Generative Keyframing

Xiaojuan Wang

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2025

Reading Committee:
Steven M. Seitz, Chair
Brian Curless, Chair
Ira Kemelmacher-Shlizerman

Program Authorized to Offer Degree:
Computer Science & Engineering

University of Washington

Abstract

Generative Keyframing

Xiaojuan Wang

Co-Chairs of the Supervisory Committee:
Steven M. Seitz
Computer Science & Engineering

Brian Curless
Computer Science & Engineering

Keyframing is a fundamental element of animation creation and video editing. It involves defining specific frames, *keyframes*, that mark important moments of change and guide how the intermediate frames are filled or interpolated. In early hand-drawn animation, a keyframe is a visual drawing created by animators, with assistants manually drawing the in-between frames. With the advent of digital animation and video editing software, a keyframe became a set of parameters that define the state of the rendered character or object at specific times, with in-between transitions produced by interpolating these parameters.

However, such parametric approaches rely heavily on manually designed controls and artist-crafted heuristics, making them difficult to capture complex, nuanced, and realistic motions. Furthermore, they do not naturally generalize to real image and video domains. The rapid progress of visual generative models that are trained on large collections of visual data and capable of learning rich appearance and motion patterns, has made it possible to generate high-fidelity imagery and realistic motion. Building on these advances, this thesis investigates *generative keyframing*, a data-driven, non-parametric, image-based approach to the keyframing process. To this end, I present a series of works in this thesis that collectively develop and explore this idea.

I begin with the basic aspect: using generative models to synthesize transitions directly from images, and even to fully generate in-between motions. I first present a GAN-based technique for smoothing jump cuts in talking head videos, synthesizing seamless transitions between the cuts even in challenging cases involving large head movement. I then introduce a method for generating in-between videos with dynamic motion between more distant key frames by adapting a pretrained large-scale image-to-video diffusion model with minimal fine-tuning effort.

Beyond automatically generating transitions between keyframes, I further explore multi-scale keyframing for achieving very deep zoom. Specifically, I introduce a multi-scale joint sampling diffusion approach for generating consistent images (keyframes) across different spatial scales while adhering to their respective input text prompts. This enables deep semantic zoom and a continuous zoom video can be rendered from these images. When working with multiple keyframes, one import question is how they should be ordered in the final video. I address this in the context of dance video generation—specifically, music synchronized and choreography-aware animal dance video—where unordered keyframes representing distinct animal poses are arranged via graph optimization to satisfy a specified choreography pattern of beats that defines the long-range structure of a dance. Finally, I conclude with discussions and directions for future works.

TABLE OF CONTENTS

ACKNOWLEDGMENTS

When I first started my Ph.D., I imagined I would become an expert with deep, specialized knowledge in a single field. What I actually learned, and now value far more, is how to discover meaningful problems and solve them with clarity and elegance. This is largely thanks to my advisors Steve Seitz, Brian Curless, Ira Kemelmacher, and to my mentor Aleksander Holynski, who guided me into a research path that encouraged creativity, originality, and ambitious thinking. Looking back, I am proud to say it has been a truly worthwhile journey, made possible by the many people who helped me, supported me, and inspired me along the way. I would like to thank them wholeheartedly, and to do so in the spirit of the lessons I learned throughout this journey.

Killer demo mindset. This is the first lesson that comes to mind, and one I owe largely to Steve Seitz, Brian Curless, and Ira Kemelmacher. At the start of every project, they encouraged me to imagine the strongest possible demo: envision the ideal input and output, consider whether the result would be surprising and impressive, even to a general audience, and ensure the input is simple enough for anyone to use. Only then, then would say, is a project truly worth pursuing. This mindset shaped the way I approached research, and taught me to focus only on projects that have the potential to be genuinely impactful.

Presentation. My presentation style has been shaped primarily by Aleksander Holynski and Steve Seitz. I remember watching Aleksander's "*How I Stopped Worrying and Loved the Data Monster*" talk multiple times and studying how he unfolded his center idea; it became my gold standard for a research presentation. He delivered the core message right

at the beginning, structured the talk into clear, digestible parts, continually reminded the audience of the big picture, and used clever visuals to reinforce each idea. Steve, on the other hand, is a master storyteller. I still think about his "*Slow Glass*" talk: how he used an intriguing story as a motivation or metaphor for the concept he wanted to introduce, and how naturally he would draw the audience in with phrases like "*Imagine if we could...*" or "*Picture yourself in...*". His "*Graphics in 5 Minutes*" cartoon series also inspired me to design well-planned animations that make complex vision and graphics concepts simple and intuitive. Together, these influences taught me to craft strong motivations, rely heavily on visuals and animations rather than words, and always offer my own insights to the audience in my talks.

Research skills. I have learned so much about the practices that lead to reliable research progress from Steve Seitz and Aleksander Holynski. Aleksander's insistence on visualizing intermediate results and exhaustively enumerating parameter configurations to determine whether a step truly works had a profound impact on my research. This pushes me to dig deeper into methodology and to ground every conclusion in solid logic and evidence. Steve taught me to constantly keep the big picture in mind and to always consider alternative solutions on the table. His guidance helped me avoid getting trapped in local fixes or narrow directions that might never address the core problem. Together, their influences shaped a balanced approach: rigorous, detail-oriented investigation paired with a clear understanding of the big picture.

Engineering. I learned a great deal about practical debugging from Meng-Li Shih and Bowei Chen, whose approaches were very different from my own. When implementing a new method, they taught me to begin by running the debugger and observing the code in action: fixing issues as they arise rather than trying to reason through every line mentally before execution. Their hands-on, iterative style fundamentally changed the way I approach

engineering and troubleshooting.

Communication and Collaboration. These skills were shaped largely through working with my advisors and with Aleksander Holynski. I have always admired Aleksander's ability to explain the same idea at different levels of abstraction, tailored to different audiences—something that stems from a deep, precise understanding of his own work. During my internship with him at Google Research, I also learned how to seek help and foster collaboration effectively: by preparing everything thoroughly on my end so that colleagues could step in with minimal effort or context switching. This mindset has stayed with me and continues to guide how I communicate and collaborate in research.

I would also like to thank my UW GARIL lab mates, who made my life so much happier and easier: Yifan Wang, Luyang Zhu, Teerapat Jenrungrot, Vivek Jayaram, Yuxuan Mei, Kostas Rematas, Roy Or-El, Chung-Yi Weng, Jingwei Ma, Mengyi Shan, Benlin Liu, Johanna Karras, Mengli Shih, Alice Gao, Baback Elmieh, and Susung Hong.

Lastly, and most importantly, I would like to thank my husband, Tong He, for his unwavering companionship, patience, and support throughout this journey and in our life together.

DEDICATION

To Tong

who has been subjected to my long discourses on my research and held me through every frustration,

this is my penance.

Chapter 1

INTRODUCTION

Everyday we watch videos: animations, TV shows, movies, TikTok, *etc.* We also create videos ourselves to capture memories or share ideas. Video has become one of the most powerful and expressive ways we communicate, tell stories, and create art. From cinematic films to short-form online content, videos convey motion, emotion, and narrative in ways that static imagery never could. Behind every compelling video lies not only visual fidelity but also temporal structure—the careful orchestration of frames that define how a story unfolds over time. A central element in this process is *keyframing*—the practice of defining specific frames, *keyframes*, that mark important moments of change in an animation or video and guide how the intermediate frames are filled or interpolated.

Formally, a keyframe represents either a visual frame or a set of underlying parameters

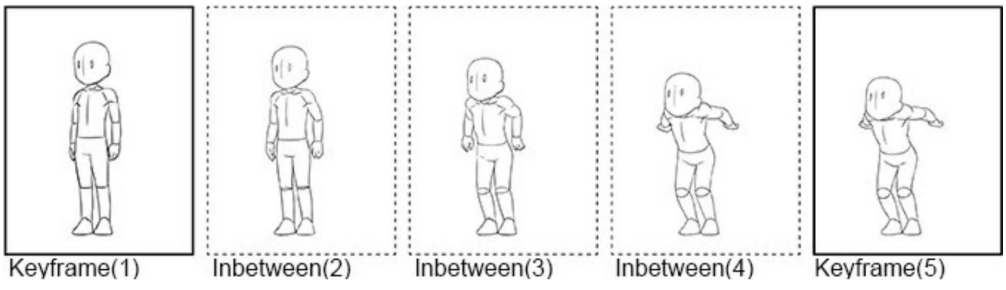


Figure 1.1: Keyframes define the essential poses of a motion, while in-betweens fill the gaps to create a smooth animated sequence. source: https://tips.clip-studio.com/en-us/articles/954

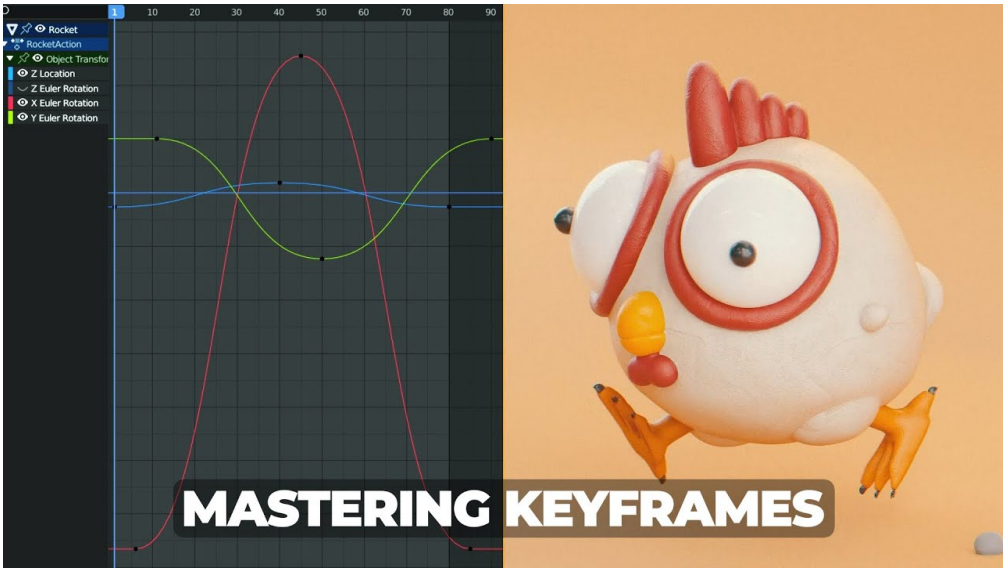


Figure 1.2: Keyframe-based character animation in Blender (tutorial by SouthernShotty).

that define the state of an object or scene at a specific time, establishing the starting and/or ending points of a smooth transition. In animation, keyframes correspond to distinct poses or states of a character at critical moments, with the intermediate frames interpolated to produce continuous motion (see Fig. 1.1 for an example). In early hand-drawn animation, pioneered by early studios such as Walt Disney Animation Studio, animators created the keyframes first, while assistant workers manually drew the in-between frames. With the advent of computer graphics, this manual inbetweening process was gradually automated: animators began defining keyframes as control points for graphical parameters such as position, rotation, or joint angles of the character, and software interpolated smooth motion curves between them (as shown in Fig. 1.2). This shift transformed animation from a labor-intensive frame-by-frame process into one driven by mathematically interpolated curves that define motion between keyframes.

The same principle later became foundational in modern video editing and compositing, where keyframes are used not to define character states but to control visual properties over time. Editors employ keyframing to create a wide range of visual effects and transitions, such

as zooming, rotating, and changing the scale, position or opacity of objects. For example, if an editor wants to gradually zoom in on an object, they can place one keyframe at the point where the zoom where the zoom parameter (*e.g.* camera focal length or scale) is set to its initial value, and another where it reaches the target value; the editing software then automatically interpolates between them to create a smooth zoom transition.

As introduced above, while fundamental to animation and video editing, traditional keyframing relies on parameterized control to define a character's or object's state and to create transitions between keyframes. The transitions are typically produced through interpolation functions that blend the parameter values over time. However, such parametric methods are inherently limited and struggle to capture complex, realistic, or semantically meaningful transitions. Moreover, because they do not operate directly in the image space, they fail to model the rich appearance changes that occur in natural videos, such as variations in lighting, texture, deformation, and object interaction.

Meanwhile, advances in visual generative models have transformed how image and videos can be created. Early approaches based on Generative Adversarial Networks (GANs) [34] demonstrated the potential of deep neural networks to synthesize realistic imagery and perform video-to-video translation. More recently, diffusion models [156, 11, 103, 104, 39, 27, 95, 8, 4, 154, 151] have surged in popularity for generative images and videos modeling. Driven by large-scale training datasets and advances in multi-modal learning, diffusion based generative models can now generate images and videos directly from texts. A user can simply specify a text prompt describing the desired content or provide a reference image, and then the model generates a high-resolution image or video clip with vivid motion.

These developments have made data-driven, image-based approaches possible. This opens up new opportunities: can we move beyond the parametric approaches in traditional keyframing with a non-parametric, image-based generative process using models that learn motion and appearance dynamics directly from visual data? Toward this goal, this thesis presents a number of works that collectively explore the idea of generative keyframing. I begin with the most fundamental aspect: generating smooth and realistic transitions, even videos be-

Figure 1.3: **Jump cut smoothing for talking head videos.** Given a talking head video, after removing the filler words and repetitive words (text in red color), Chapter 4 presents a method for synthesizing a seamless transition (marked by orange rectangles) to smooth the resulting jump cut.

tween keyframes. Beyond transitions, I further explore multi-scale keyframing for generating deep zoom, as well as choreography-aware keyframing for transforming a set of pre-generated keyframes into music-synchronized animal dance videos.

Generative inbetweening. Chapter 4 and 5 explore how to use generative models to synthesize in-between frames directly from images, even generate realistic and dynamic motion between keyframes. Chapter 4 focuses on talking head video editing, where users may remove unwanted segments from footage, and wish to smooth the resulting jump cuts. I present a method that synthesizes intermediate frames to smooth such abrupt cuts that would otherwise cause an unwanted, jarring viewing experience. The technique supports a variety of edits, such as removing filler words (as shown in Fig. 1.3), pauses, or even arbitrary cuts, and achieves seamless transitions even in challenging cases where the speaker rotates or moves significantly.

Besides producing smooth and linear transition in cut edits where "keyframes" are tem-

porally close, recent large-scale video generation models have demonstrated the ability to generate high resolution videos clips with dynamic motion. These capabilities now enable the generation of in-between videos even between distant keyframes. In Chapter 5, I present a method for generating video sequences with dynamic motion between a pair of input key frames. We adapt a pretrained large-scale image-to-video diffusion model (originally trained to generate videos moving forward in time from a single input image) for key frame inbetweening. This adaptation is achieved through a lightweight fine-tuning technique that produces a version of the model that instead predicts videos moving backwards in time from a single input image. This model (along with the original forward-moving model) is subsequently used in a generation process that merges their overlapping model predictions starting from each of the two keyframes, resulting in a video with smooth and coherent motion between the two inputs.

Multi-scale keyframing for deep zoom generation. In Chapter 6, I present *Generative Powers of Ten*, a method for generating deep zoom by generating multi-scale consistent images at each zoom level (see Fig. 1.4 for examples of these images). This enables extreme semantic zooms into a scene, from which continuous zoom videos can be rendered by treating the generated images as keyframes. The proposed approach takes as input a series of prompts describing a scene at varying scales, *e.g.*, , from a wide-angle landscape view of a forest to a macro shot of an insect sitting on one of the tree branches, and uses a pre-trained text-to-image diffusion model to generate consistent content across multiple image scales. This is achieved through a novel joint multi-scale diffusion sampling approach that encourages consistency across different spatial scales while preserving the integrity of each individual sampling process. Since each generated scale is guided by its own text prompt, the proposed method enables deeper levels of zoom than traditional super-resolution methods that may struggle to create new contextual structure at vastly different scales.

