LayerAnimate: Layer-level Control for Animation

Yuxue Yang[1,2] Lue Fan[2] Zuzeng Lin[3] Feng Wang[4] Zhaoxiang Zhang[1,2,5†]
[1]School of Artificial Intelligence, University of Chinese Academy of Sciences
[2]NLPR & MAIS, Institute of Automation, Chinese Academy of Science
[3]Tianjin University [4]CreateAI [5]Shanghai Artificial Intelligence Laboratory
{yangyuxue2023, lue.fan, zhaoxiang.zhang}@ia.ac.cn
linzuzeng@tju.edu.cn feng.wff@gmail.com

Abstract

Traditional animation production decomposes visual elements into discrete layers to enable independent processing for sketching, refining, coloring, and in-betweening. Existing anime generation video methods typically treat animation as a distinct data domain different from real-world videos, lacking fine-grained control at the layer level. To bridge this gap, we introduce ***LayerAnimate****, a novel video diffusion framework with layer-aware architecture that empowers the manipulation of layers through layer-level controls. The development of a layer-aware framework faces a significant data scarcity challenge due to the commercial sensitivity of professional animation assets. To address the limitation, we propose a data curation pipeline featuring Automated Element Segmentation and Motion-based Hierarchical Merging. Through quantitative and qualitative comparisons, and user study, we demonstrate that LayerAnimate outperforms current methods in terms of animation quality, control precision, and usability, making it an effective tool for both professional animators and amateur enthusiasts. This framework opens up new possibilities for layer-level animation applications and creative flexibility. Our code is available at https://layeranimate.github.io.*

1. Introduction

Animation is a globally beloved art form, yet its production remains a complex process involving sketch drafting, refining, coloring, and in-betweening. With the development of video generation models, automation technologies are increasingly being integrated into the animation production process. Recent animation generation models [15, 24, 32, 35] have adapted real-world generation models [1, 37] to achieve impressive results in interpolation and sketch coloring. However, previous works typically treat animation as a distinct data domain compared to real-world videos, generating videos under frame-level controls. They overlook a fundamental concept to animation, *layer*, which allows independent controls on decomposed elements as depicted in Fig. 1. The principle of layer decomposition forms a foundational methodology across animation history, manifesting as stacked translucent celluloid overlays in classical hand-drawn production and evolving into digital layer hierarchies within modern software. The hierarchical paradigm helps animators conduct nondestructive editing through layer isolation, enabling precise control over individual elements.

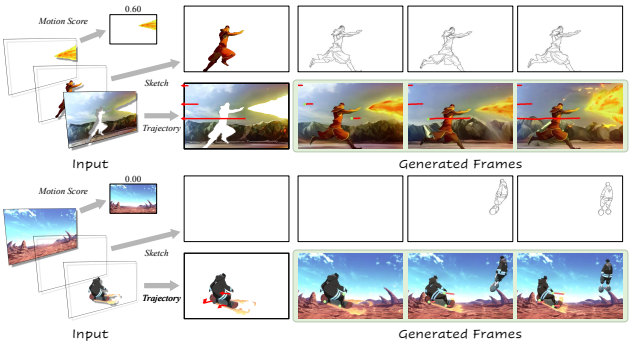

Figure 1. *LayerAnimate* enables controllable video generation under multiple layer-level controls.

Considering the scarcity of layer data due to its commercial value, it is challenging to develop a video generation model supporting layer-level controls. We design a layer curation pipeline comprising Automated Element Segmentation and Motion-based Hierarchical Merging to overcome the challenge. We iteratively leverage SAM [18] and SAM2 [27] for element segmentation, then merge oversegmented elements into layers based on their motion states with hierarchical clustering.

With the curated layer data, we propose *LayerAnimate*, a framework that facilitates flexible composition of heterogeneous layer-level control signals and fine-grained manipulation of animation layers. Initially, the frame-level reference image is decomposed into layer-level regions using layer masks from our curation pipeline, establishing explicit

†Corresponding author

spatial correspondence between layer-level control signals and target regions. Heterogeneous control modalities (e.g., motion scores, sketches, and trajectories) are injected into each layer through dedicated encoders. Following the encoding process, layer-level features are passed into ControlNet [42] branches for independent processing, followed by cross-attention for feature fusion within the UNet. LayerAnimate permits simultaneous manipulation of different elements under composite controls, which is unattainable in conventional frameworks.

We conduct extensive experiments and user studies across various video generation tasks under different conditions, i.e. first-frame Image-to-Video (I2V), I2V with trajectory, I2V with sketch, interpolation, interpolation with trajectory, interpolation with sketch, to demonstrate that LayerAnimate is versatile and superior in terms of animation quality, control precision, and usability. Our contributions are listed as follows.

- We design a layer curation pipeline to automatically extract layer data from animations, addressing the challenge of limited layer data on the Internet.
- We propose a layer-level control framework, *LayerAnimate*, that combines the traditional principle of layer decomposition with modern video generation models to achieve more precise animation control and generation.
- Extensive experimental results demonstrate the effectiveness and versatility of LayerAnimate on various tasks. It also supports innovative layer-level applications, such as a flexible composition of various layer-level controls.

2. Related Works

Video Diffusion Models. Video generation [7–9, 16, 19–21, 26, 39, 40, 46, 47] has experienced significant advancements with the development of diffusion models [5, 11, 29]. Many methods [7, 9, 12, 13, 28, 36] extend text-to-image diffusion architectures to generate temporally coherent videos. However, it remains challenging to convey user intent exclusively through text. To address this, several works [1, 3, 4, 33, 35, 37, 41, 44] incorporate images into diffusion models to enable video generation conditioned on given images. To digest the reference image condition, a common approach used by VideoComposer [33], VideoCrafter [3], and DynamiCrafter [37] is encoding the image through pre-trained CLIP or other well-designed image encoders before feeding it into diffusion models along with text prompts. Furthermore, models like PixelDance [41], SEINE [4], DynamiCrafter [37], and ToonCrafter [35] concatenate two different reference images with noisy frame latents to interpolate images with smooth transitions. However, they fill the intermediate frames with placeholders, which underutilizes the conditions. In contrast, our LayerAnimate assigns layers based on their motion states to the intermediate frame, allowing for injecting motion state.

