rendered scene, specifying objects and environmental details. For example: *"Content: There are small plants and geometries on the light green grassland."* Additionally, as described in Sec. 4.3, we incorporate a style-aligned prompt to help bridge domain gaps during appearance LoRA training. This prompt acts as a virtual indicator of the target style. With this addition, the complete training prompt $\boldsymbol{c}$ becomes, for example: *"Content: In this low-poly 3D <VIRTUAL> scene, there are small plants and geometries on the light green grassland."*

Composite Prompts for Text-Based Camera Control. For text-based camera control, we freeze the appearance LoRA and train a separate camera LoRA. At this stage, the training prompt includes both camera movement instructions $\boldsymbol{c}_m$ and scene descriptions $\boldsymbol{c}$. The camera component guides the camera LoRA to learn appropriate motion patterns, while the scene description ensures consistent content generation. For example: *"Camera: The camera pushes forward, focusing on a moving sphere. Then the camera trucks left. | Content: In this low-poly 3D <VIRTUAL> scene, there is a moving sphere. There are also small plants and geometries on the black sand ground."* It is worth noting that for trajectory-based camera control, we use only the scene description $\boldsymbol{c}$, rather than composite prompts $(\boldsymbol{c}_m \oplus \boldsymbol{c})$, since the camera condition is provided directly by the trajectory input $\boldsymbol{p}$.

Prompts at Inference Time. During inference, we use prompts similar to those employed during camera control training, with the exception that the virtual style indicator is omitted. Below are example prompts for both text-based and trajectory-based control: Text-based: *"Camera: The*

		Value
Appearance LoRA	Learning rate	1e-4
	Rank	128
	Scheduler	Cosine with Restarts
	Warm up steps	400
	Optimizer	adamw
	β_1	0.9
	β_2	0.95
Camera LoRA	Learning rate	3e-4
	Rank	512
	Scheduler	Cosine with Restarts
	Warm up steps	400
	Optimizer	adamw
	β_1	0.9
	β_2	0.95
Trajectory Encoder	Learning rate	1e-4
	Scheduler	Cosine with Restarts
	Warm up steps	250
	Optimizer	adamw
	β_1	0.9
	β_2	0.95

Table 6: Hyperparameter settings.

camera pushes forward, focusing on a static steaming coffee cup. Then the camera trucks right. | *Content: A steaming coffee cup rests on a wooden table beside a stack of books and a pair of glasses."*
Trajectory-based: *"Content: A steaming coffee cup rests on a wooden table beside a stack of books and a pair of glasses."*

C Implementation Details

For all experiments, we use `CogVideoX-5B` (Yang et al., 2024b) as the base model. The base model remains frozen throughout all experiments, and we adopt its default hyperparameters (e.g., noise sampling schedule, conditional guidance scale). Each generated video is 5 seconds long, consisting of 49 frames at a resolution of 720 $\times$ 480. For *text-based control*, the learning rate for LoRA optimization is set to 1e-4 for appearance learning and 3e-4 for camera motion learning, with the LoRA rank fixed at 128. We use one camera LoRA for different motion types, each using 500 synthetic training videos. For *trajectory-based control*, we fine-tune from the pre-trained `AC3D` model using a learning rate of 1e-4. We train one encoder for different motion types, using the same set of synthetic training videos as in text-based control.

To help reproduce our results, we report the detailed hyperparameter settings in Table 6.

D Qualitative Comparison and Analysis

Section 5.2 presents the qualitative results of the text-based methods (ViVidCam-Cog). In this section, we first present the qualitative results of the trajectory-based methods (ViVidCam-AC3D), followed by a comparison with the baseline methods and corresponding analyses.

Qualitative results of trajectory-based methods. We present the qualitative results of ViVidCam-AC3D in Figure 6. As shown, similar to the text-based method, our trajectory-based method can generate videos with precise camera control and high visual quality across a range of camera motions, from simple and composed to complex ones.

Qualitative comparison with baselines. We present qualitative comparisons in Figure 7 and Figure 8. Our observations indicate that state-of-the-art methods struggle to accurately synthesize

Camera: The camera **tilts down, revealing** a snow globe.
Content: A snow globe depicting a small town is surrounded by holiday decorations.