Choreography-aware keyframing for animal dance generation. When working with

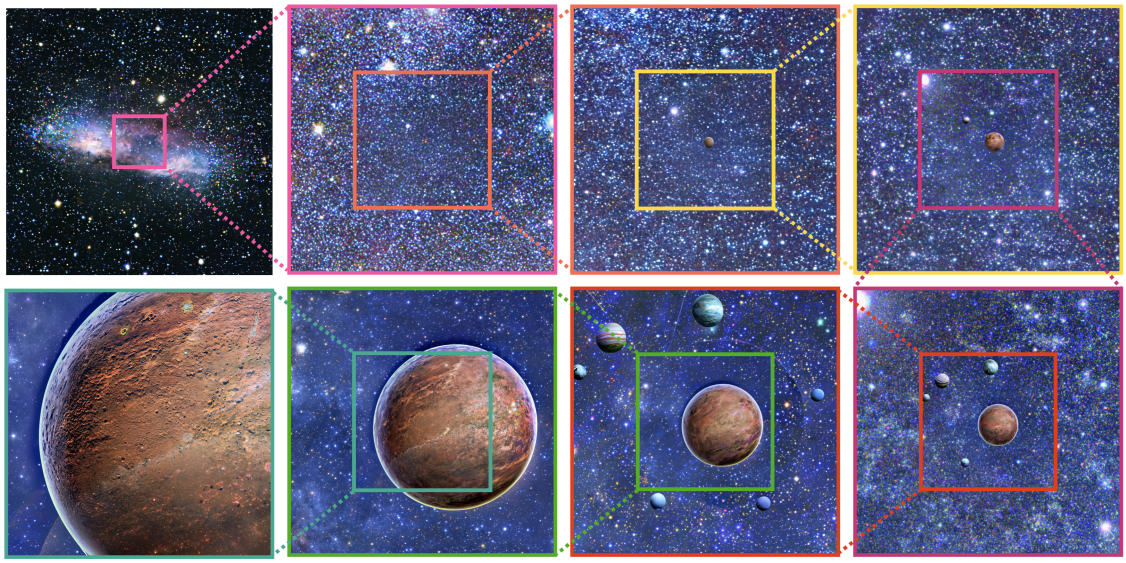


Figure 1.4: **Multi-scale keyframing for deep zoom generation.** Given a series of prompts describing a scene at varying zoom levels, *e.g.*, from a distant galaxy to the surface of an alien planet, Chapter 6 (*Generative Powers of Ten*) presents a method that uses a pre-trained text-to-image diffusion model to generate consistent keyframes at each zoom level (shown above), enabling extreme semantic zooms into a scene.

multiple keyframes, their temporal relationship becomes crucial for shaping the resulting video. For example, Chapter 6 assumes a linear progressive zoom structure, where keyframes are organized according to zoom levels, and this ordering guides the keyframes generation process to ensure multi-scale consistency. However, not all video structures are linear. Dance, for example, follows basic choreography rules that structure the movements to align with the rhythmic flow of the accompanying music, and often involves recurring patterns such as mirroring and repetition to help reinforce the musical structure. Chapter 7 explores such dance video generation, specifically, generating music-synchronized, highly structured, 30 second+ long animal dance videos. Starting from a few unordered keyframes representing distinct animal poses, generated via text-to-image prompting or GPT-4o, we formulate dance synthesis as a graph optimization problem that seeks the optimal keyframe structure to

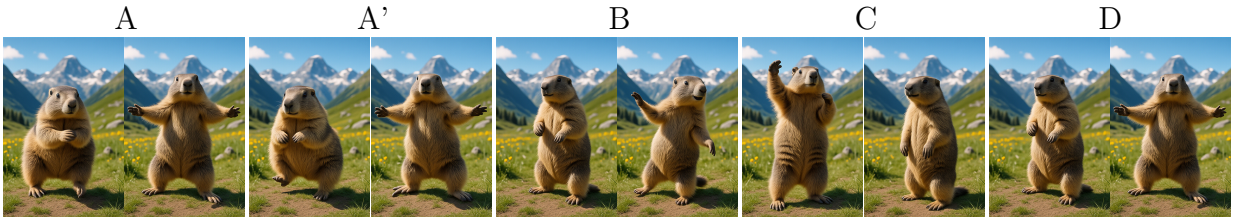


Figure 1.5: **Choreography-aware keyframing for animal dance generation.** Starting from a small set of unordered generated keyframes, *e.g.*, a marmot in various poses, Chapter 7 presents a keyframe based framework for generating animal dance videos to follow a specified choreography pattern. In this example, the choreography pattern is extracted from a Youtube dance clip [jkRIIH42Vo8] (12.0s to 33.24s): A-A'-A-A'-A-A'-A-A'-B-B-B-B-B-B-B-B-C-C-C-C-C-D-D'-D. Keyframe pairs are labeled according to the choreography segments and arranged in t he sequence defined by the pattern.

satisfy a specified choreography pattern of beats that defines the long-range structure of a dance (see Fig. 1.5 for an example). We also introduce an approach for mirrored pose image generation, essential for capturing symmetry in dance. In-between frames are generated using an video diffusion model. With as few as six input keyframes, the method can generate up to 30 seconds dance videos across a wide range of animals and music tracks.

Overview. This thesis is organized as follows. I begin with the following chapter (Chapter 2) with a survey of traditional keyframing techniques used in animations and video editing. Chapter 3 introduces the mathematical foundations of generative models, focusing on GANs and diffusion models. Chapter 4 and Chapter 5 focus on synthesizing transitions between keyframes: the former introduces a GAN-based approach for smoothing cuts in talking-head videos, where keyframes correspond to edit points and the transition is assumed to be linear, while the latter presents a diffusion-based method for generating in-between videos between distant keyframes. Chapter 6 presents a method for generating consistent images across multiple scales from a sequence of input prompts, which is then used to render a continuous

deep zoom video. Chapter 7 introduces a keyframe-based framework for generating music synchronized animal dance videos that also meet choreography pattern by optimizing the order of the keyframes and also generating in-between motions. Finally, Chapter 8 concludes with directions for future investigation.

Chapter 2

A SURVEY OF TRADITIONAL KEYFRAMING PRACTICE

Figure 2.1: Left: a celluloid sheet of Snow White singing to the seven dwarfs from *Snow White and the Seven Dwarfs*, the first **hand-drawn** feature length animation film made in 1937. Photograph: © Mike Pucher/Disney Enterprises, Inc. Right: A screenshot of animation process of *Toy Story*, the first fully **computer-animated** feature length film made in 1995. Source: Toy Story Behind the Scenes (Youtube [5TqPl3MSSow]).

This chapter introduces the fundamental components and principles of traditional keyframing in animation and visual effects editing. Section 2.1 what keyframes are and the principles for specifying them. Section 2.2 explains how the in-between frames are interpolated to complete the motion. Section 2.3 discusses the timing of keyframes, that is, where they should be placed along the timeline. Finally Section 2.4 outlines the limitations of this traditional practice.

2.1 Keyframes itself

Keyframing has been deeply rooted in animation industry for decades. In early hand-drawn animation, animators begin by drawing keyframes in which characters are placed in *key poses*. These poses are essential for laying out and structuring an animated shot and for defining the movement. Modern digital animation such as 3D animation (CGI), 2D vector and raster animation, builds on the same principles, but the keyframes here are no longer hand-drawn. Instead, they are a set of parameters that specify the key poses of rendered characters .These parameters typically include objects transformations such as position, rotation, and scale, among others.

A strong key pose is one that clearly captures the essence of an action, usually defining the start, middle, and end of an action. It's a pose where a character or an object is at its extreme. Fig. 2.2 shows an example of the key poses of a walking cycle including (1) contact pose where the heel of the front foot just touches the ground; (2) down pose where weight shifts downward as the body absorbs impact; (3) passing pose where the back foot lifts as the front leg passes under the body; and (4) up pose, where the weight rises as the body moves upward before the next contact. Adjusting the details of these key poses, such as arm swings, head tilts, or torso movement help inject personality into the walking cycle. For example, a tired character might have drooping arms and dragging feet; a soldier might keep arms stiff and shoulders square, while a child might bounce with extra energy in the up pose.

In video editing software such as Adobe After Effects[1], keyframes are used to create a wide range of visual effects. Editors place keyframes to mark where an effect begins and ends, such as the start and finish of a fade, zoom, or color adjustment. In this context, keyframes represent parameter values that control various editing properties. The software then interpolates between these property values over time, automatically generating smooth transitions. This allows editors to animate nearly any parameter, such as brightness, opacity,

[1]https://www.adobe.com/products/aftereffects.html

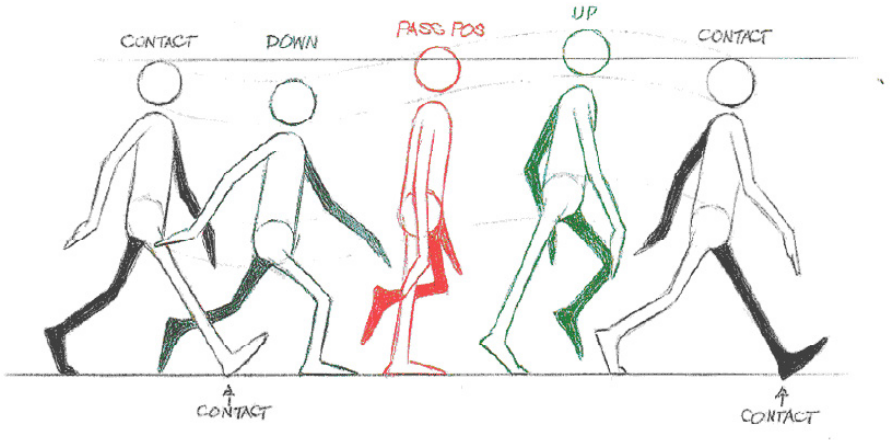


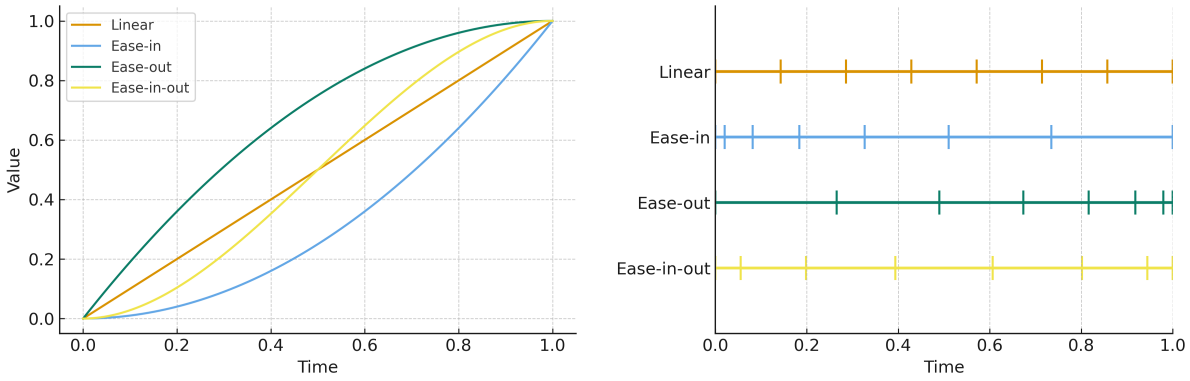
Figure 2.2: **Key poses in a walk cycle:** (1) contact pose; (2) down pose; (3) passing pose; and (4) up pose.

position, rotation, or audio volume, without manually adjusting every frame.

2.2 Inbetweening

After specifying the keyframes, the next step is to create the transition between them for smooth animation. In hand-drawn animation, transitions between the key poses are achieved by manually drawing the in-between frames, a process often carried out by junior or assistant animators. In modern digital animation and video editing, these transitions are produced by interpolating the keyframe parameters, allowing the character or object to move as its properties follow the interpolation curves over time.

The way in-between frames are distributed determines how the speed of the motion changes, and relates to the concept of *spacing*, which describes an object's position in each frame and how much it moves from one frame to the next. When an object's position changes only slightly between frames, the motion appears slower; when the spacing increases, the motion appears faster. For instance, when animating a car accelerating from a stop over 24 frames, the car's position in the first few frames should change only slightly. As the car gains speed, the spacing between its positions should increase, reflecting natural accelera-



(a) Interpolation curves.

(b) Spacing of sampled in-between frames.

Figure 2.3: Common interpolation functions used in keyframing and their induced spacing patterns, where denser sampling indicates slower motion and sparser sampling indicates faster motion.

tion. In digital animation, spacing here is encoded in the curvature of the interpolation function. Common methods such as linear, ease-in-out, and Bezier interpolation mathematically control the rate of change between keyframe values, *i.e.*, how motion progresses. Fig. 2.3 visually compares these interpolation curves, and their induced spacing patterns. Linear interpolation produces uniform motion, where position changes at a constant rate from start to finish. In contrast, ease-in-out interpolation models gradual acceleration and deceleration, resulting in motion that feels more organic, believable, and visually pleasing. For example, imagine a car instantly accelerating to full speed and instantly stopping, that's linear interpolation. The car's motion feels abrupt and unrealistic because real objects must accelerate and decelerate. Similarly, video editors can apply various interpolation methods to fine-tune the pacing of an visual effect, so it synchronizes with music or scene transitions.

2.3 Timing the keyframes

When a series of keyframes are defined, an important consideration is where to place them along the timeline, that is, their *timing*. The distance between the keyframes determines how long an action or movement or transition takes to complete. In animation, adding more in-between frames slows the motion, while fewer frames make it appear faster. Animators adjust timing across different actions to reflect natural variation in movement, guided both by physical observation and expressive intent. As noted in the traditional principles of animation established by the Disney animators [57], timing plays a central role in conveying weight, emotion, and personality. A slight change in timing can completely alter the perceived character of an action: a short pause can heighten anticipation, while extended spacing between key poses can evoke gravity, hesitation, or drama. Thus, timing functions as a creative control that balances physical plausibility with expressive clarity.

2.4 Limitations

As introduced above, traditional parametric keyframing relies on hand-crafted mathematical functions that operate on predefined control parameters, which makes it unable to capture complex and nuanced motions such as articulated movements, or non-rigid deformations. It also places a heavy burden on the animator: designing and tuning timing curves, adjusting control points, and manually refining the motion require significant expertise and effort, especially for long sequences or high-fidelity animations. This workflow is not only time-consuming but also scales poorly when dealing with complex scenes, multiple interacting characters, or realistic natural movements. Furthermore, the parametric paradigm does not naturally generalize to real images or videos, where motion is defined in pixel space rather than by neatly separated object parameters. As a result, traditional keyframing cannot be applied directly to the synthesis or manipulation of real-world footage, greatly limiting its applicability.

Chapter 3

HOW GENERATIVE MODELS WORK

This chapter introduces the foundational principles of generative models, and focuses on the classes of generative models that will be used in subsequent chapters, specifically we present Generative Adversarial Networks (GANs) in Section 3.1 and diffusion models in Section 3.2. Section 3.3 further discusses the connection between diffusion and score-based generative models, as well as how guidance techniques can be applied in diffusion models to generate data under specific conditions, a capability leveraged in both Chapter 5 and Chapter 6.

Generative models aim to learn the underlying patterns of training data, namely, its data distribution, to generate new samples that share characteristics with the original training data but are not identical to any specific example in the training set. Examples of such data include natural images, videos, or text, which exhibit specific, complex regularities that distinguish them from random noises. For instance, if a generative model is trained on a million cat photos, the data distribution captures the collective characteristics of all possible cat images, including variations in breeds, poses, lighting, and textures, and the model learns these patterns to generate novel, realistic cat images, rather than simply memorizing and replicating the existing ones.

Formally, such training data are governed by an unknown data distribution $\{x_i\}_{i=1}^{N} \sim p_{\text{data}}(x)$, where $p_{\text{data}}(x) : \mathcal{X} \to R$, which assigns a probability to each sample x in the data domain $\mathcal{X}$. Fig. 3.1 illustrates such a data distribution using the MNIST [70] dataset. In practice, this true distribution is inaccessible, so the goal of generative modeling is to learn a parameterized distribution $p_\theta(x)$ that approximates $p_{\text{data}}(x)$. New samples can then be generated by drawing $x \sim p_\theta(x)$. At their core, all modern generative models seek to

(a) PCA visualization (b) t-SNE visualization

Figure 3.1: Visualization of the MNIST [70] data distribution using PCA and t-SNE embeddings [83]. Each color represents a distinct digit class. Regions with high sample density correspond to areas of higher probability mass, while sparse regions indicate low-probability areas. The goal of generative modeling is to model this distribution and generate new samples from this distribution.

optimize the same fundamental objective:

$$\theta^* = \arg\min_{\theta} D(p_{\text{data}}(x) \,\|\, p_\theta(x)) \tag{3.1}$$

where $D(\cdot \,\|\, \cdot)$ is a divergence measure such as Kullback–Leibler divergence. Different generative model families differ primarily in (i) how they define $p_\theta(x)$, (ii) how they evaluate or approximate this divergence, and (iii) how they perform learning and sampling.