Controllable Video Generation. Image-to-Video and interpolation models define videos' endpoints but struggle to provide motion information for intermediate frames. Approaches [6, 14, 15, 24, 25, 30, 34, 35, 45] like SparseCtrl [6] and ToonCrafter [35] introduce an auxiliary branch for controllable video generation, inspired by ControlNet [42]. LVCD [15] introduces a sketch-guided ControlNet to facilitate color transfer from the reference image to other frames. However, these methods require frame-level control. When applied to animation, such frame-level control will make regions without signals undergo unpredictable deformation. The most recent work AniDoc [24] facilitates high-quality animation with a reference character and sketch guidance without backgrounds, while it is tailored for characters. Another classic control manner is movement control through trajectories, such as DragAnything [34] and Tora [45]. However, neither of them is adaptable to anime generation. In this paper, our proposed *LayerAnimate* allows users to provide layer-level control signals and supports applying multiple controls simultaneously in a more user-friendly manner for controllable anime generation.

3. Layer Curation

The construction of a well-curated dataset with detailed layer information is a prerequisite for training a layer-level controllable animation generation framework, which remains constrained by two critical challenges. First, the commercial sensitivity of professional animation assets and the high cost of manual annotation make layer data hard to be scalable. Second, unlike real-world video processing where depth estimation facilitates element stratification (e.g., MIMO [23]), the inherent 2D property of animations constrains reliable geometric cues for decomposing a frame into layers. Conventional segmentation models applied to animations typically yield over-segmented color patches that lack semantics and are impractical for manipulation. To address the challenges, we devise a novel layer curation pipeline comprising *Automated Element Segmentation* and *Motion-based Hierarchical Merging*, as illustrated in Fig. 2.

3.1. Automated Element Segmentation

Taking advantage of recent advancements in visual foundation models [18, 27], we develop an iterative segmentation pipeline for automated element extraction in animation clips. The process initiates with uniform temporal sampling at 4-frame intervals to establish Key Frames, where the first Key Frame K_0 is segmented via SAM [18] to generate atomic element masks $\mathcal{M}_0$. These masks then serve as prompts, which are propagated to all F frames in a clip

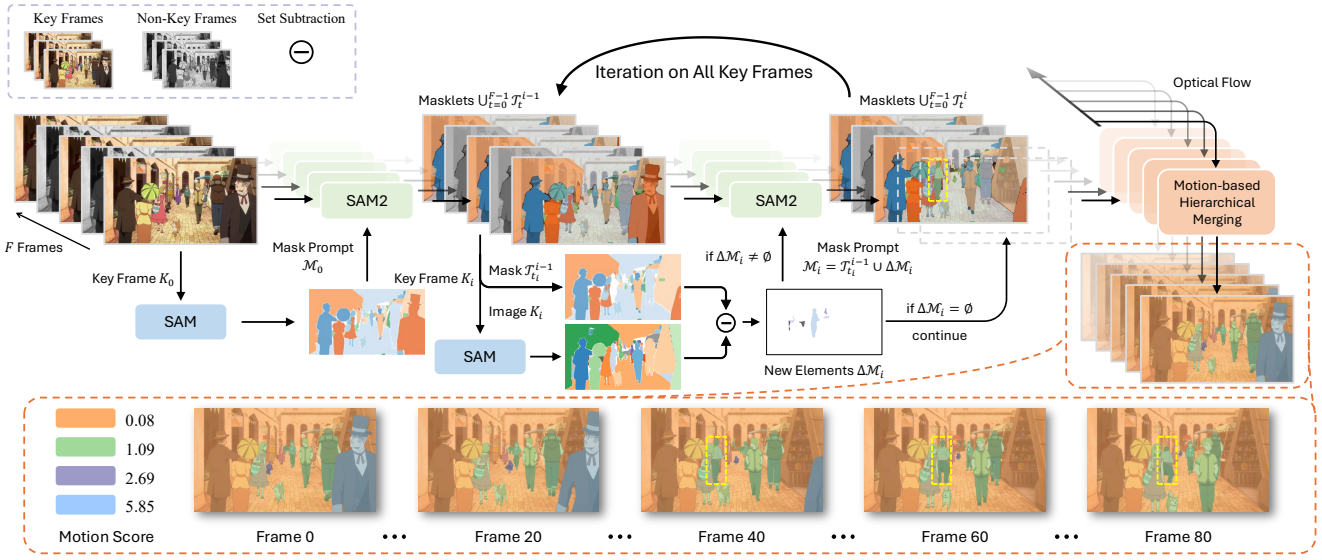

Figure 2. **Layer Curation Pipeline.** The bottom orange dashed box illustrates curated layer masks with different motion scores, where motion scores remain temporally constant throughout the animation clip. Yellow dashed boxes denote new elements absent in the first frame, demonstrating our pipeline's capability to segment dynamically appearing elements. We transparently present some frames of masklets $\bigcup_{t=0}^{F-1}\mathcal{T}_t^i$ to highlight the new elements in Key Frame K_i.

through SAM2 [27], establishing initial masklets with temporal coherence.

Considering the frequent occurrence of new elements appearing in subsequent frames, the initial masklets cannot segment these elements. Thus, we implement an iterative refinement to solve the issue. We first denote the initial masklets as $\bigcup_{t=0}^{F-1}\mathcal{T}_t^{i=0}$, where $\mathcal{T}_t^i$ denotes the refined masks for the t-th frame at the i-th iteration. We detect new elements for each Key Frame K_i with its frame index t_i through mask set subtraction:

$$\Delta\mathcal{M}_i = \text{SAM}(K_i) \setminus \mathcal{T}_{t_i}^{i-1}. \tag{1}$$

Any new elements $\Delta\mathcal{M}_i \neq \emptyset$ will update mask prompts

$$\mathcal{M}_i = \mathcal{T}_{t_i}^{i-1} \cup \Delta\mathcal{M}_i, \tag{2}$$

which is repropagated through SAM2 to obtain refined masklets $\bigcup_{t=0}^{F-1}\mathcal{T}_t^i$. If there is no new element, the masklets at iteration i remain the same as iteration $i-1$. The pipeline's iterative refinement enables coherent element extraction across the temporal dimension, even for the dynamically appearing elements.