Camera: The camera **pulls back**, moving away from a static steaming coffee cup. Then the camera **trucks left**.
Content: A steaming coffee cup rests on a glass table beside a plate of croissants and an open newspaper.

Camera: The camera **pushes forward**, focusing on a static microscope. Then, the camera **trucks left**.
Content: A laboratory microscope sits on a counter, flanked by glass slides, test tubes, and an open science book.

Camera: The camera **pans around, searching** for a static bouquet of wildflowers. Upon finding it, the camera **locks focus** and **pushes in** on the petals. **Content**: A bouquet of wildflowers rests in a jar on a picnic table, with a red-checkered cloth beside it.

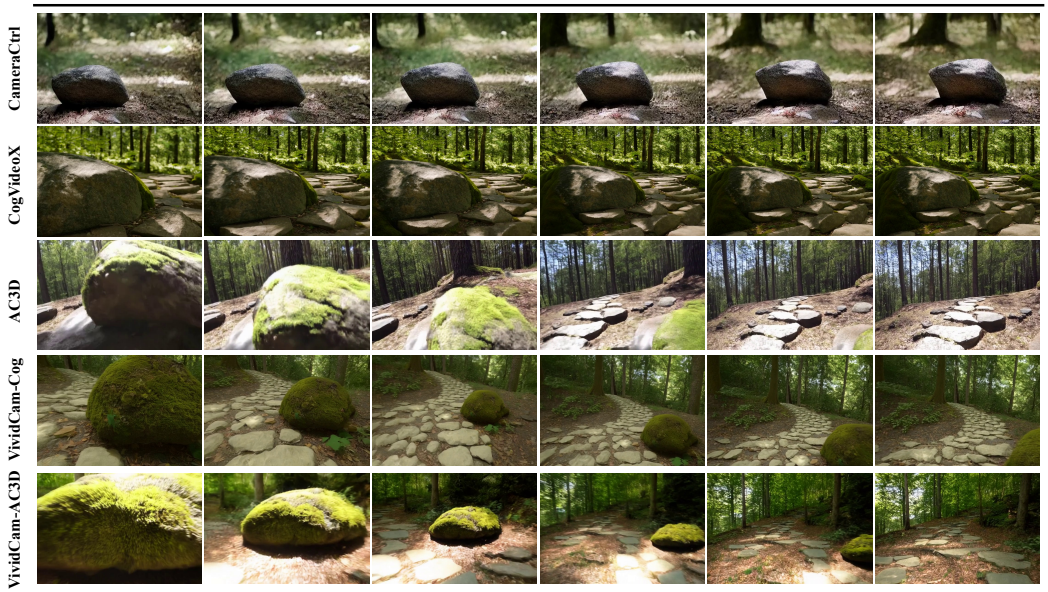
Camera: The camera **pans around**, **searching** for a glowing lantern. Upon finding it, the camera **locks focus** and **pushes in** on the warm light inside. **Content**: A lantern glows on a cobblestone path at dusk, surrounded by fallen leaves and a faint mist.

Camera: The camera **pulls out** from a moss-covered rock. Then, the camera **zooms in, shifting focus** to a stone pathway**.**
Content: The rock rests under the shade of tall trees, while the pathway winds through the forest, its stones carefully placed.