3.1 Generative adversarial networks

GANs [34] define an implicit generative model that does not provide a tractable likelihood. It introduces a generator $G_\theta(z) : \mathcal{Z} \to \mathcal{X}$ that directly maps latent variable z sampled from a simple prior distribution (*e.g.*, $z \sim \mathrm{N}(0, I)$, or $\mathrm{Uniform}[0, 1]$), to the data space. A discriminator $D_\phi(x)$ is trained simultaneously to distinguish real samples from generated ones, estimating the probability that a given sample originates from the true data distribution

rather than the generator. The training procedure for the generator $G_\theta(z)$, in turn, is to maximize the probability of $D_\phi(x)$ making a mistake. This adversarial training framework corresponds to a minimax two-player game. In the space of arbitrary functions $G_\theta(z)$ and $D_\phi(x)$, a unique solution exists, with $G_\theta(z)$ recovering the training data distribution and $D_\phi(x)$ equal to 1/2 everywhere.

The original GAN objective [34] corresponds to minimizing the Jensen–Shannon divergence between p_{data} and p_θ:

$$\min_\theta \max_\phi \; \mathbb{E}_{x\sim p_{\text{data}}}[\log D_\phi(x)] + \mathbb{E}_{z\sim p(z)}[\log(1 - D\phi(G_\theta(z)))] \tag{3.2}$$

Under idealized conditions (optimal discriminator), the generator minimizes: $\text{JS}(p_{\text{data}} \,\|\, p_\theta)$

3.2 Diffusion models

Given samples from a data distribution $q(\mathbf{x}_0)$, diffusion models including Denoising Diffusion Probabilistic Models (DDPMs) [115, 42], and Denoising Diffusion Implicit Models (DDIMs) [116] are latent variable models, where they learn the distribution p_θ of the form

$$p_\theta(\mathbf{x}_0) := \int p_\theta(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T}, \text{where } p_\theta(\mathbf{x}_{0:T}) := p_\theta(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta^{(t)}(\mathbf{x}_{t-1}|\mathbf{x}_t) \tag{3.3}$$

to approximates $q(\mathbf{x}_0)$, where $\mathbf{x}_1, ..., \mathbf{x}_T$ are latents of the same dimensionality as $\mathbf{x}_0$. The parameters θ are learned to fit the data distribution $q(\mathbf{x}_0)$ by maximizing a variational lower bound:

$$\max_\theta \mathbb{E}_{q(\mathbf{x}_0)}[\log p_\theta(\mathbf{x}_0)] \geq \max_\theta \mathbb{E}_{q(\mathbf{x}_0,\mathbf{x}_1,...,\mathbf{x}_T)}[\log p_\theta(\mathbf{x}_{0:T}) - \log q(\mathbf{x}_{1:T}|\mathbf{x}_0)] \tag{3.4}$$

where $q(\mathbf{x}_{1:T}|\mathbf{x}_0)$ is some inference distribution over the latent variables.

Forward process. DDPMs construct $q(\mathbf{x}_{1:T}|\mathbf{x}_0)$ using a *forward* diffusion process that gradually adds Gaussian noise to the clean image $\mathbf{x}_0$ through a Markov chain with Gaussian transitions, parameterized by a decreasing sequence $\alpha_{1:T} \in (0, 1]^T$:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}), \text{where } q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\sqrt{\frac{\alpha_t}{\alpha_{t-1}}}\mathbf{x}_{t-1}, (1 - \frac{\alpha_t}{\alpha_{t-1}})\mathbf{I}) \tag{3.5}$$

A notable property of the forward process is that it admits sampling $\mathbf{x}_t$ at an arbitrary timestep t in closed form:

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t}\mathbf{x_0}, (1 - \alpha_t)\mathbf{I}), \tag{3.6}$$

so the intermediate noised data $\mathbf{x}_t$ is created by $\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1 - \alpha_t}\boldsymbol{\epsilon}_t$, where $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. When α_t is close to 0, $q(\mathbf{x}_t|\mathbf{x}_0)$ converges to a standard Gaussian, and α_t is close to 1, it is clean data observation.

Generative process. The *generative* process, defined by the latent variable model $p_\theta(\mathbf{x}_{0:T})$, is a Markov chain that begins with pure noise $\mathbf{x}_T$ and progressively denoises it to produce clean data $\mathbf{x}_0$. More specifically, the generative process models each reverse transition as

$$p_\theta^{(t)}(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x_t}, t), \Sigma_\theta(\mathbf{x}_t, t)) \tag{3.7}$$

where the network parameterizes the mean μ_θ and variance $\Sigma_\theta(\mathbf{x}_t, t)$ of the Gaussian transition used to reconstruct $\mathbf{x}_{t-1}$ from the noisier sample $\mathbf{x}_t$. When the variance is fixed, the objective in Eq. 3.4 can be simplified as:

$$\mathcal{L}(\theta) = \mathbb{E}_{t\sim U[1,T], \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0},\mathbf{I})}[w(t)\|\boldsymbol{\epsilon}_\theta(\sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1 - \alpha_t}\boldsymbol{\epsilon}_t, t) - \boldsymbol{\epsilon}_t\|_2^2] \tag{3.8}$$

where $\boldsymbol{\epsilon}_\theta$ is a neural network approximator intended to predict $\boldsymbol{\epsilon}_t$ from $\mathbf{x}_t$, and $w(t)$ is a weighting function typically set to 1 [42]. To sample $\mathbf{x}_{t-1} \sim p_\theta^{(t)}(\mathbf{x}_{t-1}|\mathbf{x}_t)$, DDPMs compute:

$$\mathbf{x}_{t-1} = \sqrt{\frac{\alpha_{t-1}}{\alpha_t}}(\mathbf{x}_t - \frac{1 - \alpha_t/\alpha_{t-1}}{\sqrt{1 - \alpha_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)) + \sigma_t\mathbf{z} \tag{3.9}$$

where $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$ else $\mathbf{z} = \mathbf{0}$.

Sampling efficiency. The length T of the forward process is an important hyperparameter in DDPMs, which typically require large T (*e.g.*, $T = 1000$). To improve sampling

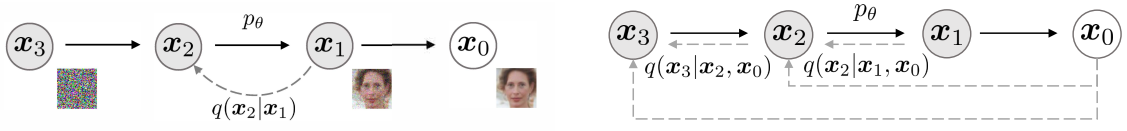


Figure 3.2: Graphical models for DDPM [42] (left) and non-Markovian DDIM [116] (right) inference models. Figure borrowed from [116].

efficiency, DDIMs [116] reformulate the generative process as a deterministic mapping using non-Markovian transitions (see Fig. 3.2 for illustration). Accordingly, the forward process is also non-Markovian with the transitions defined as $q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)$. This reformulation leads to the same surrogate objective function as DDPMs, and the DDIMs sampling update is given by:

$$\mathbf{x}_{t-1} = \sqrt{\alpha_{t-1}} \underbrace{\left(\frac{\mathbf{x}_t - \sqrt{1-\alpha_t}\epsilon_\theta(\mathbf{x}_t, t)}{\sqrt{\alpha_t}} \right)}_{\text{“ predicted } \mathbf{x}_0\text{”}} + \underbrace{\sqrt{1-\alpha_{t-1}} \cdot \epsilon_\theta(\mathbf{x}_t, t)}_{\text{“direction pointing to } \mathbf{x}_t\text{”}} \tag{3.10}$$

The non-Markovian diffusion processes in DDIMs lead to “short” generative chains that can be simulated in a small number of steps, accelerating sampling by $10x$ to $100x$ compared to DDPMs.

3.3 Guidance in generative models sampling

Certain generative models, such as GANs, exhibit a useful behavior known as truncated or low-temperature sampling [10, 67]: by restricting the variance or range of the input noise during sampling, the model produces outputs with lower diversity but higher quality.

In diffusion models, a similar effect is achieved through *guidance* during the sampling process. Diffusion models can be interpreted as score-based models [117, 118, 115], where the network prediction $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ in Eq. 3.9 approximates the score function of the noisy marginal distributions: $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t) \approx -\sqrt{1-\alpha_t}\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)$. A key advantage of diffusion models is their

ability to adapt outputs at inference time by steering the denoising trajectory. From the score-based viewpoint, guidance [27] can be seen as composing or modifying score functions to sample from richer or conditioned distributions. In practice, this is achieved by adjusting the update direction at each denoising step, thereby biasing the generation toward desired attributes or conditions.

Classifier guidance. Classifier guidance [20] can generate conditional samples from an unconditional model by combining the unconditional score function for $p(\mathbf{x}_t)$ with a classifier $p(\mathbf{c}|\mathbf{x}_t)$ to generate samples from $p(\mathbf{x}_t|\mathbf{c}) \propto p(\mathbf{c}|\mathbf{x}_t)p(\mathbf{x}_t)$. To use classifier guidance, one needs access to a labeled dataset and has to learn a noise-dependent classifier $p(\mathbf{c}|\mathbf{x}_t)$ that can be differentiated with respect to the noisy image $\mathbf{x}_t$. While sampling, we can incorporate classifier guidance by modifying the diffusion score as follows:

$$\hat{\epsilon} = \epsilon_\theta(\mathbf{x}_t, t, \mathbf{c}) - w\sqrt{1-\alpha_t}\nabla_{\mathbf{x}_t} \log p(\mathbf{c}|\mathbf{x}_t) \tag{3.11}$$

where w is an additional parameter controlling the guidance strength. Classifier guidance moves the sampling process towards images that are more likely according to the classifier, achieving a similar effect to truncation in GANs.

Classifier-free guidance. Instead of training a separate classifier model, classifier-free guidance [43] uses a single neural network to parameterize both conditional models and unconditional models, where the unconditional one is realized by supplying a null token $\emptyset$ as the class identifier during score prediction, *i.e.*, $\epsilon_\theta(\mathbf{x}_t, t) = \epsilon_\theta(\mathbf{x}_t, t, \mathbf{c} = \emptyset)$. It trains the unconditional and conditional models jointly by randomly setting $\mathbf{c}$ to be $\emptyset$. Then the diffusion score is computed as a linear combination of the conditional and unconditional score estimates:

$$\hat{\epsilon} = (1+w)\epsilon_\theta(\mathbf{x}_t, t, \mathbf{c}) - w\epsilon_\theta(\mathbf{x}_t, t) \tag{3.12}$$

As highlighted in the *Diffusion Self Guidance* work [27], the guidance term in diffusion sampling need not come from a classifier's output. In fact, any energy function $g(\mathbf{x}_t, t, \mathbf{c})$

can be employed. Such a function may represent an approximate energy from an auxiliary model, a CLIP-based similarity score, an arbitrary time-independent energy as in universal guidance, or even spatial constraints such as bounding-box penalties on attention maps. Such additional guidance can be incorporated alongside the classifier-free guidance, enabling the model to produce high-quality samples that also attain low energy according to g:

$$\hat{\epsilon} = (1 + w)\epsilon_\theta(\mathbf{x}_t, t, \mathbf{c}) - w\epsilon_\theta(\mathbf{x}_t, t) + v\sqrt{1 - \alpha_t}\nabla_{\mathbf{x}_t} g(\mathbf{z}_t, t, \mathbf{c}) \tag{3.13}$$

where v is an additional guidance weight for g.

The core technique in Chapter 6 for generating multi-scale consistent keyframes can also be interpreted as applying a form of guidance, where the text-to-image diffusion model is steered to produce multiple zoom-consistent images jointly. Similarly, the diffusion sampling process proposed in Chapter 5 for generating videos between keyframes—by merging two diffusion models—can also be viewed as an instance of such guided generation.

Chapter 4

JUMP CUT SMOOTHING FOR TALKING HEADS: SEAMLESS TRANSITION SYNTHESIS

This chapter presents the research project *Jump Cut Smoothing for Talking Heads* [138][1] with Taesung Park, Yang Zhou, Eli Shechtman, and Richard Zhang. The subsequent analysis and comparisons to related studies in this chapter are based on the prevailing state-of-the-art during that time.

With the continuous advent of social media platforms, talking head videos have become increasingly popular. Such videos usually focus on the head and upper body of a person who narrates an idea, story or concept while looking at the camera or an interviewer. Frequently, the subject may use filler words ("uhh"), stutter, make an unwanted pause, or repeat words. Directly removing these frames produces unnatural jump cuts, which can range from half a second to minutes long. In such time, large body movements, head movements, and hand gestures may occur. Can the video editor avoid presenting this unnatural experience to the viewer?

While some cuts can be smoothed through playing B-roll or kept intentionally for artistic choice, there lacks a robust tool for replacing the jump cut with a seamless transition. For example, Adobe's video editing tool, i.e., Premiere Pro includes a feature called *MorphCut*[2] (Fig. 4.1) to help create a smooth transition. However, it fails when there are relatively large motion change such as head pose change or hand gesture change, resulting in noticeable motion blurs and other visual artifacts in the generated transitions. In influential work,

[1] https://morphcut.github.io/

[2] https://helpx.adobe.com/premiere-pro/using/morph-cut.html

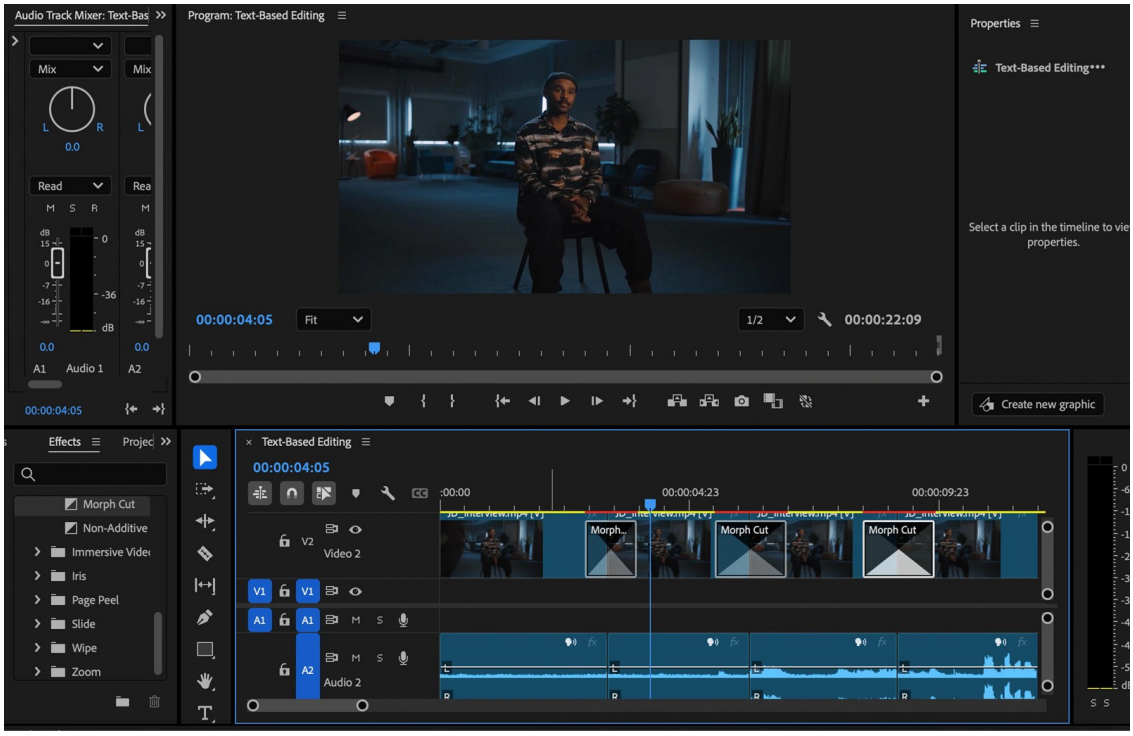


Figure 4.1: **Morph Cut in Adobe Premiere Pro** the Morph Cut effect from the Effect panel for a smooth transition between clips.

Berthouzoz et al. [7] generate hidden transitions by constructing a similarity graph between the raw frames and walking the graph. Such an approach is heavily limited by the contents of the video itself. Compared to traditional methods [7, 63], recent generative technology, offers the possibility of synthesizing new intermediate frames.

On the other hand, existing frame interpolation works utilize deep networks to perform video frame interpolation [87, 99] have greatly advanced, showing the impressive ability to increase video temporal sampling, creating slow motion videos. However, these methods are not designed to handle relatively large or complex motion changes in the talking head videos, such as large head rotation and body translation.