3.2. Motion-based Hierarchical Merging

While SAM2 is capable of managing automated element segmentation for animations, it will cause an issue of over-segmentation. This issue arises when regions that should belong to the same layer are divided by inner boundaries, resulting in a large count of elements. If we regard each element as a layer, such over-segmentation breaks semantic objects into granular yet meaningless subdivisions, but also diminishes usability.

To address this, we introduce *Motion-based Hierarchical Merging (MHM)*, designed to merge over-segmented masklets based on their motion states. It is inspired by animation workflow, where animators dynamically merge or separate layers according to their motion states. Firstly, we employ Unimatch [38] to estimate optical flow, computing a motion score for each masklet by averaging flow magnitudes across all pixels in the masklet. Notably, we do not use the direction of flow to represent motion state since pixels may move in diverse directions within a layer, such as dispersing smoke. MHM regards masklets as nodes and constructs a treemap using hierarchical clustering based on motion scores, merging layers with similar motion scores from the bottom up. Considering the variability in layer numbers during production, we do not restrict a fixed number of layers. Instead, we define the maximum layer capacity N, which is much less than the number of masklets, and a motion score merging threshold η_s. Layers are merged from the bottom up until the count of layers falls below the capacity N and the motion score difference exceeds the threshold η_s. The motion score of the final merged layer is set by averaging the motion scores from the merged layers. A simple illustration of Motion-based Hierarchical Merging can be found in Supplementary Material.

4. LayerAnimate

Given a reference image c_{image}, layer masks $\mathbf{M}$, and layer-level control signals, our objective is to generate animation videos from Gaussian noise $\mathbf{z}$ through a conditional denoising network ϵ_θ. Hence, we propose *LayerAnimate*, a framework that enhances fine-grained control over layers within

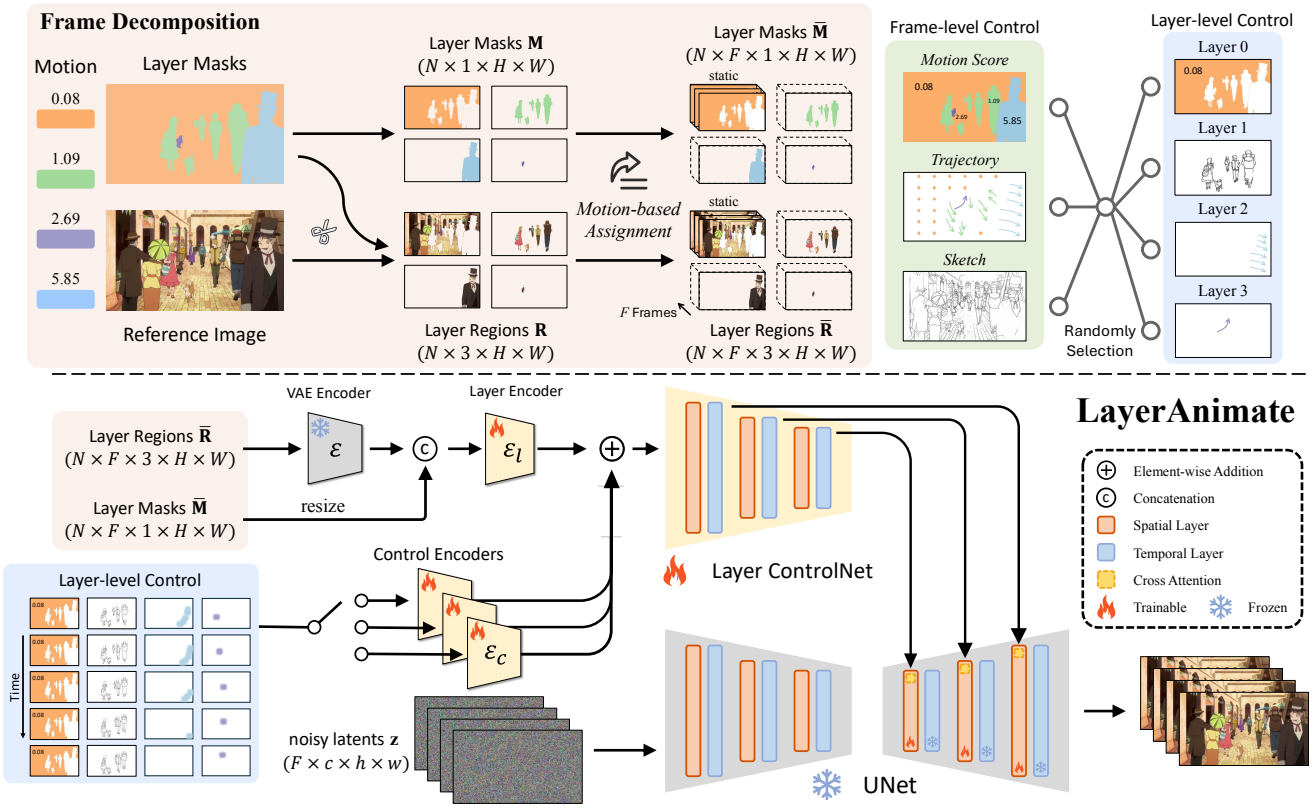

Figure 3. **Overview of *LayerAnimate***. LayerAnimate establishes a layer-level control architecture for animation generation. It enables the flexible composition of control signals at the layer level, allowing for injecting distinct conditions (e.g., motion scores, trajectories, and sketches) for different layers. For simplicity, the text and image injection branches are omitted from the core architecture schematic.

a video diffusion model, as illustrated in Fig. 3.

4.1. Frame Decomposition

To unify the representation of layer information across various videos, we begin with padding the variable number of layer masks $\mathbf{M}$ to the fixed maximum capacity N. We then decompose the reference image c_{image} with binary layer masks $\mathbf{M} \in \mathbb{R}^{N\times1\times H\times W}$ to get layer regions $\mathbf{R} \in \mathbb{R}^{N\times3\times H\times W}$ and indicate layer motion state with their motion scores obtained from Sec. 3.2. With the layer information in hand, we need to consider it for non-reference frames across the temporal dimension.