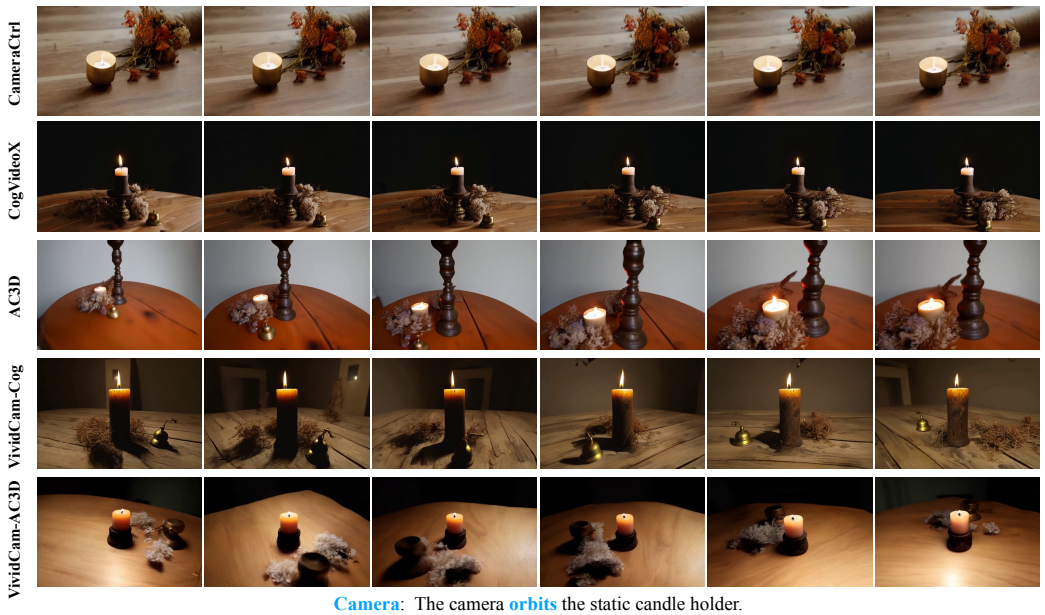
Camera: The camera orbits the static globe.
Content: A classic globe with detailed continents rests on a wooden stand, surrounded by history books.

Figure 6: Qualitative results of diverse camera motions using VIVIDCAM-AC3D. Note that some complex camera motions are difficult to demonstrate through images; please refer to the videos on our anonymous webpage `https://anonymousers196.github.io/VividCamDemo/` for better visual results.

Figure 7: Qualitative results comparison. We observe that camera motions such as "panning around to search for an object, then pushing in to focus on the object" are particularly challenging for state-of-the-art models. Even when provided with exact trajectories, these methods often degrade into simpler camera motions—such as a rightward truck in CAMERACTRL or a turbulent push-in in AC3D. In contrast, our method faithfully produces the intended camera motions. Additionally, we note that certain effects, such as explosive camera motions, are difficult to convey through static images. Please refer to the videos on our anonymous webpage `https://anonymousers196.github.io/VividCamDemo/` for better visual results.



Camera: The camera **pulls out** from a moss-covered rock. Then, the camera **zooms in, shifting focus** to a stone pathway.
Content: The rock rests under the shade of tall trees, while the pathway winds through the forest, its stones carefully placed.



Camera: The camera **orbits** the static candle holder.
Content: A rustic candle holder with a flickering candle rests on a wooden table, accompanied by dried flowers and a brass bell.

Figure 8: Qualitative results comparison. We observe that camera motions such as "pulling out from an object, then zooming in to shift focus to another object" are particularly challenging for state-of-the-art models. Even when provided with exact trajectories, these methods often fail to accurately reproduce the desired camera motions. For example, while AC3D attempts to depict a focus shift from a rock to a stone, it does not successfully demonstrate the pull-out from the rock followed by the push-in toward the road. In contrast, our method faithfully captures and reproduces the intended camera motions. Additionally, we note that such unconventional camera movements are difficult to fully appreciate through static images alone. Please refer to the videos on our anonymous webpage `https://anonymousers196.github.io/VividCamDemo/` for better visual results.