In this paper, we propose to readdress the jump cut smoothing by synthesizing new intermediate frames guided by the interpolated motion. We leverage a mid-level motion represen-

tation, i.e., interpolated DensePose [35] keypoints between the cut end frames, augmented with face landmarks, and formulate our learning problem as transferring the appearances of multiple source images to the given target DensePose keypoints. This is related to image animation using single/multiple source images through the transfer of the motion of a driving video. However, these methods either cannot interpolate the motion learned from the driving video via the unsupervised way due to a lack of semantic correspondence, or they suffer from serious identity preservation problems through the jump cut end frames (see Fig. 4.2). During the learning process, simply warping the source appearances based on DensePose keypoints correspondence often loses details and produces unfaithful images because of misalignment of the DensePose, disocclusion, and so on. Besides, when there are multiple source images available, the naive way to average the warping results often causes blurry artifacts. Therefore we introduce a cross model attention mechanism to improve the dense correspondence and help pick the appropriate source among multiple sources for each location in the target representation, which achieves better warping results to be used in the image generation network.

To conclude, we present a novel algorithm to smooth jump cuts in talking head videos through motion guided re-synthesis, a non-trivial task that requires balancing between realistic motion interpolation and preserving identity. For improved identity preservation, a larger informational bottleneck is needed, such as more keypoints or larger latent codes. Yet, more latent codes make motion interpolation challenging. We navigated this by (1) using more reliable and denser DensePose keypoints and face landmarks, (2) designing cross model attention, derived from initial DensePose keypoints correspondence, improving warping and allowing the synthesis network attend to more frames from the video and pick the most suitable features, and (3) Employing smooth linear interpolation, as well as interpolation ablation augmentation, and training the model to handle missing correspondences between the jump cut end frames. Our experiments show that we can seamlessly bridge various jump cuts, even those with significant head movements.

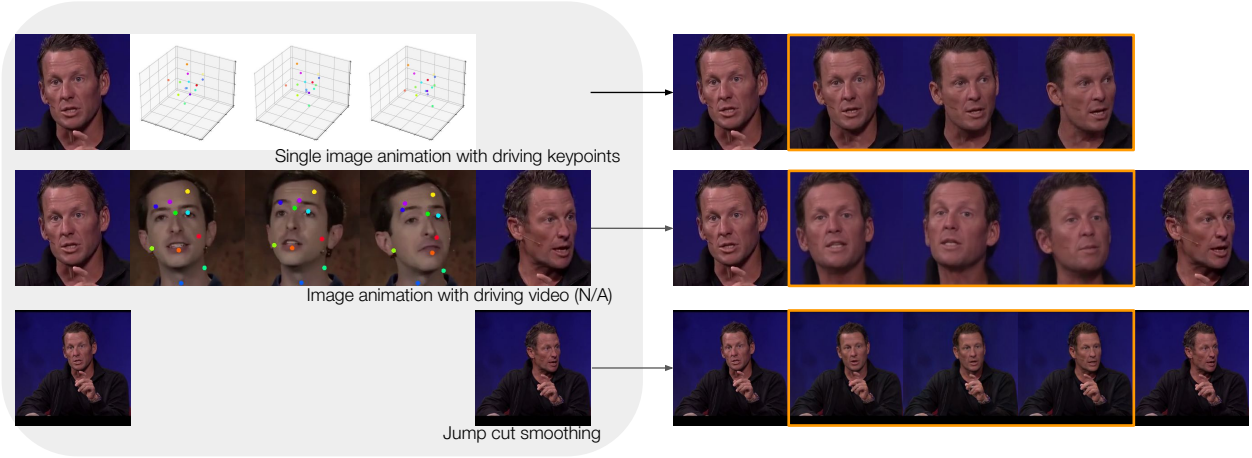


Figure 4.2: **Image animation methods cannot be applied for jump cut smoothing.** Row#1: Single image animation works (FaceVid2Vid [133], Face2Face$^{\rho}$ [150]) animate one of the cut end frames according to the key points sequence, neglecting the other end frame; Row#2: Other works (FOMM [112], ImplicitWarping [84]) require a driving video for motion extraction, which is absent in our scenario. Row#3: Our approach utilizes at least two cut end frames to generate the transition (shown in orange).

4.1 Related work

CNN-based frame interpolation. Frame interpolation, i.e., synthesizing intermediate images between a pair of input frames, is a long-standing research area in computer vision. Example applications include temporal up-sampling to increase refresh rate, create slow-motion videos and so on. Kernel-based methods [87] formulate frame interpolation as a convolution process, and they estimate spatially-adaptive convolution kernels for each output pixel and convolve the kernels with the input frames to generate a new frame. Flow-based methods [99, 91, 86, 72, 90, 50, 56] first estimate optical flow between the input frames and then synthesize the middle images guided by the flow via either warping or splatting techniques. These methods have demonstrated impressive results for input frames with small motion change. However, in our talking head video editing situation where we cut

out various lengths of filler words, unnecessary pauses, repeated words and so on, various motion change might happen in the jump cut such as large head rotation. This poses a major challenge for existing frame interpolation methods. Recently, [99] proposed to use multi-scale feature extractor and present a "scale-agnostic" bidirectional flow estimation module to handle large motion between duplicate photos, and has achieved state-of-the-art results in frame interpolation. Even so, it cannot address large head rotation, translation, and complex motion change between the two frames. Different from these flow based methods, we tackle this problem with DensePose key point guided image synthesis, which gives flexibility and controlability over the synthesized intermediate images, and our framework allows us to utilize other additional frames in the video.

Image animation with driving video. Our technique is related to the works [84, 133, 24, 153, 111, 112, 113, 150, 25] in the image animation domain, which generates videos by animating a single source image of a subject using the motions of a driving video possibly containing a different subject. Typically, these techniques first extract motion representation from the driving video in an unsupervised way, and "warp" the source image feature to the learned motion for synthesis. However, popular methods like FOMM [112], Mega-Portraits [25] and ImplicitWarping [84] often produce latent motions that lack semantic correspondence (see Row#2 in Fig. 4.2), making them inapplicable for generating transition motions in our jump cut smoothing task. Even if the keypoints has semantic meaning, as seen in Face-Vid2Vid [133] which discerns 3D keypoints, head pose, and expression deformation, these methods grapple with identity preservation due to their reliance on a single image for animation (see Row#1 in Fig. 4.2). Recently [84] introduced Implicit Warping approach, which animates using multiple source images, leveraging cross-model attention to pick the most fitting keypoint features. Despite its impressive results, like FOMM [112], it mandates a driving video to guide the motion. This stipulation, coupled with our lack of visual signals for the intermediate images we want to generate, renders it inapplicable for our purpose.

Attention in computer vision. Given a query and a set of key-value pairs, the attention function outputs the value for the query, which is computed as a weighted sum of the values for the keys, and the weight is determined by the similarity between query and the corresponding key. [130] proposed a transformer network stacked of attention layers and has achieved remarkable success in the task of machine translation. Since then, a number of works extended such transformer networks to the compute vision domain for the task of image recognition [23, 80, 127, 135], image generation [53, 54, 29, 51, 92] and achieved state-of-the-art numbers on the benchmark. There are also works using cross-modal attention where the queries, keys, and values are computed with different modalities such as vision and text [152, 149]. More recently, diffusion models have gained significant success and roaring attention in high resolution image synthesis conditioning on various input modalities such as image to image and text to image. The image generator in the diffusion model uses UNet backbone with the cross-attention mechanism for the conditioning [101, 53, 54] and have proved great efficiency. In our method, we are doing cross attention between DensePose keypoint features of source and target, where they come from different poses. Upon this initial correspondence, our attention learns to pick the most relevant source among the set of source images for each location in image. The warped feature are further fed into the generator network to output the intermediate frame.

4.2 Method

We solve the jump cut smoothing problem in two stages (see Fig. 4.3): in the training stage, given a set of source images, our network learns to generate a target image with the corresponding DensePose keypoints as motion guidance. In the inference stage, we synthesize each intermediate frame between the jump cut end frames guided by the interpolated DensePose keypoints and other available frames in the video.

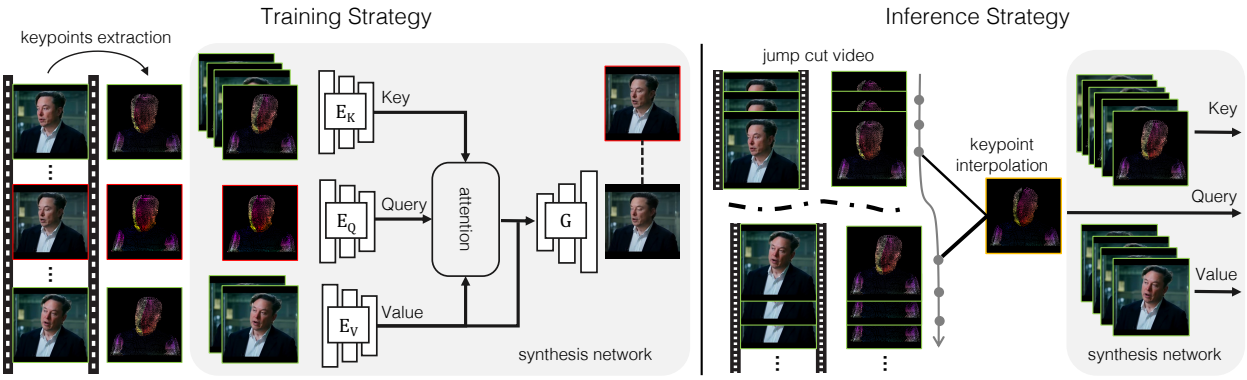


Figure 4.3: **Method overview.** In the training stage, we randomly sample source (denoted in green rectangle) and target (denoted in red rectangle) frames, and extract their corresponding DensePose keypoints augmented with facial landmarks (not shown here for simplicity). Our method extracts source dense keypoint features as *key*, target dense keypoint feature as *query*, and source image features as *value*, then a cross attention is applied to get the values for the *query*, i.e., warped feature. This warped feature is fed into the generator inspired from Co-Mod GAN to synthesize a realistic target image compared with the ground truth target frame. For applying jump cut smoothing in the inference stage, we interpolate dense keypoints between jump cut end frames, and synthesize the transition frame with the interpolated keypoints (in yellow rectangle) sequence.

4.2.1 DensePose keypoints representation

We use DensePose [35] keypoints augmented with face landmarks as motion guidance to synthesize the corresponding image. Given an input image I of size $(H \times W)$ and its continuous DensePose P, i.e., image-space UV coordinate map per body part, the DensePose keypoints $\{x_k\}_{k=1}^{K}$ are extracted from DensePose by quantizing the UV values. For each body part UV map, we discretize it into $n \times n$ cells with each cell representing a key point. We focus on the videos where only the upper body of a person appears, and thus there are $K = 14 \times n \times n$ DensePose keypoints per image (14 is the number of upper body parts in

then DensePose representation). The key point coordinates and UV value are the average of the pixel coordinates and UV values which fall into the corresponding cell. These keypoints with UVs are splatted into the grid of size $(H \times W)$ based on their coordinates, and then we get a discretized DensePose IUV. If there are not UV values in the DensePose that fall into the cell, the corresponding key point is not visible, and will not be splatted.

4.2.2 Cross model attention warping

Given a set of N source images $\{I_i\}_{i=1}^{N}$ with their respective DensePose keypoints $\{x_{i,k}\}$ and the target DensePose keypoints $\{x_{t,k}\}$, we aim at generating a realistic target image by transferring the appearances from the source images. The basic idea is that we utilize this dense correspondence to warp the source image features to the target dense key point, then the warped feature is fed into a generator similar as Co-Mod GAN [160] to generate the respective realistic target image. Therefore it is critical to get a high quality warped feature. However, the DensePose based correspondence is often inaccurate; in addition, in the case of multiple source images, it is more natural to select the one that is closer to the target image instead of doing naive averaging. We propose to use an attention mechanism to achieve this selection ability.

We adopt the commonly used scaled dot-product attention [130]. The input of the attention function consists of queries and keys of dimension d_k, and values of dimension d_v. We compute the dot products of the query with all keys, divide each by $\sqrt{d_k}$ and apply a softmax function to obtain the weights on the values. The weight can be interpreted as the similarity of the query and the corresponding key, then the output per query is a weighted average of the values. In practice, the set of queries are packed together into a matrix Q of size $n_q \times d_k$. The keys and values are also packed together into matrices K of size $n_k \times d_k$ and V of size $n_k \times d_v$. The matrix of outputs of size $n_q \times d_v$ are computed as:

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{4.1}$$

Q and K are the feature representation of target and source DensePose keypoints respectively,

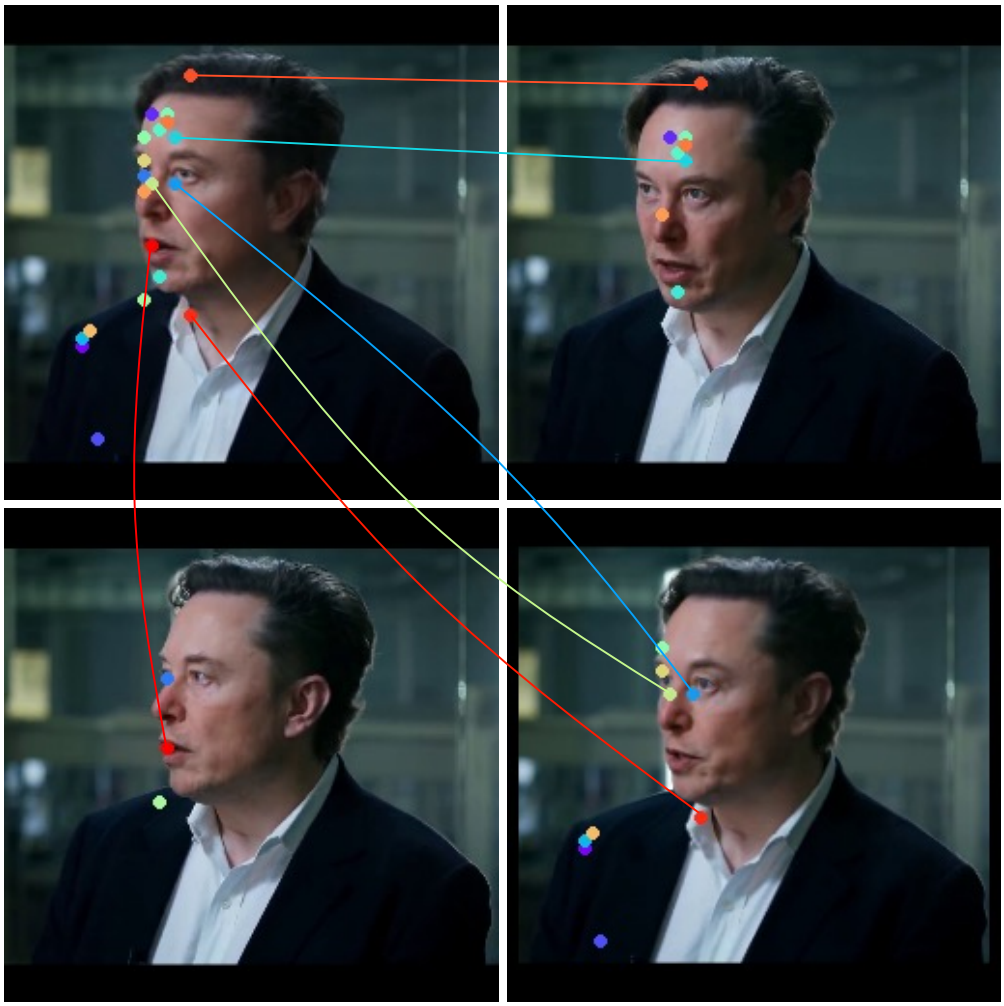


Figure 4.4: **Visualization of learned correspondence with our attention mechanism.** The top left is our synthesized image given the other three images as sources. We highlight the locations in the synthesized image where the peak attention score ≥ 0.75, and show their learned corresponding locations (marked with same color) in the source images. Our attention picks appropriate feature from different sources per location, e.g., for the blue point in the lower eyelid, our attention learned to associate with the eyelid feature in the bottom right source image.

and V is the appearance feature of source images. We use StyleGAN2 [61] based encoder with a final projection layer to map the feature to the dimension d_k for query and key. When encoding the source DensePose keypoints, we also concatenate it with the source image. The source image appearance features are encoded by the encoder with similar structure. All of these encoders output at 1/4 resolution of the inputs. For example, the image and discretized DensePose keypoints input are of size 256×256, the encoders produce $n_q = 64 \times 64$ queries, and $n_k = 64 \times 64 \times N$ keys for N number of sources. The warped image features are computed according to Eq. 4.1, and reshaped back to 64×64, which will be fed into the generator to produce the according target image.