Some multi-frame control methods, such as SparseCtrl [6] and ToonCrafter [35], employ zero images to imply unconditional frames. Conversely, approaches like SVD [1] and DynamiCrafter [37] that condition on a single reference image replicate the reference across all frames and then concatenate them with the input of the diffusion model. In LayerAnimate, we integrate the aforementioned methods to propose *Motion-based Assignment*. It first categorizes layers into dynamic and static based on motion scores and a predefined threshold η, where the static layers are expected to remain unchanged along the temporal. Specifically, we assign static layers from the reference to all $F-1$ non-reference frames, while assigning zero images to the $F-1$ non-reference frames of dynamic layers, where F is the number of frames in the video. Through the assignment, we unsqueeze layer masks and layer regions from $\mathbf{M} \in \mathbb{R}^{N\times1\times H\times W}, \mathbf{R} \in \mathbb{R}^{N\times3\times H\times W}$ to $\overline{\mathbf{M}} \in \mathbb{R}^{N\times F\times1\times H\times W}, \overline{\mathbf{R}} \in \mathbb{R}^{N\times F\times3\times H\times W}$ in the temporal dimension.

4.2. Layer Controlling

Following frame decomposition, precise control signal injection at the layer level is crucial for layer-level controllable animation generation. Considering user accessibility, we implement three control modalities, which are in ascending order of control information: motion score (scalar fields), trajectory (directional guidance), and layer-level sketch (dense structural priors). During training, layer-level control signals are randomly selected from frame-level signals through layer masks, enabling a flexible composition of control signals. At inference, users can freely decompose the reference frame into layers and apply layer-level controls through an interactive interface.

Motion Score. In Image-to-Video (I2V) task, motion is conventionally depicted by the text prompt; however, it's difficult for users to express precise motion descriptions for

each layer. Besides, certain elements like flames and particle effects, which are challenging to describe using trajectories, are common in animation clips. Hence, we introduce layer-level motion scores to provide a more user-friendly control manner. As detailed in Sec. 3.2, we obtain layer motion scores $\mathbf{s}$ via optical flow estimation. For consistent representation, we define an upper score s_{max} and normalize $\mathbf{s}$ to $[0, 1]$ by $\mathbf{s}' = \lceil \frac{\mathbf{s}}{s_{\text{max}}} \rceil$. The scores $\mathbf{s}' \in \mathbb{R}^{N\times 1}$ are spatially and temporally aligned with layer masks $\overline{\mathbf{M}} \in \mathbb{R}^{N\times F\times 1\times H\times W}$ through broadcasting and concatenated in the channel dimension. Notably, layer masks $\overline{\mathbf{M}}$ only rely on the reference frame, eliminating the requirement for per-frame mask annotations.

Trajectory. Trajectory offers enhanced spatial-temporal controllability compared to scalar motion scores. We implement CoTracker3 [17] to track the 60×60 grid points across animation clips. To filter out low-quality point trajectories wandering across different layers, we enforce constraints using masklets $\bigcup_{t=0}^{F-1} \mathcal{T}_t^i$ from Sec. 3.1. We assign the masklets to each trajectory based on their coordinates in the first frame, then retain those trajectories maintaining more than 80% overlap within the masklet to ensure layer-consistency. The filtered trajectories are converted into a three-channel map, including one channel that indicates a Gaussian Heatmap like DragAnything [34] and the other two channels store a normalized offset map like Tora [45]. This hybrid representation combines the strengths of both forms: the heatmap channel resolves static/dynamic ambiguity in the offset map, i.e., static and uncontrolled regions are both zero, while the offset map models temporal correspondences between heatmap peaks in adjacent frames. As demonstrated in Sec. 5.5, the hybrid scheme achieves better performance.

Sketch. Sketch enables precise manipulation of complex motions with dense structure guidance. Unlike conventional complete frame-level sketch requirements, we randomly select layer-level sketches with curated layer masklets from Sec. 3 and remove the area of other layers to develop the capability of permitting partial sketching.

4.3. Layer Feature Fusion

As illustrated in Fig. 3, the decomposed layer regions $\overline{\mathbf{R}}$ are encoded into latent space by a VAE encoder. To distinguish valid regions from invalid zero values, we resize layer masks $\overline{\mathbf{M}}$ to match the size of layer latents by bilinear interpolation for concatenation and further encode them with the layer encoder ε_l. Layer-level controls are organized into the image format for subsequent encoding. We implement conventional blocks to achieve an 8x spatial compression as control encoders ε_c except for sketch, which is encoded by a VAE Encoder and a trainable convolution layer.

After the layer encoder and control encoders, the encoded layer features are combined and fed into ControlNet for parallel processing at the layer level, i.e., each layer is regarded as an independent sample. Since the processed layer-level features $\mathbb{R}^{N\times F\times c\times h\times w}$ from ControlNet are N times the number of frame-level features $\mathbb{R}^{F\times c\times h\times w}$ in the denoising UNet, we implement cross-attention to fuse layer features, where the frame-level features in UNet act as queries and the layer features serve as keys and values. It's crucial to note that we introduce a validity mask to indicate padded layers, ensuring only valid layers participate in feature fusion.

4.4. Training and Inference

Training. During training, we optimize the conditional denoising network ϵ_θ, which consists of layer encoder ε_l, control encoders ε_c, UNet, and ControlNet. The encoders $\varepsilon_l, \varepsilon_c$, the spatial layer in the decoder of UNet and ControlNet are trainable, while all other parameters are frozen. The objective is given by:

$$\min \mathbb{E}_{\mathbf{z}_0,t,\epsilon\sim\mathcal{N}(0,\mathbf{I})} \left[||\epsilon - \epsilon_\theta(\mathbf{z}_t; c, \overline{\mathbf{R}}, \overline{\mathbf{M}}, \mathbf{L}_\text{c})||_2^2\right], \quad (3)$$

where $\mathbf{z}_0$ represents the initial video latents from VAE encoder, $\mathbf{z}_t$ is the noised video latents at timestep t, $\overline{\mathbf{R}}, \overline{\mathbf{M}}$ denote the layer regions and masks obtained from Sec. 4.1, $\mathbf{L}_\text{c}$ corresponds to layer-level controls, and c indicates other conditions like the reference image c_image and the text prompt c_text. Moreover, we implement random control selection to enhance the model's robustness against diverse conditions. We apply a 10% dropout probability to layer masks simulating incomplete user annotations. For each retained layer, the control among the three modalities will be randomly selected in the following probabilities: 20% for motion score, 40% for trajectory, and 40% for sketch. Since the three modalities are in ascending order of guidance information, the simultaneous application of weaker control does not provide additional guidance when selecting a strong control; therefore, we only select one control for each layer.