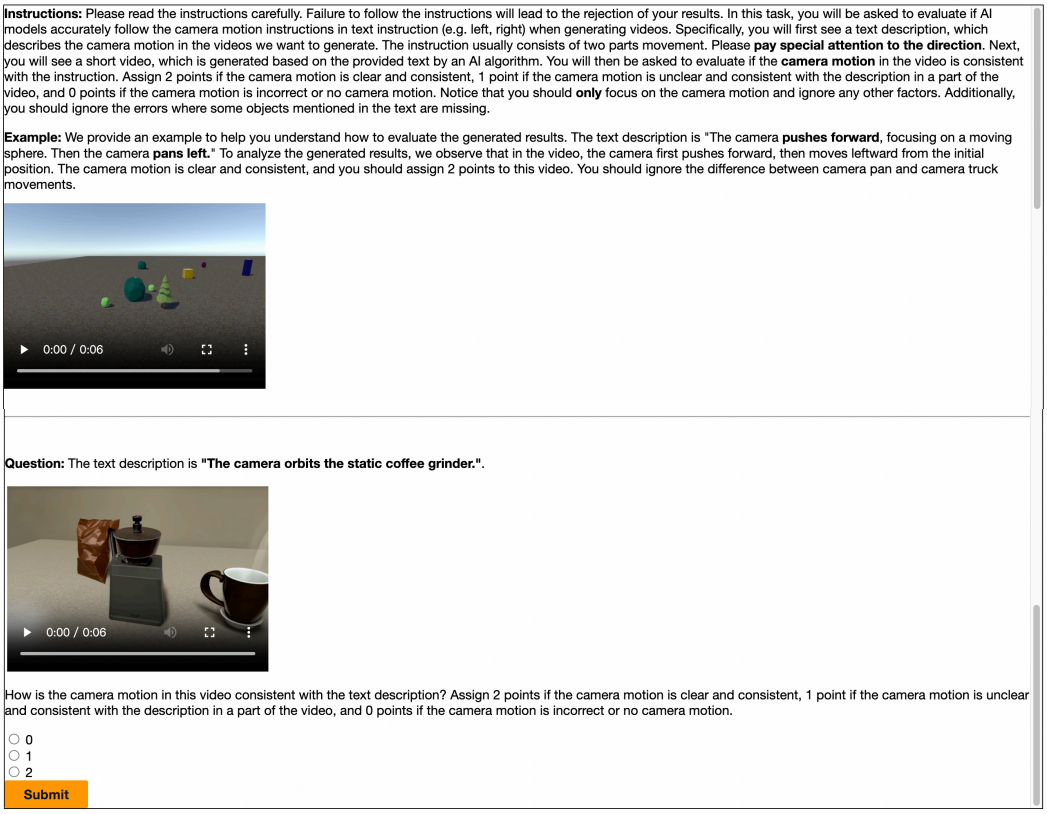


Figure 9: The example interface of Amazon Mechinical Turk in our human study.

unconventional camera motions. For instance, in the upper panel of Figure 7, even when provided with exact trajectories, these methods often simplify the intended motion—resulting in a pan to the right in CAMERACTRL or a turbulent push-in in AC3D. Similarly, in the upper panel of Figure 8, while AC3D attempts to depict a focus shift from a rock to a stone, it fails to effectively illustrate the pull-out from the rock followed by a push-in toward the road. In contrast, our method faithfully captures and reproduces the intended camera motions. Additionally, we note that such unconventional motions are difficult to fully appreciate through static images alone. We encourage readers to refer to the videos on our anonymous webpage `https://anonymousers196.github.io/VividCamDemo/` for better visual results.

E DETAILS OF HUMAN STUDY

Our human study is conducted on Amazon Mechanical Turk. We consider three levels of camera motion: *simple*, *composed*, and *complex*. Please refer to Table 1 for the specific camera motions covered in each category. For each category, we sample 25 prompts and input them into our model and baseline models for evaluation. Each of the 25 prompts is tested twice, resulting in 50 videos per camera motion category and a total of 150 videos. Each question is awarded $0.03. In total, 88 unique workers participate in the study. For each question, we present the tested videos along with the input text prompt and ask participants to answer two types of questions: ❶ (**Action Correctness**) How consistent is the camera motion in the video with the text description? ❷ (**Realism**) How is the visual quality of the video? Participants rate each question on a scale from 0 to 2. To ensure precise evaluation, we provide detailed explanations, a scoring rubric, and examples.

Figure 9 shows an example of the interface that participants will see during the human study.

F CLIP SIMILARITY

	Simple	Composed	Complex
CameraCtrl	0.3254	0.3306	0.3006
CogVideoX	0.3272	0.3215	0.3070
AC3D	0.3397	0.3437	0.3153
ViVidCam-Cog	0.3327	0.3362	0.3230
ViVidCam-AC3D	0.3352	0.3341	0.3134

Table 7: Results on CLIP similarity.

We present the CLIP similarity results in Table 7. Overall, all methods achieve comparable CLIP similarity scores. Specifically, both ViVidCam-Cog and ViVidCam-AC3D exhibit less than a 0.01 difference in CLIP score compared to their vanilla counterparts, CogVideoX and AC3D, respectively. This indicates that our methods maintain the same level of alignment with the desired content described in the text prompts. We also observe that CogVideoX performs worse in scenarios involving complex camera motions, possibly due to quality degradation when generating motion patterns that are likely underrepresented in the training dataset, as shown in Sec. D.