With cross model attention based warping, on the one hand, the feature representation of the DensePose keypoints is more robust and can correct the misalignment of the DensePose. On the other hand, our method can pick the most relevant source features among multiple source images per location (see Fig. 4.4). This is especially useful for jump cut smoothing where we have frames from the entire video as sources.

4.2.3 Recursive synthesis for jump cut transitions

When creating the transition for jump cut smoothing, we firstly create a linearly interpolated dense keypoints sequence between the jump cut end frames, then each intermediate frame will be generated accordingly. This causes an issue when occlusion and disocclusion happen between the end frames. For example, when the speaker's head rotates from one side to the other side, part of the keypoints on the face disappears and another part of keypoints appears. We can only interpolate visible keypoints in both ends, which causes incomplete set of keypoints (see Fig. 4.10). We simulate this case in the training by only use visible keypoints in all source images for the target, and make the generator learn to inpaint the hole and generate a realistic image.

Since dense keypoints only model the foreground part, we additionally concatenate the two end frame features with the warped feature together as input to make the network learn to selectively copy the background from the end frames as well the remaining part that are

not modeled by the DensePose such as hair. We further proposed a recursive synthesis, as shown in Fig. 4.5, for a transition sequence of length T, we start the synthesis from the two frames that are closest to the end frame, i.e., I_1 and I_T, and then move towards the middle with the synthesized frames before as end frames to provide background information.

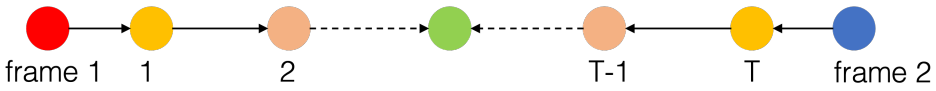


Figure 4.5: **Recursive synthesis.** To fill in a jump cut with smooth, intermediate frames, we recursively fill in frames from the end towards the middle.

4.2.4 Blended transition

Given a jump cut, we denote the frame before the cut as I_m and the frame after the cut as I_n, and their respective DensePose keypoints as $\{x_{m,k}\}$ and $\{x_{n,k}\}$, where $k = 1 \dots K$. K is the number of keypoints. We provide two ways to smooth the jump cut between I_m and I_n for a seamless transition.

The first way is to add a T number of intermediate frames between these two frames. The generation of each middle frame I_t is guided by the linearly interpolated DensePose keypoints $\{(1-\alpha_t)x_{m,k} + \alpha_t x_{n,k}\}$, with $\alpha_t = t/(T+1)$. However, accompanying new frames with silent audio results in an awkward, broken speech in some cases. For example, when a person is speaking quickly, but with short "umms" or "uhs" in the middle, if we remove these filler words and fill in with new frames with silence, the resulting video will sound inarticulate.

Thus, we provide another method, called *blended transition*, in order to avoid this audio artifact. Similarly to Zhou et al. [161], we synthesize blended frames to replace original frames around a small temporal neighborhood with a synthetic transition, so that the video can smoothly transit from frames before I_m to frames after I_n, as shown in Fig. 4.6.

We define the neighborhood using the frame range $[m-H, m]$ and $[n, n+H]$, with H

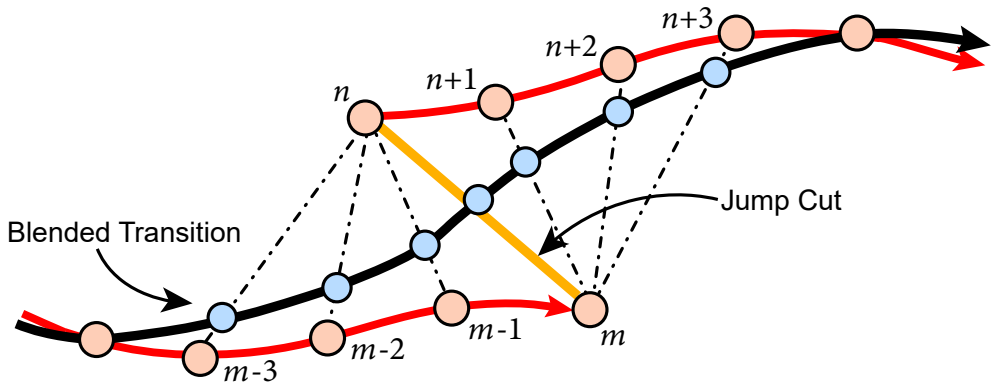


Figure 4.6: **Blended transitions.** We modify the frames before and after the jump cut, blending them smoothly. This allows us to smooth the jump cut without inserting additional frames.

as neighborhood size. We choose $H = 4$ in our experiments. Each frame I_i is blended with frame I_n, where $i \in [m - H, m]$, with weight $\alpha_i \in [0, \frac{1}{2H}, \ldots, \frac{1}{2}]$, with DensePose keypoints blended by $\{x'_{i,k} = (1 - \alpha_i)x_{i,k} + \alpha_i x_{n,k}\}$. Similarly, each frame $I_j, k \in [n, n + H]$ is blended with frame I_m with weight $\alpha_j \in [\frac{1}{2}, \frac{H+1}{2H}, \ldots, 1]$ and the corresponding DensePose keypoints are $\{x'_{j,k} = (1 - \alpha_k)x_{j,k} + \alpha_j x_{m,k}\}$. The blended frames are resynthesized and guided by the blended, dense keypoints. The blended transition does not change the existing number of frames in the video and thus does not suffer the audio insertion issue. Please refer to the project website for the video demo with audio for the filler words removal jump cut smoothing.

4.3 Experiments

We show our synthesized transition sequence under diverse jump cut situations. See Fig. 4.8 for selected examples. Our method successfully achieves seamless transition under challenging head pose changes such as extreme rotation or the back-and-forth movement of the head. We show our dataset collection and pre-processing in Sec. 4.3.1, and the comparison with baselines in Sec. 4.3.2. We further analyze how using more source frames from the video

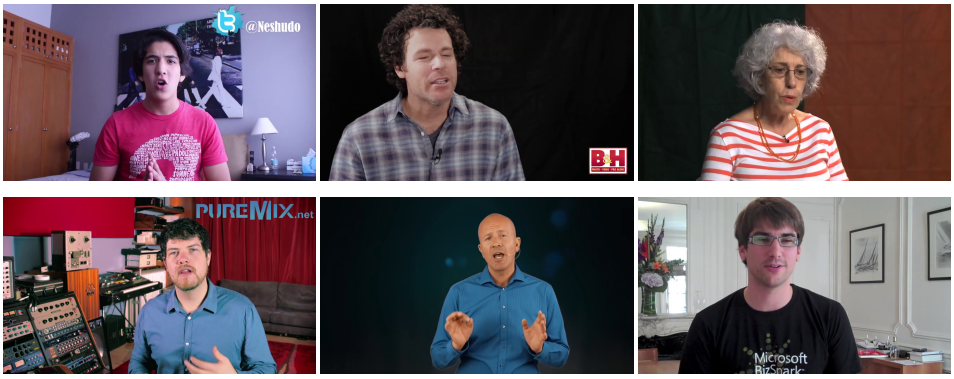


Figure 4.7: **Examples of our talking head videos.** We collect 600 talking head video clips and smooth jump cuts corresponding to filler words.

with our proposed attention mechanism improves synthesis quality in Sec. 4.3.3. Finally, we demonstrate how we apply our jump cut smoothing technique for removing filler words in talking head videos.

4.3.1 Dataset and pre-processing.

We target our jump cut smoothing application for the talking videos that contain upper torso. Note that this is a wider (and hence more challenging) cropping scheme than many existing talking head video datasets that only contain facial portions. Therefore, we collected 600 720p video clips from the raw AVSpeechDataset collection [26] for training and 50 videos for testing. Each video contains a person talking either facing towards the camera or the interviewer (see Fig. 4.7) for 10 to 20 seconds in a static background. While we do not add any manual annotation, we use Detectron2 [146] to detect the DensePose body part coordinate map per frame, and quantize the DensePose uv map to turn them into keypoints. For more accurate facial expression representation, we augment the DensePose keypoints with face landmarks using HRNet [121]. For training, we crop the person with the detected DensePose bounding box, and resize to 256×256. In the training stage, we randomly pick two sources and one target frame. At inference time, our method can be easily extended to

incorporate more source images (Section 4.3.3).

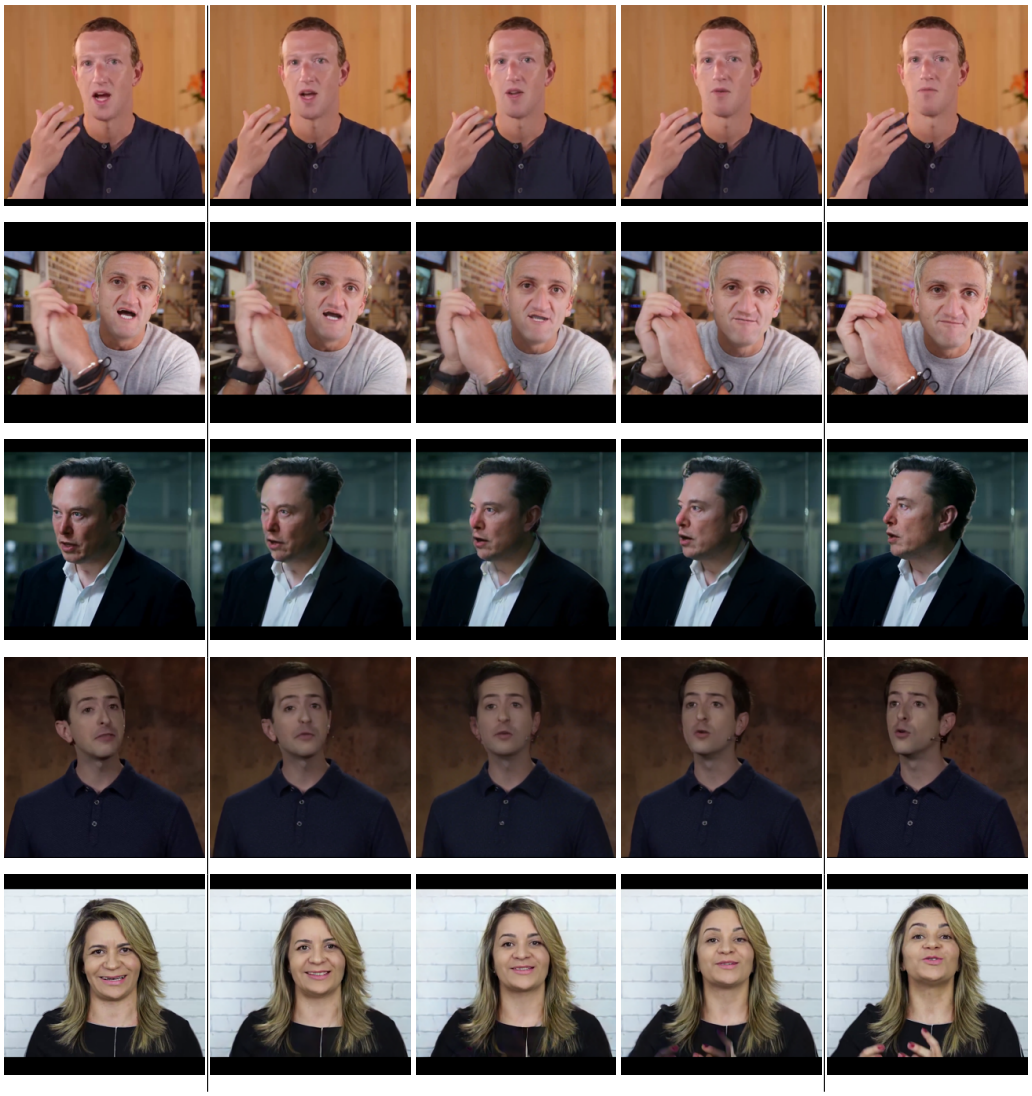


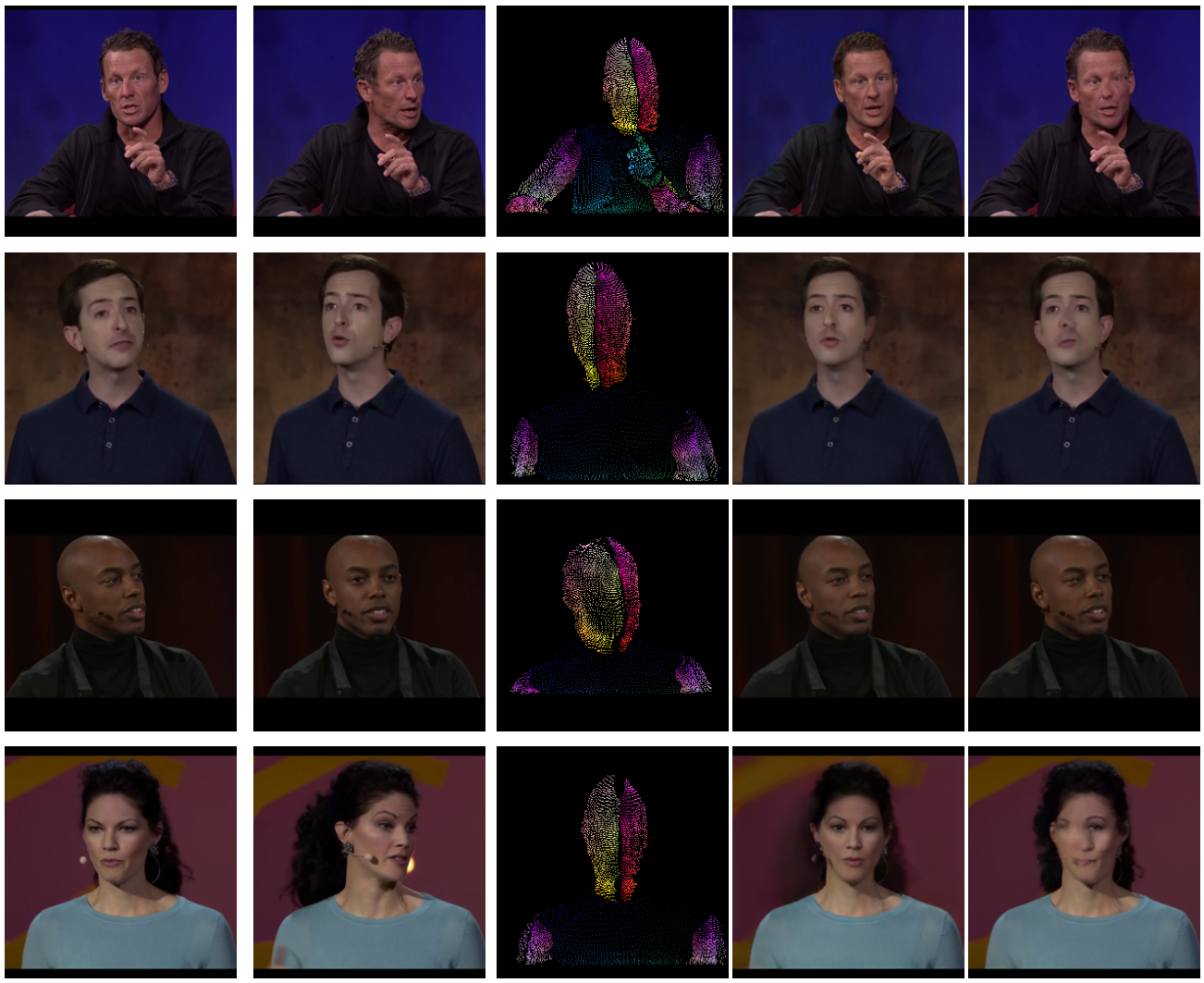
Figure 4.8: **Examples of our synthesized transition sequence** from the most left frame to the most right frame. Our method can create a seamless transition under different head pose changes. Row #2: head moving from back to front; Row #3: head rotating from front view to the side view; Row #4: head rotating from one side to another side. Row #5: head moving up and down.

Cut frame 1 Face2Face$^{\rho}$ Face-Vid2Vid Ours Cut frame 2

Figure 4.9: Comparison of synthesized frames just before the cut frame 2 by Face2Face$^{\rho}$, Face-Vid2Vid, and our method. Face2Face$^{\rho}$ generates distorted images, while Face-Vid2Vid loses sharpness and facial identity: microphones are gone and hair color does not match cut frame 2.