Inference During inference, LayerAnimate allows users to generate layer masks on the reference image by simply clicking using SAM [18]. Users can freely input distinct controls for different layers to generate an animation video tailored to the users' specifications.

5. Experiments

5.1. Implementation

During the layer curation phase, we collect a considerable number of raw animation videos, which are systematically cleaned following OpenSora [46]. On this basis, we curate layer data through our layer curation pipeline. Throughout the process, we define the maximum layer capacity as $N = 4$ and set the motion score merging threshold $\eta_s = 1.0$. The pipeline yields a dataset of 665K clips, ranging from 16

to 128 frames per clip, from which 1K clips are randomly selected as the evaluation set.

We adopt the pre-trained UNet from ToonCrafter [35], designed for cartoon interpolation, as our denoising UNet. We replace its specially designed interpolation-oriented VAE with a standard VAE utilized in an I2V model DynamiCrafter [37] in the I2V task. In LayerAnimate, the control encoders ε_c and layer embedding ε_l are implemented with convolutional blocks. During the training, we classify layers with motion scores below $\eta = 0.1$ as static and define the upper score $s_{\text{max}} = 30.0$. The sketches utilized in the experiments are extracted from original videos using the method in [2].

All experiments are conducted over 30,000 steps using AdamW [22] optimizer with a learning rate of 2e-5 on 32 NVIDIA A100 GPUs. The total batch size is set to 96. Our LayerAnimate, with a maximum layer capacity of $N = 4$, is trained to generate 16 frames at a resolution of 320×512 on our collected anime dataset.

5.2. Comparison

To demonstrate the versatility of our model, we conduct comparisons across six video generation tasks under different conditions: first-frame Image-to-Video (I2V), I2V with trajectory, I2V with sketch, interpolation, interpolation with trajectory, and interpolation with sketch. For these tasks, we compare our method against the latest representative state-of-the-art methods: SEINE [4], DynamiCrafter [37], and CogVideoX [40] for I2V and interpolation tasks, DragAnything [34] and Tora [45] for I2V with trajectory task, Framer [32] for interpolation with trajectory task, AniDoc [24] and LVCD [15] for the I2V with sketch task, and ToonCrafter [35] for interpolation and interpolation with sketch tasks.

Discussion. Here we first briefly discuss the core differences between our methods and some related methods. (1) DragAnything [34] assigns trajectories to distinct entities based on masks, which is similar to our concept of layer-level control; however, its control is limited to the simple displacement of entities. (2) AniDoc [24] is tailored for character sketch coloring with the reference character specification. Here, we take the first frame as the reference. (3) Framer [32] enables interpolation with given trajectories, where we provide the trajectories obtained by CoTracker3 [17]. To ensure a fair comparison, we do not input motion scores in I2V and interpolation tasks and adopt the same trajectories and sketches as the counterparts in related tasks.

Quantitative Comparison. To evaluate the quality of the generated videos in both spatial and temporal domains, we employ FVD [31] and FID [10] metrics. Additionally, to assess reconstruction quality in sketch-conditioned tasks, we adopt LPIPS [43], PSNR, and SSIM to measure the similarity between the generated videos and the original videos. As presented in Tab. 1, our method demonstrates superior performance in all tasks. Although we only demonstrate the performance with a certain single control in Tab. 1 for fairness, our approach also allows for the combination of multiple control modalities, indicating that our model possesses greater applicability, which can be seen in Sec. 5.3.

Task		Method	FVD↓	FID↓	LPIPS↓	PSNR↑	SSIM↑
I2V	-	SEINE [4]	236.04	30.14	0.458	13.06	0.465
		DynamiCrafter [37]	114.80	**14.36**	**0.354**	14.89	0.554
		CogVideoX [40]	170.31	19.77	0.355	14.35	0.543
		LayerAnimate (ours)	**87.96**	14.66	0.370	**15.45**	**0.556**
	Traj.	DragAnything [34]	300.51	26.10	0.464	14.00	0.514
		Tora [45]	190.61	22.03	0.376	15.32	0.525
		LayerAnimate (ours)	**72.04**	**12.55**	**0.281**	**17.46**	**0.634**
	Sketch	AniDoc [24]	42.26	12.16	0.131	21.39	0.792
		LVCD [15]	29.85	7.01	0.076	**26.22**	**0.862**
		LayerAnimate (ours)	**26.64**	**5.92**	**0.075**	25.71	0.858
Interpolation	-	SEINE [4]	97.13	11.96	0.267	18.19	0.641
		DynamiCrafter [37]	98.72	13.03	0.282	17.95	0.629
		CogVideoX [40]	88.28	9.93	0.250	19.39	0.684
		ToonCrafter [35]	74.63	9.97	0.244	19.92	0.668
		LayerAnimate (ours)	**59.64**	**8.38**	**0.216**	**20.07**	**0.696**
	Traj.	Framer [32]	62.17	7.67	**0.150**	22.48	0.760
		LayerAnimate (ours)	**44.69**	**6.94**	0.164	**22.50**	**0.764**
	Sketch	ToonCrafter [35]	66.26	8.40	0.128	23.28	0.794
		LayerAnimate (ours)	**15.63**	**3.23**	**0.044**	**29.84**	**0.908**

Table 1. **Quantitative comparison with other state-of-the-art video generation models** across various tasks on our evaluation set. Traj.: Trajectory control.

Qualitative Comparison. Unlike real-world videos, anime videos feature *special effects*, *objects appearing from nowhere*, and *unconventional visual styles*. We select several representative clips for qualitative comparison, as depicted in Fig. 4.