4.3.2 Evaluation

We show how image animation techniques *cannot* be applied to our jump cut smoothing problem in Fig. 4.2. More specifically, FOMM [112] and ImplicitWarping [84] cannot generate new transition frames *without* driving videos, as their latent motion representations lack 3D correspondence. The same key point could correspond to different head regions (see Row#2 in Fig. 4.2). Face-Vid2Vid [133] and Face2Face$^{\rho}$ animate only one image, and thus necessitate using just one of the end frames as the source. Fig. 4.9 presents the generated frame just before the second jump cut end frame. Evidently, the resulting images sacrifice facial identity and clarity, leading to another abrupt jump in the transition.



a) Jump cut frame 1 b) Jump cut frame 2 c) Our kpts d) Our results e) FILM results

Figure 4.10: **Qualitative comparison with FILM** for selected synthesized transition frame from jump cut end frame 1 to frame 2.

Methods	All			rotation ≥ 15°		rotation ≥ 30°		rotation ≥ 45°	
	FID [40]↓	PSNR↑	ArcFace [19]↑	PSNR↑	ArcFace [19]↑	PSNR↑	ArcFace [19]↑	PSNR↑	ArcFace [19]↑
FILM	4.99	25.37	0.51	25.68	0.39	25.65	0.33	25.27	0.23
Ours. + FILM kpts	**4.35**	25.29	0.51	25.57	0.41	25.55	0.35	25.23	0.25
Ours.	**4.53**	25.19	**0.52**	25.43	**0.45**	25.38	**0.40**	24.84	**0.31**

Table 4.1: **Quantitative comparison with FILM** in terms of synthesized frame realism and reconstruction fidelity. Here "rotation" means the head rotation along the yaw axis.' 'Ours+FILM kpts" means synthesize the image using DensePose keypoints interpolated with FILM. Our method outperforms FILM in terms of FID and ArcFace similarity, especially in the cases where head rotation happens in the jump cut.

Comparison to frame interpolation methods. We compare our method for jump cut smoothing against the state-of-the-art frame interpolation method FILM [99]. For testing, we randomly pick an equally spaced triplet, and run each method to synthesize the middle frame based on the two end frames. For our method, the keypoints of the middle frame are generated by linearly interpolating the DensePose keypoints of the two end frames. Furthermore, we also report a variant that runs FILM in the keypoint space, and then decodes the keypoints into pixels using our attention-based synthesis network. In Table 4.1, we measure the realism of the synthesized frames against the target frames with Frechet Inception Distance (FID) [40], using around 8000 test frames. In addition, we measure the fidelity of the synthesized image against the ground truth test frame with PSNR. We also measure how much the facial identity is preserved in the synthesized frames by reporting the ArcFace [19] cosine similarity between the facial embeddings of source and output frames. To further quantify how our method and FILM perform under different amounts of head pose changes, we categorize the jump cut end frame difference by the head rotation degrees along the yaw axis, and report the evaluation metrics in each category. As shown in Table 4.1, our method achieves lower FID score overall, meaning our method attains higher photorealism than

FILM. We also achieve higher ArcFace similarity across all rotation angles. In particular, our advantage grows larger at more extreme head pose changes (e.g., rotation $\geq 60°$). FILM, an optical flow-based method, likely fails to find correspondence when the head rotates from the front view (or more extremely one side view) to another side view, causing distortion in the synthesized face (see the rightmost column in Fig. 4.10 for reference). On the contrary, our method does not rely on the appearance feature to find correspondence, as we build upon the DensePose priors with attention mechanism to find better correspondence. Lastly, "Ours+FILM kpts" achieves the lowest FID score, likely because FILM does a better job in synthesizing complex objects like hands than linear interpolation.

4.3.3 Attention with more frames

One big advantage of our method compared to flow-based frame interpolation methods is that at test time, we can accept a larger number of input source frames in the video to assist the synthesis process. With the attention mechanism, our method can select the most appropriate feature per location among source frames. As shown in Fig. 4.4, different locations in the synthesize image correspond to their most relevant locations that are distributed among different sources. We find that this is especially helpful when certain parts are occluded in the two end frames, but are visible in the middle transition frames. For example, in Fig. 4.11, the speaker's mouth is widely open in one end frame and fully closed in the other end frame (see the images within red rectangle). If we only use these two end frames as source, the person's mouth in the generated middle frame looks unnatural, as there are no teeth features for a "half-open" mouth in the end frames. However, there are teeth features among the other video frames. We add 10 extra frames randomly chosen from the video, and our method now generates a high-quality middle frame, with teeth exposed in the half-opened mouth.

Figure 4.11: **Attention with more frames help improving the quality.** The top left two images within the red rectangle are the end frames of the jump cut. The top right image within green rectangle shows one of the 10 random images we add as additional sources to our attention. In the bottom row, we show our synthesized image using only the two end frames (left), and using the entire 12 source images (right). After adding 10 extra randomly chosen frames from the video, our method generates more accurate teeth representation, thanks to the attention mechanism that can incorporate variable number of source frames as reference.

4.3.4 Jump cut smoothing for filler words removal

We further demonstrate our method on filler words removal video editing. We collected several talking videos where the person stutters and then apply filler words detection algorithm [162] to cut out the filler words, resulting in unnatural jump cuts. We also manually remove some unwanted pauses and repetitive words to output a fluent talking video. Then we apply our jump cut smoothing with blended transition in Sec. 4.2.4 to the video.

4.4 Discussion

Our method can create smooth transitions under diverse jump cut cases, especially for the head movement. However, our method fails when there are complex hand gesture movements. Synthesizing realistic hand is even more challenging because 1.) the video frames with hand movement often have motion blur, which makes it hard for our network to discriminate real or fake hands, 2.) DensePose itself cannot model fine-grained hand features such as fingers, and 3.) The hands motion is more complicated than head. For example, when the speaker's hand moves from clenched to stretched, or from palm facing to back facing towards the camera, this non-planer motion cannot be modeled with linearly interpolated key points.

Chapter 5

DIFFUSION-BASED GENERATIVE KEYFRAME INBETWEENING

This chapter presents the research project *Generative Inbetweening: Adapting Image-to-Video Models for Keyframe Interpolation*[1] with Boyang Zhou, Brian Curless, Ira Kemelmacher-Shlizerman, Aleksander Holynski, and Steven M. Seitz. The findings of this work are published in ICLR 2025 [139]. The subsequent analysis and comparisons to related studies in this chapter are based on the prevailing state-of-the-art during that time.

Recent advances of large-scale text-to-video and image-to-video models [9, 8, 145, 147, 4, 154] have shown the ability to generate high resolution videos with dynamic motion. While these models can accept a variety of input conditioning signals, such as text captions or single images, most available models remain unsuitable for an obvious application: keyframe interpolation. Interpolating between a pair of keyframes—that is, producing a video that simulates coherent motion between two input frames, one defining the starting frame of the video, and one defining the ending frame—is certainly possible if a large-scale model has been trained to accept these particular two conditioning signals, but most open-source models have not. Despite the task's similarity to existing conditioning signals, creating an interpolation model requires further training, and therefore both large amounts of data and substantial computational resources beyond what most researchers have access to.

Given the similarity between the input signals needed for keyframe interpolation (i.e., two-frame conditioning) and the input signals to existing models (e.g., one-frame conditioning), an interesting alternative solution is to instead *adapt* an existing pre-trained image-to-video model, without training a specialized model from scratch. In this paper, we propose

[1] https://svd-keyframe-interpolation.github.io/

an approach for enabling keyframe interpolation by doing precisely this. Our approach is founded upon the observation that a keyframe interpolation model needs to know how to accomplish three objectives: (1) given a starting frame, it needs to predict coherent motion starting from that frame and advancing into the *future*, (2) given an ending frame, it needs to predict coherent motion starting from that frame and advancing backwards into the *past*, and (3) given these two predictions, produce a video that has a coherent combination of the two. Since existing image-to-video models can already accomplish the first of these three objectives, we focus our efforts on the the latter two, i.e., producing a single-frame conditioned model that can generate motion *backwards* in time, and a mechanism for combining forward and backward motion predictions into coherent videos.

One may imagine that producing such a single-image conditioned model that produces backwards motion should be trivial: simply pass an image into a regular image-to-video model, and reverse the output. Unfortunately, real-world motion is inherently asymmetric, and reversed motion into the future is notably different from motion into the past. As such, we first propose a novel, lightweight fine-tuning mechanism that reverses the arrow of time by rotating the temporal self-attention maps (i.e., reversing the temporal interactions) within the diffusion U-Net. This enables the reuse of the existing learned motion statistics in the pretrained model, and enables generalization while only requiring a small number of training videos.

Given both the original image-to-video model and this adapted reverse model, we also propose a sampling mechanism that merges the scores of both to produce a single consistent sample. These two sampling paths are synchronized through shared rotated temporal self-attention maps, ensuring they generate exactly opposite motions, an effect which we term "forward-backward motion consistency". At each sampling step, their intermediate noise predictions are fused, resulting in a generated video with coherent motion that starts and ends with the provided frames. We compare our work qualitatively and quantitatively to related methods on two curated difficult datasets targeted for generative inbetweening: Davis [96]

and Pexels[2], and our method produces notably higher quality videos with more coherent dynamics given distant keyframes.

5.1 Related work

Frame interpolation. Frame interpolation [22] synthesizes intermediate images between two frames by taking a pair of input frames or multiple adjacent frames in the context of video frame interpolation, and has been a long-standing research area in computer vision. Example applications include temporal up-sampling to increase refresh rate, create slow-motion videos, or interpolating between near-duplicate photos. Much of the research in this field [56, 86, 50, 90, 72, 91] employs flow-based methods, which estimate optical flow between the frames and then synthesize the middle images guided by the flow via either warping or splatting techniques. There are also works [58, 110] that use CNNs or transformers to learn to extract features and directly output the middle frames. Traditionally, this task assumes unambiguous motion and the input frames are usually closely spaced ($\leq$ 1/30s) samples in the video. Recent studies have begun to address large motions [114, 99], or quadratic motion [148, 79], though these still involve a single motion interpolation and cannot address distant input frames. In contrast, we aim to generate in-between frames that capture dynamic motions across distant input keyframes ($\geq$ 1s apart) with a generative model, a challenge that goes beyond the capability of current frame interpolation techniques.

Diffusion models for in-between video generation. Diffusion models have shown remarkable capabilities for generative modeling of images [42, 20, 117, 116, 118, 115] and videos [44, 41, 145, 9]. Early work MCVD [131] devises a general-purpose diffusion model for a range of video generative modeling tasks including in-between video generation. More recent works [36, 55, 147] explicitly train diffusion models to accept two end frames with conditioning to generate 7 or 16 intermediate frames at maximum resolution of 320×512 at once, and achieved superior results in generating dynamic motions. In this work, we focus on

[2]https://www.pexels.com/

adapting a pre-trained large-scale image-to-video model to do keyframe inbetweening without having to train or fine-tune from scratch. Exposed to millions of videos, these models have demonstrated remarkable capabilities in generating high-resolution (up to 1080p) and long (up to $4s$) videos with rich motion priors.

Diffusion sampling for consistent generation. In diffusion-based image generation tasks, novel joint diffusion sampling techniques [5, 158, 124, 74] for consistent generation are usually employed in generating arbitrary-sized images or panoramas from smaller pieces. These methods involve concurrently generating these multiple pieces and merging their intermediate results in the overlapping areas within the sampling process. For example, MultiDiffusion [5] averages the diffusion model predictions to reconcile the different denoising processes. Recent work, TRF [30] extends this joint generation approach to the bounded video generation taking two end frames as input. By running two parallel image-to-video generations guided by the start and end frames, it merge their outputs by averaging in each denoising step. However, a significant drawback of this method is that it cannot generate coherent motion in-between: simply fusing a forward video generation from the first end frame and the reversed forward video starting from the second end frame using a single image-to-video model designed for forward motion only causes the generated videos to oscillate between moving forward and then reversing, rather than continuously progress forward as our method does.

5.2 Background

We introduce some background on Stable Video Diffusion [8], the base image-to-video diffusion model used in our work, and then specifically explain the temporal self-attention layers within its architecture, which are key to modeling motion within the generated video.

5.2.1 Stable Video Diffusion

Diffusion models are trained to convert random noise into high-resolution images/videos via an iterative sampling process [20, 115, 117, 116, 118, 78]. This sampling process aims to reverse a fixed, time-dependent destructive process (forward process) that gradually corrupts data by adding Gaussian noise. In particular, Stable Video Diffusion (SVD) is a latent diffusion model where the diffusion process operates in the latent space of a pre-trained autoencoder with encoder $\mathcal{E}(\cdot)$ and decoder $\mathcal{D}(\cdot)$.

In the forward process, a video sample $\mathbf{x} = \{I_0, I_1, ..., I_{N-1}\}$ composed of N frames, is first encoded in the latent space $\mathbf{z} = \mathcal{E}(\mathbf{x})$, then the intermediate noisy video at time step t is created as $\mathbf{z}_t = \alpha_t \mathbf{z} + \sigma_t \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is Gaussian noise, and α_t and σ_t define a fixed noise schedule. The denoising network f_θ receives this noisy video latent $\mathbf{z}_t$ and the conditioning $\mathbf{c}$ computed from the input image, i.e., the first frame I_0 in the video, and is trained by minimizing the loss:

$$\mathcal{L}(\theta) = \mathbb{E}_{t \sim U[1,T], \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0},\mathbf{I})}[\| f_\theta(\mathbf{z}_t; t, \mathbf{c}) - \mathbf{y} \|_2^2]$$

where the target vector $\mathbf{y}$ here is $\mathbf{v} = \alpha_t \boldsymbol{\epsilon} - \sigma_t \mathbf{z}_t$, referred to as v-prediction.

Once the denoising network is trained, starting from pure noise $\mathbf{z}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, the sampling process iteratively denoises the noisy latent by predicting the noise in the input and then applying an update step to remove a portion of the estimated noise from the noisy latent

$$\mathbf{z}_{t-1} = \text{update}(\mathbf{z}_t, f_\theta(\mathbf{z}_t; t, \mathbf{c}); t)$$

until we get clean latent $\mathbf{z}_0$, followed by decoding $\mathcal{D}(\mathbf{z}_0)$ to get the generated video. The exact implementation of the update$(\cdot, \cdot)$ function depends on the specifics of the sampling method; SVD uses EDM sampler [59].

5.2.2 Temporal self-attention

The denoising network f_θ in SVD is a 3D U-Net, composed of "down", "mid", and "up" blocks. Each block contains spatial layers interleaved with temporal layers, with the temporal

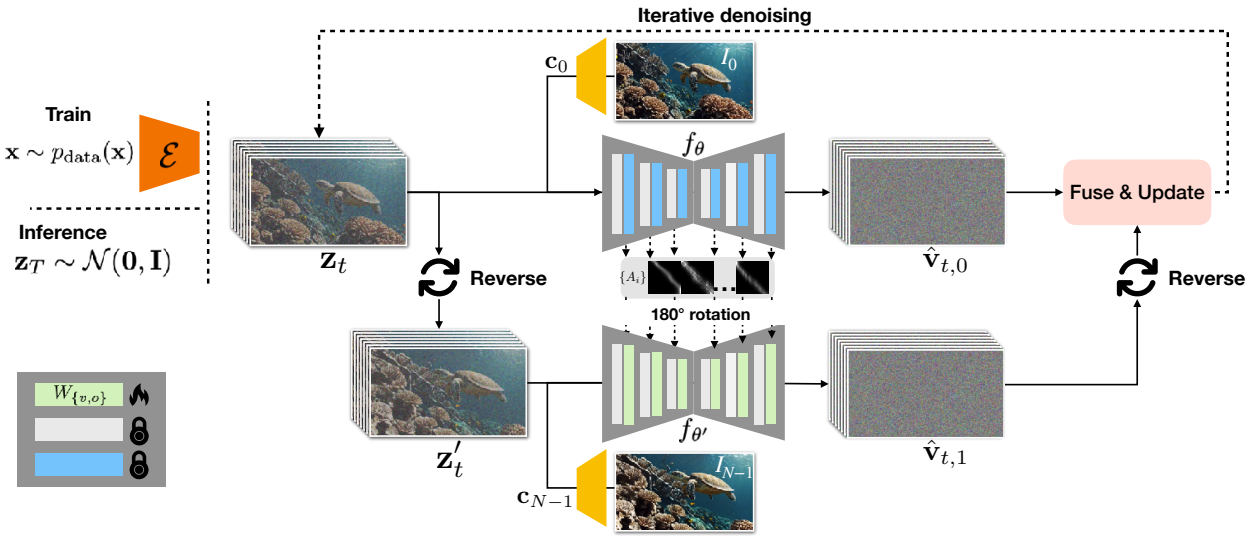


Figure 5.1: **Method overview.** In the lightweight backward motion fine-tuning stage, an input video $\mathbf{x} = \{I_0, I_1, ..., I_{N-1}\}$ is encoded into the latent space by $\mathcal{E}(\mathbf{x})$, and noise is added to create noisy latent $\mathbf{z}_t$; during inference, $\mathbf{z}_t$ is created by iterative denoising starting from $\mathbf{z}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. (1) **Forward motion prediction:** we first take the conditioning $\mathbf{c}_0$ of the first input image (inference stage) or the first frame in the video (training stage) I_0, along with the noisy latent $\mathbf{z}_t$ to feed into the pre-trained 3D U-Net f_θ to get the noise predictions $\hat{\mathbf{v}}_{t,0}$, as well as the temporal self attention maps $\{A_i\}$. (2) **Backward motion prediction:** We reverse the noisy latent $\mathbf{z}_t$ along temporal axis to get $\mathbf{z}'_t$. Then we take the conditioning $\mathbf{c}_{N-1}$ of the second input image, or the last frame in the video I_{N-1}, along with the 180-degree rotated temporal self-attention maps $\{A'_i\}$, and feed them through the fine-tuned 3D U-Net $f_{\theta'}$ for backward motion prediction $\hat{\mathbf{v}}_{t,1}$. (3) **Fuse and update:** The predicted backward motion noise is reversed again to fuse with the forward motion noise to create consistent motion path. Note that only the value and output projection matrices $W_{\{v,o\}}$ in the temporal self-attention layers (**green**) are fine-tuned; see Fig. 5.2 for more details.

self-attention layers responsible for modeling motion in the generated video. This layer takes a spatio-temporal tensor $X \in \mathbb{R}^{1\times N\times H\times W\times C}$ as input, where N is the number of frames,

and C is the number of channels. Here we use batch size of 1 for simplicity. The tensor is reshaped by moving the spatial dimensions (H, W) into the batch dimension. This creates $X' \in \mathbb{R}^{HW \times N \times C}$, where self-attention operates solely on the temporal axis. More specifically, X' is projected through three separate matrices W_q, $W_k, W_v \in \mathbb{R}^{d \times C}$ (d is the dimensionality of the projected space.), resulting in the corresponding query ($Q = W_q X'$), key ($K = W_k X'$) and value ($V = W_v X'$) features. Then the scale-dot product attention is applied:

$$\text{Attention}(Q, K, V) = \text{softmax}(QK^T/\sqrt{d})V$$

The attention output is fed through another linear layer W_o to get the final output. We refer to $A = QK^T \in \mathbb{R}^{HW \times N \times N}$ as the temporal self-attention map, which models the inter-frame correlations per spatial location. This temporal attention mechanism allows each frame's updated feature to gather information from other frames.

5.3 Method

Given a pair of keyframes I_0 and I_{N-1}, our goal is to generate a video $\{I_0, I_1, I_2, ..., I_{N-1}\}$ that begins with frame I_0 and ends with frame I_{N-1}, leveraging the pre-trained image-to-video Stable Video Diffusion (SVD) model. The generated video should exhibit a natural and consistent motion path, such as a car traveling or a person walking in a steady direction.

Image-to-video models typically generate video with motions that run forward in time. It is primarily the temporal self-attention layers that learn this motion-time association. In Sec 5.3.1, we discuss how this forward motion can be reversed by rotating the temporal self-attention maps by 180 degrees. Then we introduce an efficient lightweight fine-tuning technique to reverse this association and enable SVD to generate backward motion videos from the input image in Sec. 5.3.2. Finally we present our dual-directional sampling approach that fuses the forward motion generation starting with frame I_0 and backward motion video generation starting with frame I_{N-1} in a consistent manner in Sec. 5.3.3. An overview of our method is shown in Fig. 5.1.

5.3.1 Reverse motion-time association by self-attention map rotation

The temporal self-attention maps $\{A_i\}$ in the network f_θ feature the forward motion trajectory in video $\{I_0, I_1, ..., I_{N-1}\}$. By rotating these attention maps by 180 degrees, we obtain a new set $\{A'_i\}$ that depicts the opposite backward motion, corresponding to the reversed one $\{I_{N-1}, I_{N-2}, ..., I_0\}$ starting from the last frame I_{N-1}.

Specifically, rotating the temporal self-attention maps by 180 degrees—flipping them vertically and horizontally—yields a backward motion opposite to the original forward motion. For example, consider attention map A; the rotated map $A'_{N-j,N-k} = A_{j,k}$, where $A_{j,k}$ indicates the attention score between the j-th and k-th frames (I_j and I_k). In the corresponding reversed video, the reverse frame indices $N-j$ and $N-k$ maintain the same relative response.

5.3.2 Lightweight backward motion fine-tuning

We introduce a lightweight fine-tuning framework that specifically fine-tunes the value and output projection matrix W_v, W_o in the temporal self-attention layers, using the 180-degree rotated attention map from the forward video as additional input (see Fig. 5.2). We use $f_{\theta'}(\mathbf{z}_t; t, \mathbf{c}, \{A'_i\})$ to denote the backward motion generation network. This fine-tuning approach offers two key advantages:

First, by utilizing existing forward motion statistics from the pre-trained SVD model, fine-tuning $W_{\{v,o\}}$ simplifies the model's task to focus on learning how to synthesize reasonable content when operating in reverse. This strategy requires significantly less data and fewer parameters compared to full model fine-tuning. Second, it enables the control for the model to generate a backward motion trajectory corresponding to the opposite of the forward trajectory described by the attention map. This feature is particularly beneficial when planning to merge forward and backward motions converging towards each other, and thus achieving forward-backward consistency.

The detailed training process is shown in Alg. 1. For latent video $\mathbf{z} \in \mathbb{R}^{1\times N\times C\times H\times W}$, we denote flip($\mathbf{z}$) specifically by the second dimension, i.e., reversing the latent video along the

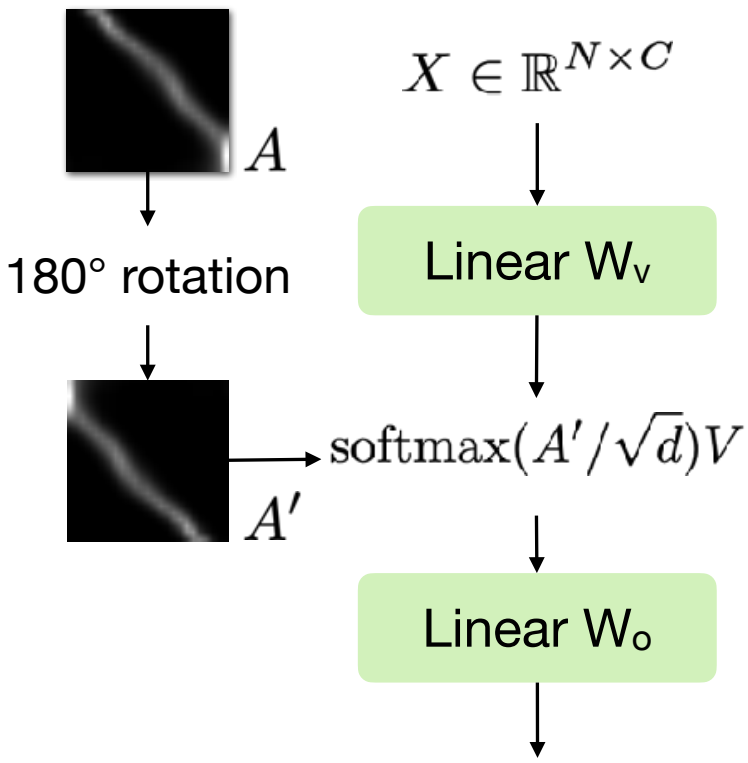


Figure 5.2: **Temporal self-attention module in the backward motion generation.** Given input tensor X, our attention mechanism additionally takes the respective attention map A from the pre-trained SVD featuring forward motion, rotating it by 180 degrees to create a reverse motion-time association A'. Note that $W_{\{v,o\}}$ are the only trainable parameters in this module.

time axis. In every training iteration, we sample an input video of N frames, and random time step t, then the noisy video latent $\mathbf{z}_t$ is created by adding the noise in that time step. The noisy video latent along with the input conditioning $\mathbf{c}_0$ (computed from I_0) is fed into the pre-trained 3D U-Net f_θ to extract the self attention maps $\{A_i\}$ from the temporal attention layers. Then we reverse the noisy video latent, along with the last frame conditioning $\mathbf{c}_{N-1}$, feed them into the backward motion 3D-U-Net $f_{\theta'}$. The loss function is computed by taking the predictions of the network and the ground truth reverse video.

Algorithm 1 Light-weight backward motion fine-tuning

Require: $f_\theta,\ p_{\text{data}}(\mathbf{x}),\ \mathcal{E}(\cdot)$

Ensure: $W_{\{v,o\}}$

while not converged **do**

  Sample $\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$; $\mathbf{x} = \{I_n\}_{n=0}^{N-1}$; $\mathbf{z} = \mathcal{E}(\mathbf{x})$

  Compute conditioning $\mathbf{c}_0$ from I_0

  Sample $t \sim \text{Uniform}(\{1, \ldots, T\})$; $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

  $\mathbf{z}_t \leftarrow \alpha_t \mathbf{z} + \sigma_t \boldsymbol{\epsilon}$

  $\{A_i\} \leftarrow \text{extract\_attention\_map}(f_\theta(\mathbf{z}_t; t, \mathbf{c}_0))$

  $\mathbf{z}'_t \leftarrow \text{flip}(\mathbf{z}_t)$

  Compute conditioning $\mathbf{c}_{N-1}$ from I_{N-1}

  Take gradient descent step on $\nabla_{W_{\{v,o\}}} \| f_{\theta'}(\mathbf{z}'_t; t, \mathbf{c}_{N-1}, \{A'_i\}) - \mathbf{y} \|_2^2$, where $\mathbf{y} = \alpha_t \text{flip}(\boldsymbol{\epsilon}) - \sigma_t \mathbf{z}'_t$

end while

5.3.3 Dual-directional sampling with forward-backward motion consistency

Our complete *dual-directional sampling* process is detailed in Alg. 2. Given a pair of keyframes I_0 and I_{N-1}, their corresponding conditioning $\mathbf{c}_0$ and $\mathbf{c}_{N-1}$ are pre-computed. Then each sampling step (illustrated in Fig. 5.1) works as follows:

(1) Forward motion denoising with I_0 as input: The noisy video latent z_t along with the conditioning $\mathbf{c}_0$ is fed into the pre-trained 3D U-Net f_θ in SVD to predict the noise volume $\hat{\mathbf{v}}_{t,0}$. Additionally, the temporal self-attention maps $\{A_i\}$ in the 3D U-Net are extracted.

(2) Backward motion denoising with I_{N-1} as input: The noisy video $\mathbf{z}_t$ is flipped along the temporal dimension to create the reverse video latent $\mathbf{z}'_t$ corresponding to the backward motion. This backward video, along with the conditioning $\mathbf{c}_{N-1}$, as well as the 180-degree rotated attention maps $\{A'_i\}$, are fed into our fine-tuned 3D U-Net $f_{\theta'}$. This step predict the noise volume $\hat{\mathbf{v}}_{t,1}$ representing a reverse motion from I_{N-1}.

(3) Finally, the predicted noise volumes from both forward and reverse motion paths are fused and then denoised using the update$(\cdot,\cdot)$ function to create less noisy video $\mathbf{z}_{t-1}$. In this way, we ensure forward-backward consistency and thus a consistent moving direction in the generated video. The fuse$(\cdot,\cdot)$ function performs a simple average. In practice, we also adopt per-step recurrence to enhance the fusion as seen in [2, 30], by re-injecting Gaussian noise into the update $\mathbf{z}_{t-1}$ and repeating the denoising 5 times before continuing the sampling for the next step.

Algorithm 2 Dual-directional diffusion sampling

Require: $I_0, I_{N-1}, f_\theta, f_{\theta'}, \mathcal{D}(\cdot)$

Compute conditions $\mathbf{c}_0$ and $\mathbf{c}_{N-1}$ from I_0 and I_{N-1}
Set $\mathbf{z}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T$ down to 1 **do**
    $\hat{\mathbf{v}}_{t,0} \leftarrow f_\theta(\mathbf{z}_t; t, \mathbf{c}_0)$
    $\{A_i\} \leftarrow$ extract_attention_map$(f_\theta(\mathbf{z}_t; t, \mathbf{c}_0))$
    $\mathbf{z}'_t \leftarrow \text{flip}(\mathbf{z}_t)$
    $\hat{\mathbf{v}}_{t,1} \leftarrow f_{\theta'}(\mathbf{z}'_t; t, \mathbf{c}_{N-1}, \{A'_i\})$
    $\hat{\mathbf{v}}'_{t,1} \leftarrow \text{flip}(\hat{\mathbf{v}}_{t,1})$
    $\hat{\mathbf{v}}_t \leftarrow \text{fuse}(\hat{\mathbf{v}}_{t,0}, \hat{\mathbf{v}}'_{t,1})$
    $\mathbf{z}_{t-1} \leftarrow \text{update}(\mathbf{z}_t, \hat{\mathbf{v}}_t; t)$
end for
return $\mathcal{D}(\mathbf{z}_0)$

5.3.4 Implementation Details

Our lightweight fine-tuning technique fine-tunes less than 2% of the U-Net parameters, and does not rely on large collection of training videos. So we collected 100 high quality videos

which are originally generated from SVD from a community website[3] as our training data. Our experimental results show that our method generalizes well to the real image data. We select the ones with large object motion such as animal running, vehicle moving, people walking, and so on. We use the Adam optimizer with learning rate of $1e-4$, $\beta_1 = 0.9, \beta_2 = 0.999$, and weight decay of $1e-2$. The training takes around $15K$ iterations with batch size of 4. We trained on 4 A100 GPUs. For sampling, we apply 50 sampling steps. For other parameters in SVD, we use the default values: *motion bucket id* $= 127$, *noise aug strength* $= 0.02$.

5.4 Experiments

In Figs. 5.3, 5.4, 5.5, we demonstrate that our approach successfully generates high quality videos with consistent motion given distant keyframes. We highly recommend viewing the videos in the project webpage to see the results more clearly. Sec. 5.4.1 describes the data we used to evaluate our method and the baselines. Sec. 5.4.2 demonstrates how our method outperforms traditional frame interpolation method FILM, and the recent work TRF [30] that also leverages SVD for video generation. Sec. 5.4.3 justifies our design decisions with an ablation study. Sec. 5.4.5 discusses the optimal scenarios where our method excels and sub-optimal ones where it outperforms the baselines but remains limited by SVD itself. Sec. 5.4.6 discusses our failure cases.

5.4.1 Evaluation Dataset

We use two high-resolution (1080p) datasets for evaluations: (1) The Davis dataset [96], where we create a total of 117 input pairs from all of the videos. This dataset mostly features subject articulated motions, such as animal or human motions. (2) The Pexels dataset, where we collect a total of 106 input keyframe pairs from a compiled collection of high resolution videos on Pexels[4], featuring directional dynamic scene motions such as vehicles moving,

[3] https://www.stablevideo.com/

[4] https://www.pexels.com/

animals, or people running, surfing, wave movements, and time-lapse videos. All input pairs are at least 25 frames apart and have the corresponding ground truth video clips.

5.4.2 Baseline Comparisons

We mainly compare our approach to FILM [99], the current state-of-the-art frame interpolation method for large motion, and TRF [30] which also adapts SVD for bounded generation. We show representative qualitative results in Figs. 5.3, 5.5. In addition, we also include results for the keyframe interpolation feature from the recent work DynamiCrafter [147]—a large-scale image-to-video model. The keyframing feature is modified from it and specially trained to accept two end frames as conditions, while we focus on how to *adapt* a pretrained image-to-video model in a lightweight way with small collection of training videos and much less computational resources. This feature generates videos at resolution 512×320, while ours generates at resolution 1024×576. Nonetheless, we present its results for reference.

Quantitative evaluation. For each dataset, we evaluate the generated in-between videos using FID [40] and FVD [31], widely used metrics for evaluating generative models. These two metrics measure the distance between the distributions of generated frames/videos and actual ones. The results are shown in Tab. 5.1, and our method outperforms **all** of the baselines by a significant margin.