- For I2V, DynamiCrafter struggles to maintain character consistency, CogVideoX does not animate any elements but merely blurs the image, whereas our method not only generates particle effects but also preserves the character's facial consistency after particles pass across it.
- For I2V with trajectories, the flying red mecha controlled by DragAnything disappears halfway through, and the glass canopy of the aircraft not controlled by Tora fails to maintain consistency. Our method exhibits excellent tracking on the movement of the red flying mecha, and enables aircraft to be unchanged through a fixed point trajectory.
- For I2V with sketches, our method generates the knife with luster, exhibiting greater detail.
- For interpolation, our method generates more reasonable arm movements and facial expressions.
- For interpolation with trajectories, our method achieves

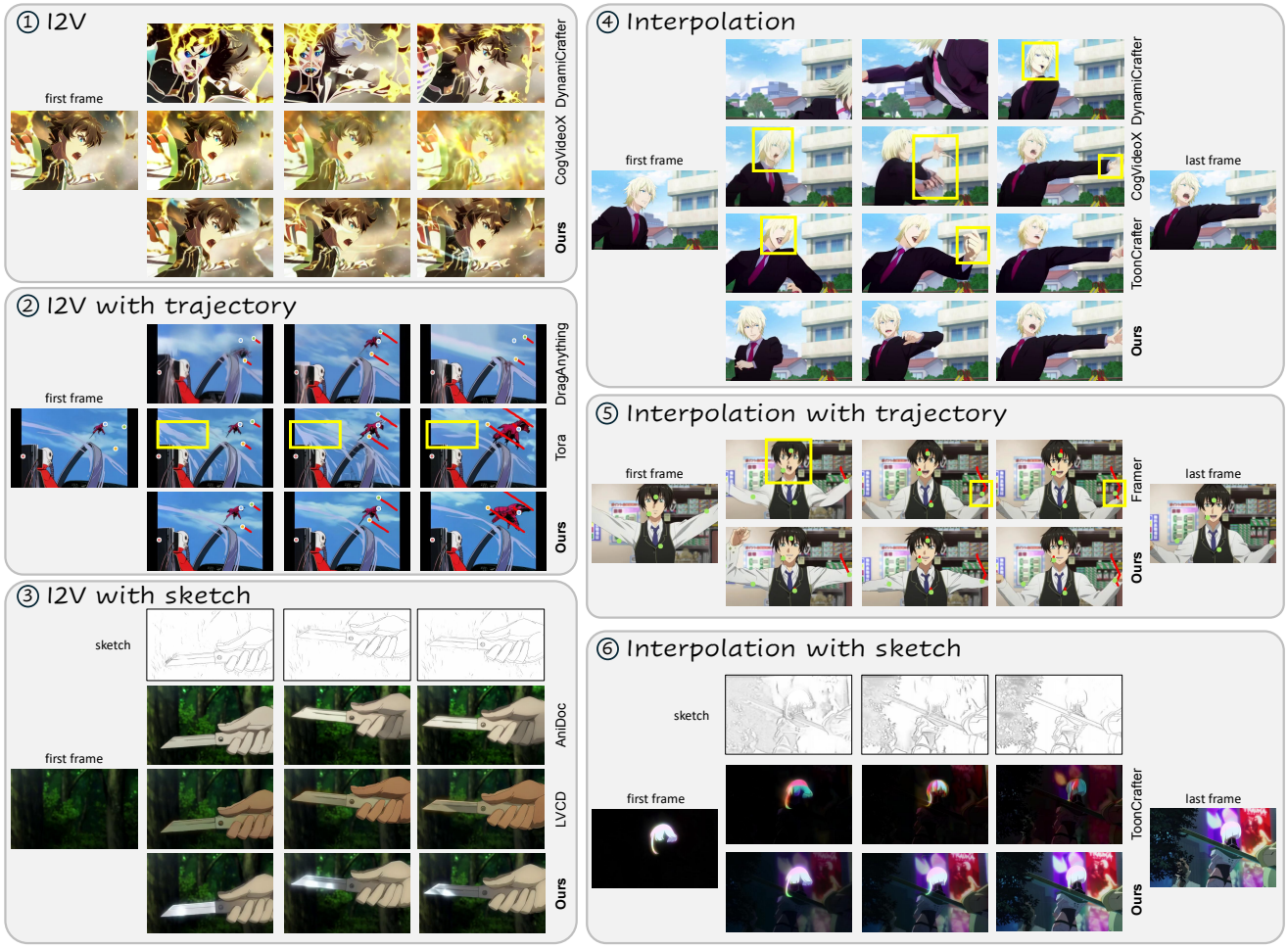

Figure 4. **Qualitative comparison with other competitors.** We select several clips to exemplify the representative characteristics of animation, including particle effects in ①, a knife appearing off-screen ③, and an unconventional fade-in visual style in ⑥. We provide the **corresponding videos** in the supplementary materials, offering more clear and vivid comparisons.

excellent tracking and generates a more natural facial expression than Framer.

- For interpolation with sketches, which involves a fade-in scene, ToonCrafter fails to reveal the background properly, and the character's hair color alters over frames. Our method maintains consistent hair color while accurately generating the intended fade-in visual style.

5.3. Composite Control

Our proposed LayerAnimate enables multiple heterogeneous control over animation layers. Combining the multiple control signs leads to a composite manner of control, as illustrated in Fig. 6. Taking the 4-layer sample as an example (the first row of Fig. 6), we employ sketch for the character layer to depict complex facial expressions while using trajectory movement for the sky and assigning different motion scores to mecha and light effects. Ultimately, this approach enables the generation of animation clips with less cost than conventional frame-level sketching. Furthermore, other samples showcase effects such as the dragging of the luminous shockwave (the 2nd row) and the rotation of stages (the 3rd row).

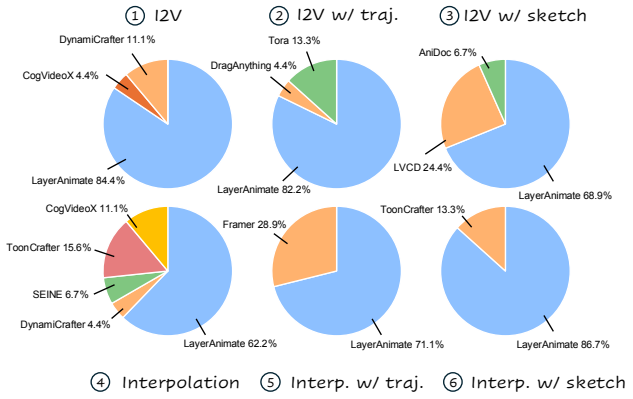

Figure 5. **Voting results of the user study.** LayerAnimate exhibits superior performance across different tasks. Interp.: Interpolation. traj.: trajectory.

5.4. User Study

To further evaluate the effectiveness of our method, we conduct a user study involving 20 participants who voted the

Figure 6. **Composite control**. LayerAnimate provides multiple user-friendly control options at the layer level, leading to a composite control manner. We also provide the **corresponding videos** in the supplementary materials for clear illustration.

best-generated videos among LayerAnimate and other competitors across six different tasks, as discussed in Sec. 5.2. As shown in Fig. 5, our LayerAnimate exhibits superior performance.

5.5. Ablation

Layer Capacity. To investigate the impact of layer capacity N settings on the performance, we test $N = 1, 2, 4$ under I2V with motion scores condition. As can be seen in Tab. 2, increasing N will improve the performance, demonstrating the superiority of our layer-level design. In practice, using 4 layers is adequate in most animation cases, so we select $N = 4$ as the default layer capacity.

Motion Score. To demonstrate the effectiveness of layer-level motion information, here we progressively input motion information from binary motion state to specific motion score. The binary motion state (i.e., w/ MA in Tab. 2) means a certain layer is either static or dynamic. Specific motion score provides more detailed information indicating the degree of movement, demonstrated by "w/ MA & scores" in Tab. 2. As showcased in Tab. 2, more motion information enables better generation quality in both I2V task and interpolation task.

Trajectory Representation. For the representation of the trajectory, we integrate the commonly used Gaussian Heatmap and offset map forms in Sec. 4.2. As demonstrated in Tab. 2, our design results in a significant performance enhancement.

	Method	FVD↓	FID↓	LPIPS ↓	PSNR↑	SSIM↑
Capacity	I2V (N = 1)	87.88	14.63	0.376	15.05	0.546
	I2V (N = 2)	81.93	14.15	0.363	15.39	0.560
	I2V (N = 4)	**81.36**	**13.84**	**0.348**	**15.81**	**0.574**
Motion Score	I2V	87.96	14.66	0.370	15.45	0.556
	I2V w/ *MA*	87.12	14.44	0.363	15.64	0.565
	I2V w/ *MA* & scores †	**81.36**	**13.84**	**0.348**	**15.81**	**0.574**
	Interp.	59.64	8.38	0.216	20.07	0.696
	Interp. w/ *MA*	59.79	8.31	**0.214**	20.15	0.699
	Interp. w/ *MA* & scores	**53.42**	**8.03**	0.215	**20.20**	**0.701**
Represent.	I2V w/ traj. (offset)	87.83	12.74	0.298	16.94	0.612
	I2V w/ traj. (heatmap)	80.57	12.65	**0.281**	**17.57**	**0.635**
	I2V w/ traj. (ours)	**72.04**	**12.55**	**0.281**	17.46	0.634

Table 2. **Ablation study on LayerAnimate.** *MA*: Motion-based Assignment. Interp.: Interpolation. traj.: trajectory control. †: the same setting with "I2V (N = 4)".

6. Conclusion

We propose *LayerAnimate*, a layer-level control framework combining the traditional layer separation philosophy in animation production with video generation models. LayerAnimate enables layer-level control over individual animation layers, allowing users to apply multiple controls to distinct layers. To address the issue of scarce layer-level data, we design a data curation pipeline to automatically extract layer from animations. Extensive experiments demonstrate its effectiveness and versatility. This framework opens up new possibilities for layer-level animation applications and creative flexibility.

Acknowledgements

This work was supported in part by the National Key R&D Program of China (No. 2022ZD0160102), the National Natural Science Foundation of China (No.U21B2042, No.62320106010).

References

[1] 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, 4

[2] Caroline Chan, Frédo Durand, and Phillip Isola. Learning to generate line drawings that convey geometry and semantics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7915–7925, 2022. 6

[3] 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

[4] Xinyuan Chen, Yaohui Wang, Lingjun Zhang, Shaobin Zhuang, Xin Ma, Jiashuo Yu, Yali Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. SEINE: Short-to-long video diffusion model for generative transition and prediction. In *ICLR*, 2024. 2, 6

[5] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *NeurIPS*, 34:8780–8794, 2021. 2

[6] Yuwei Guo, Ceyuan Yang, Anyi Rao, Maneesh Agrawala, Dahua Lin, and Bo Dai. Sparsectrl: Adding sparse controls to text-to-video diffusion models. In *ECCV*, pages 330–348. Springer, 2024. 2, 4

[7] 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. In *ICLR*, 2024. 2

[8] Yoav HaCohen, Nisan Chiprut, Benny Brazowski, Daniel Shalem, Dudu Moshe, Eitan Richardson, Eran Levin, Guy Shiran, Nir Zabari, Ori Gordon, et al. Ltx-video: Realtime video latent diffusion. *arXiv preprint arXiv:2501.00103*, 2024.

[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*, 2022. 2

[10] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *NeurIPS*, 30, 2017. 6

[11] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 33:6840–6851, 2020. 2

[12] 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

[13] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *NeurIPS*, 35:8633–8646, 2022. 2

[14] Li Hu. Animate anyone: Consistent and controllable image-to-video synthesis for character animation. In *CVPR*, pages 8153–8163, 2024. 2

[15] Zhitong Huang, Mohan Zhang, and Jing Liao. Lvcd: Reference-based lineart video colorization with diffusion models. *arXiv preprint arXiv:2409.12960*, 2024. 1, 2, 6

[16] Yang Jin, Zhicheng Sun, Ningyuan Li, Kun Xu, Hao Jiang, Nan Zhuang, Quzhe Huang, Yang Song, Yadong Mu, and Zhouchen Lin. Pyramidal flow matching for efficient video generative modeling. *arXiv preprint arXiv:2410.05954*, 2024. 2

[17] Nikita Karaev, Iurii Makarov, Jianyuan Wang, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker3: Simpler and better point tracking by pseudo-labelling real videos. *arXiv preprint arXiv:2410.11831*, 2024. 5, 6

[18] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *ICCV*, pages 4015–4026, 2023. 1, 2, 5

[19] Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, et al. Hunyuanvideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024. 2

[20] Bin Lin, Yunyang Ge, Xinhua Cheng, Zongjian Li, Bin Zhu, Shaodong Wang, Xianyi He, Yang Ye, Shenghai Yuan, Liuhan Chen, et al. Open-sora plan: Open-source large video generation model. *arXiv preprint arXiv:2412.00131*, 2024.

[21] Yixin Liu, Kai Zhang, Yuan Li, Zhiling Yan, Chujie Gao, Ruoxi Chen, Zhengqing Yuan, Yue Huang, Hanchi Sun, Jianfeng Gao, et al. Sora: A review on background, technology, limitations, and opportunities of large vision models. *arXiv preprint arXiv:2402.17177*, 2024. 2

[22] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 6

[23] Yifang Men, Yuan Yao, Miaomiao Cui, and Liefeng Bo. Mimo: Controllable character video synthesis with spatial decomposed modeling. *arXiv preprint arXiv:2409.16160*, 2024. 2

[24] Yihao Meng, Hao Ouyang, Hanlin Wang, Qiuyu Wang, Wen Wang, Ka Leong Cheng, Zhiheng Liu, Yujun Shen, and Huamin Qu. Anidoc: Animation creation made easier. *arXiv preprint arXiv:2412.14173*, 2024. 1, 2, 6

[25] Bohao Peng, Jian Wang, Yuechen Zhang, Wenbo Li, Ming-Chang Yang, and Jiaya Jia. Controlnext: Powerful and efficient control for image and video generation. *arXiv preprint arXiv:2408.06070*, 2024. 2

[26] Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, et al. Movie gen: A cast of

media foundation models. *arXiv preprint arXiv:2410.13720*, 2024. 2

[27] 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, 2, 3

[28] Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, Devi Parikh, Sonal Gupta, and Yaniv Taigman. Make-a-video: Text-to-video generation without text-video data. In *ICLR*, 2023. 2

[29] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2021. 2

[30] Shuai Tan, Biao Gong, Xiang Wang, Shiwei Zhang, Dandan Zheng, Ruobing Zheng, Kecheng Zheng, Jingdong Chen, and Ming Yang. Animate-x: Universal character image animation with enhanced motion representation. *arXiv preprint arXiv:2410.10306*, 2024. 2

[31] Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphaël Marinier, Marcin Michalski, and Sylvain Gelly. FVD: A new metric for video generation. In *ICLR workshop*, 2019. 6

[32] Wen Wang, Qiuyu Wang, Kecheng Zheng, Hao OUYANG, Zhekai Chen, Biao Gong, Hao Chen, Yujun Shen, and Chunhua Shen. Framer: Interactive frame interpolation. In *ICLR*, 2025. 1, 6

[33] 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. *NeurIPS*, 36, 2024. 2

[34] Weijia Wu, Zhuang Li, Yuchao Gu, Rui Zhao, Yefei He, David Junhao Zhang, Mike Zheng Shou, Yan Li, Tingting Gao, and Di Zhang. Draganything: Motion control for anything using entity representation. In *ECCV*, pages 331–348. Springer, 2024. 2, 5, 6

[35] Jinbo Xing, Hanyuan Liu, Menghan Xia, Yong Zhang, Xintao Wang, Ying Shan, and Tien-Tsin Wong. Tooncrafter: Generative cartoon interpolation. *arXiv preprint arXiv:2405.17933*, 2024. 1, 2, 4, 6

[36] Jinbo Xing, Menghan Xia, Yuxin Liu, Yuechen Zhang, Yong Zhang, Yingqing He, Hanyuan Liu, Haoxin Chen, Xiaodong Cun, Xintao Wang, et al. Make-your-video: Customized video generation using textual and structural guidance. *IEEE TVCG*, 2024. 2

[37] Jinbo Xing, Menghan Xia, Yong Zhang, Haoxin Chen, Wangbo Yu, Hanyuan Liu, Gongye Liu, Xintao Wang, Ying Shan, and Tien-Tsin Wong. Dynamicrafter: Animating open-domain images with video diffusion priors. In *ECCV*, pages 399–417. Springer, 2024. 1, 2, 4, 6

[38] Haofei Xu, Jing Zhang, Jianfei Cai, Hamid Rezatofighi, Fisher Yu, Dacheng Tao, and Andreas Geiger. Unifying flow, stereo and depth estimation. *IEEE TPAMI*, 2023. 3

[39] Jiaqi Xu, Xinyi Zou, Kunzhe Huang, Yunkuo Chen, Bo Liu, MengLi Cheng, Xing Shi, and Jun Huang. Easyanimate: A high-performance long video generation method based on transformer architecture. *arXiv preprint arXiv:2405.18991*, 2024. 2

[40] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer. *arXiv preprint arXiv:2408.06072*, 2024. 2, 6

[41] Yan Zeng, Guoqiang Wei, Jiani Zheng, Jiaxin Zou, Yang Wei, Yuchen Zhang, and Hang Li. Make pixels dance: High-dynamic video generation. In *CVPR*, pages 8850–8860, 2024. 2

[42] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *ICCV*, pages 3836–3847, 2023. 2

[43] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, pages 586–595, 2018. 6

[44] Shiwei Zhang, Jiayu Wang, Yingya Zhang, Kang Zhao, Hangjie Yuan, Zhiwu Qin, Xiang Wang, Deli Zhao, and Jingren Zhou. I2vgen-xl: High-quality image-to-video synthesis via cascaded diffusion models. *arXiv preprint arXiv:2311.04145*, 2023. 2

[45] Zhenghao Zhang, Junchao Liao, Menghao Li, Zuozhuo Dai, Bingxue Qiu, Siyu Zhu, Long Qin, and Weizhi Wang. Tora: Trajectory-oriented diffusion transformer for video generation. *arXiv preprint arXiv:2407.21705*, 2024. 2, 5, 6

[46] Zangwei Zheng, Xiangyu Peng, Tianji Yang, Chenhui Shen, Shenggui Li, Hongxin Liu, Yukun Zhou, Tianyi Li, and Yang You. Open-sora: Democratizing efficient video production for all. *arXiv preprint arXiv:2412.20404*, 2024. 2, 5

[47] Yuan Zhou, Qiuyue Wang, Yuxuan Cai, and Huan Yang. Allegro: Open the black box of commercial-level video generation model. *arXiv preprint arXiv:2410.15458*, 2024. 2