Comparison to FILM. The flow-based frame interpolation method FILM suffers from two problems. First, it struggles to find correspondences in scenes with large motions. For example, in the second row of Fig. 5.3, in a highway where vehicles moving in both directions, FILM fails to find the correspondence between the moving cars across the input keyframes, resulting in implausible intermediate motions. For example, some cars in the first frame disappear in the middle and reappear at the end. Second, it generates undesirable unambiguous motion which takes the shortest path between the end frames. In the example in Fig. 5.5, given two similar-looking frames that captures different states of a person running, FILM produces a motion that merely translates the person across the frames, losing the

	Pexels		Davis	
	FID ↓	FVD ↓	FID ↓	FVD↓
FILM [100]	25.16	371.83	41.85	1048.65
TRF [30]	31.43	563.16	36.79	563.07
DynamiCrafter [147]	32.06	393.12	38.32	439.74
Ours w/o RA	26.42	458.76	36.70	549.98
Ours w/o FT	37.68	555.10	47.23	604.76
Ours	**22.99**	**306.84**	**32.68**	**424.69**

Table 5.1: Comparisons with baselines and our ablation variants. *Ours w/o RA*: full pipeline with fine-tuning all parameters $W_{\{q,k,v,o\}}$ without using the 180-degree rotated temporal attention map. *Ours w/o FT*: full pipeline using rotated attention map only in the "up" blocks and without fine-tuning $W_{\{v,o\}}$ for backward motion.

natural kinematic motions of the legs.

Comparison to TRF. TRF fuses the forward video generation starting from the first frame and the reversed forward video starting from the second frame, both using the original SVD. The reversed forward video from the second frame creates a backward motion video that ends at the second frame. Fusing these generation paths results in a back-and-forth motion in the generated videos. One notable effect we observe with TRF is that the generated videos exhibit a pattern of progressing forward first and then reversing to the end frame. For example, in the third row of Fig. 5.3, we can see the red truck moving backward over time; in the seventh row, the dog's legs are moving backwards, leading to unnatural motions. In contrast, our approach fine-tunes SVD to generate a backward video starting from the second frame in the opposite direction to the forward video from the first frame.This forward-backward motion consistency leads to the generation of a motion-consistent video.

5.4.3 Ablations

In Fig. 5.4 and Tab. 5.1, we show visual and quantitative comparisons to simpler versions of our method to evaluate the effect of the key components in our method.

Fine-tuning without rotated attention map (Ours w/o RA). We compare with a variant that fine-tunes all parameters in the temporal self-attention layers, namely, $W_{\{q,k,v,o\}}$, but without using the 180-degree rotated temporal self-attention map from the forward video as extra input. Though fine-tuning all parameters can generate backward motion from the second input image, there is no guarantee that the backward motion will mirror the forward motion from the first input image. This discrepancy makes it hard for the model to reconcile the two motion paths, often resulting in blending artifacts, as shown in the top row of Fig. 5.4. In contrast, fine-tuning $W_{\{v,o\}}$ with the rotated attention maps generates coherent and high-fidelity in-between videos.

Fine-tuning $W_{\{v,o\}}$ vs. no fine-tuning (Ours w/o FT). In Sec. 5.3.1, we show that rotating the temporal attention maps by 180 degrees reverses the motion-time association, creating a backward motion trajectory. Here we show that fine-tuning the value and output projection matrices $W_{v,o}$ is necessary for the model to synthesize high-fidelity content given the input backward motion-time association. We run our full pipeline without any fine-tuning, and our attention map rotation operation is only applied to the "up" blocks in this variant. As shown in the second row of Fig. 5.4 and Tab. 5.1, without fine-tuning these parameters, the model can create consistent motion but suffers from poor frame quality due to the low frame quality of the backward video generation. For example, the person is disfigured in the generated video. Note that applying the attention map rotation operation to the "down" and "mid" blocks in this variant worsens visual fidelity even further; thus, we show the best-case scenario without fine-tuning (i.e., applying rotated attention maps to the "up" blocks only).

5.4.4 Sensitivity to the scale of training set

As stated in Sec. 5.3.4, our method fine-tunes fewer than 2% parameters of the original model by using the attention map from the pretrained model, and thus we reduce the need for extensive training data. We use 100 synthetic training videos in our experiments. Here we we conduct an ablation by varying the training dataset size to be 50 and 150 videos, and evaluate the performance as done in Tab. 5.1. Our method still outperforms the baselines even with a training size of 50, and its performance increases slowly as more data is added (see Fig. 5.6).

5.4.5 Optimal and sub-optimal scenarios

Our method is limited by the motion quality and priors learned by SVD. Firstly, our empirical experiments indicate that SVD works well with generating rigid motions, but struggles with non-rigid, articulated movements. It has difficulty accurately rendering the limb movements of animal/people. In Fig. 5.5, though our method significantly improves upon FILM and TRF, it still appears unnatural compared to the ground truth movements. The bottom row, showing the sequence generated by SVD using only the first input frame, confirms that SVD itself struggles to generate natural running movements in between.

5.4.6 Failures

When the input pairs are captured at such distant intervals that they have sparse correspondences, as shown in Fig. 5.7, where only a small portion of cars appear in both input frames, it becomes difficult for our method to fuse the forward and backward motions. This situation, where the overlapping areas are minimal, leads to artifacts in the intermediate frames.

5.5 Discussions

Our method is limited by the motion quality of the underlying base model, Stable Video Diffusion (SVD), as discussed in Sec. 5.4.5. Another limitation is that SVD has strong motion priors derived from the input image, tending to generate only specific motions for a given input. As a result, the actual motion required to connect the input key frames may not be represented within SVD's motion space, making it challenging to synthesize plausible intermediate videos. However, with advancements in large scale image-to-video models like Sora[5], we are optimistic that these limitations can be addressed in the future. Including better motion datasets and incorporating articulated motion/physical movement priors may also help. Another potential improvement involves using motion heuristics between the input key frames to prompt the image-to-video model to generate more accurate in-between motions.

[5]https://openai.com/index/sora/

Figure 5.3: **Qualitative baseline comparisons.** Leftmost ($i = 0$) and rightmost columns ($i = 24$): start and end frames. The red arrow indicates the direction of motion.

Figure 5.4: **Ablation study.** We evaluate other options for generating in-between motion consistency. (1) *Ours w/o RA*: full pipeline with fine-tuning all parameters $W_{\{q,k,v,o\}}$ in the temporal attention layers but without using 180-degree rotated temporal self-attention maps as extra input (top row). (2) *Ours w/o FT*: full pipeline without fine-tuning $W_{\{v,o\}}$ for backward motion (second row). The differences are highlighted in the red rectangle.

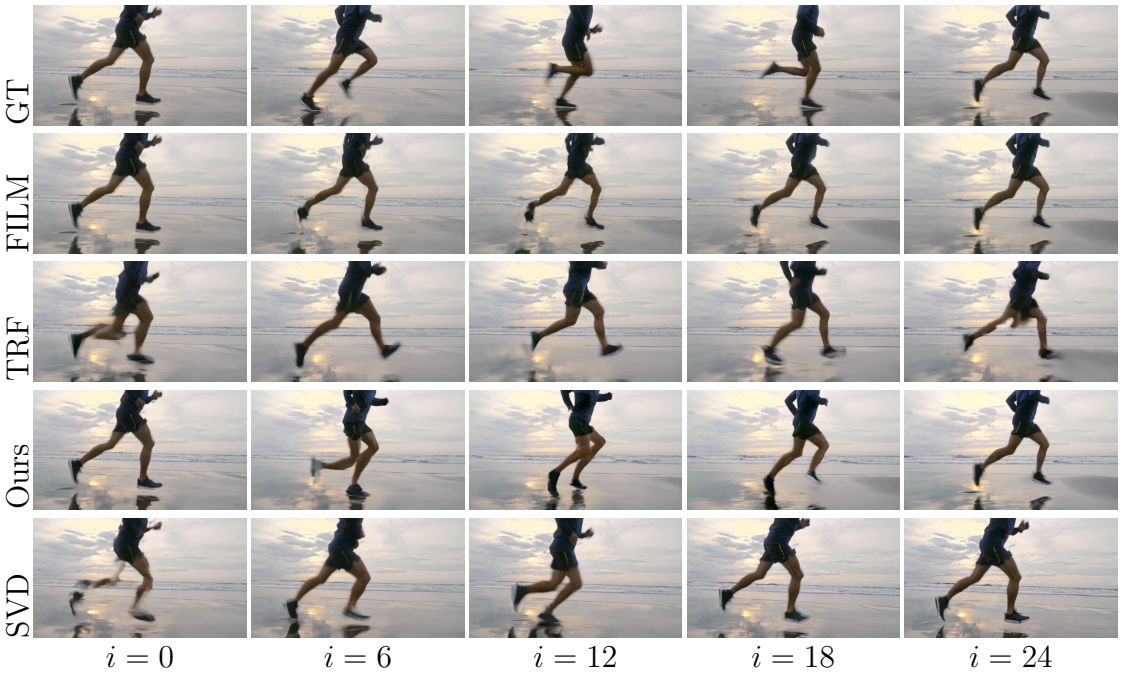


Figure 5.5: Our method outperforms FILM and TRF in generating articulated movements inbetween, but still struggles to create natural kinematic motions because of the limitation of SVD itself failing to generated complex kinematics (bottom row). Note that the input image serve as conditioning to SVD, so generated first frame might differ from the input image if SVD struggles to create plausible videos from that input.

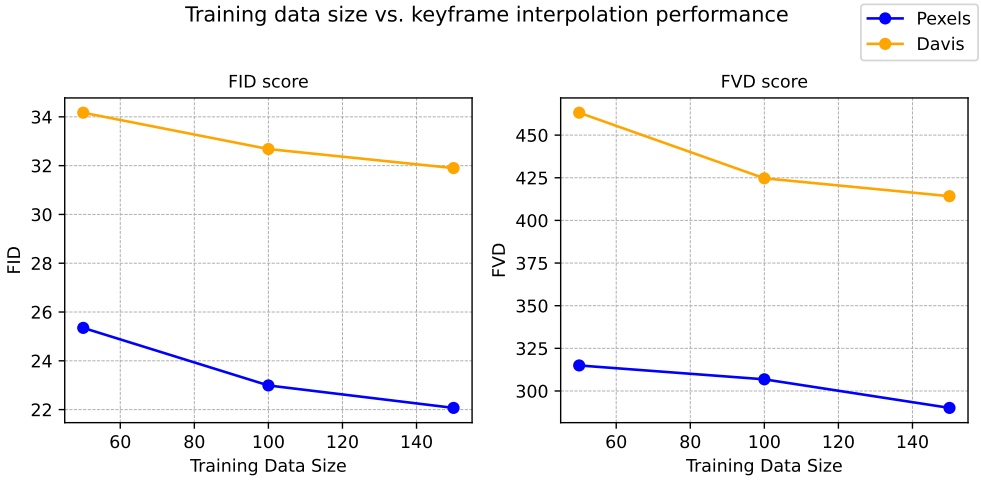


Figure 5.6: Ablation on how the scales of the training dataset affect our model's performance.

input frame 1 input frame 2 mid frame

Figure 5.7: **Failure case.** Our method fails to work well in the cases where input pairs have sparse correspondences.

Chapter 6

MULTI-SCALE KEYFRAMING FOR DEEP ZOOM GENERATION

Figure 6.1: **Powers of Ten (*1977*)**. This documentary film shows a continuous zoom from universe level (top left) to a single human lying on the grass (bottom right), and further into the hand and microscopic molecular structures. In this project, we aim to generate zoom videos with a similar effect.

This chapter presents the research project *Generative Powers of Ten*[1] with Janne Kontkanen, Brian Curless, Steven M. Seitz, Ira Kemelmacher-Shlizerman, Ben Mildenhall, Pratul P. Srinivasan, Dor Verbin, and Aleksander Holynski. The findings of this work are published in CVPR 2024 [137].

Recent text-to-image models [156, 11, 103, 104, 39, 27, 95] have been transformative in

[1]https://powers-of-10.github.io/

enabling applications like image generation from a single text prompt. While digital images exist at a fixed resolution, the real world can be experienced at many different levels of scale. Few things exemplify this better than the classic 1977 short film *"Powers of Ten"* (a few frames are shown in Fig. 6.1), which showcases the sheer magnitudes of scale that exist in the universe by visualizing a continuous zoom from the outermost depths of the galaxy to the cells inside our bodies[2]. Unfortunately, producing such animations or interactive experiences has traditionally required trained artists and many hours of tedious labor. Although we might want to replace this process with a generative model, existing methods have not yet demonstrated the ability to generate consistent content across multiple zoom levels.

Unlike traditional super-resolution methods, which generate higher-resolution content conditioned on the pixels of the original image, extreme zooms expose entirely new structures. For example, magnifying a hand should reveal its underlying skin cells, and thus generating such a zoom requires *semantic* knowledge of human anatomy. In this paper, we focus on solving this *semantic zoom* problem, *i.e.*, enabling text-conditioned multi-scale image generation, to create *Powers of Ten*-like zoom videos. As input, our method expects a series of text prompts that describe different scales of the scene, and produces as output a multi-scale image representation that can be explored interactively or rendered to a seamless zooming video. These text prompts can be user-defined (allowing for creative control over the content at different zoom levels) or crafted with the help of a large language model (*e.g.*, by querying the model with an image caption and a prompt like *"describe what might you see if you zoomed in by 2x"*).

At its core, our method relies on a joint sampling algorithm that uses a set of parallel diffusion sampling processes distributed across zoom levels. These sampling processes are coordinated to be consistent through an iterative frequency-band consolidation process, in which intermediate image predictions are combined across scales. Unlike existing approaches that accomplish similar goals by repeatedly increasing the effective image resolution (*e.g.*,

[2] https://www.youtube.com/watch?v=0fKBhvDjuy0

through super-resolution or image outpainting), our sampling process jointly optimizes for the content of all scales at once, allowing for both (1) plausible images at each scale and (2) consistent content across scales. Furthermore, existing methods are limited in their ability to explore wide ranges of scale, since they rely primarily on the input image content to determine the added details at subsequent zoom levels. In many cases, image patches contain insufficient contextual information to inform detail at deeper (*e.g.*, 10x or 100x) zoom levels. On the other hand, our method grounds each scale in a text prompt, allowing for new structures and content to be conceived across extreme zoom levels. In our experiments, we compare our work qualitatively to these existing methods, and demonstrate that the zoom videos that our method produces are notably more consistent. Finally, we showcase a number of ways in which our algorithm can be used, *e.g.*, by conditioning purely on text or grounding the generation in a known (real) image.

6.1 Prior work

Super-resolution and inpainting. Existing text-to-image based super resolution models [107, 1] and outpainting models [1, 105, 98, 123] can be adapted to the zoom task as autoregressive processes, *i.e.*, by progressively outpainting a zoomed-in image, or progressively super-resolving a zoomed-out image. One significant drawback of these approaches is that later-generated images have no influence on the previously generated ones, which can often lead to suboptimal results, as certain structures may be entirely incompatible with subsequent levels of detail, causing error accumulation across recurrent network applications.

Perpetual view generation. Starting from a single view RGB image, perpetual view generation methods like Infinite Nature [76] and InfiniteNature-Zero [77] learn to generate unbounded flythrough videos of natural scenes. These methods differ from our generative zoom in two key ways: (1) they translate the camera in 3D, causing a "fly-through" effect with perspective effects, rather than the "zoom in" our method produces, and (2) they synthesize the fly-through starting from a single image by progressivly inpainting unknown parts of novel views, wheras we generate the entire zoom sequence simultaneously and coherently

across scales, with text-guided semantic control.

Diffusion joint sampling for consistent generation. Recent research [5, 158, 124, 74] leverages pretrained diffusion models to generate arbitrary-sized images or panoramas from smaller pieces using joint diffusion processes. These processes involve concurrently generating these multiple images by merging their intermediate results within the sampling process. In particular, *DiffCollage* [158] introduces a factor graph formulation to express spatial constraints among these images, representing each image as a node, and overlapping areas with additional nodes. Each sampling step involves aggregating individual predictions based on the factor graph. For this to be possible, a given diffusion model needs to be finetuned for different factor nodes. Other works such as *MultiDiffusion* [5] reconciles different denoising steps by solving for a least squares optimal solution: *i.e.*, averaging the diffusion model predictions at overlapping areas. However, none of these approaches can be applied to our problem, where our jointly sampled images have spatial correspondence at vastly different spatial scales.

6.2 Method

Let $y_0, ..., y_{N-1}$ be a series of prompts describing a single scene at varying, corresponding zoom levels $p_0, ..., p_{N-1}$ forming a geometric progression, *i.e.*, $p_i = p^i$ (we typically set p to 2 or 4). Our objective is to generate a sequence of corresponding $H \times W \times C$ images $\mathbf{x}_0, ..., \mathbf{x}_{N-1}$ from an existing, pre-trained, text-to-image diffusion model. We aim to generate the entire set of images jointly in a zoom-consistent way. This means that the image $\mathbf{x}_i$ at any specific zoom level p_i, should be consistent with the center $H/p \times W/p$ crop of the zoomed-out image $\mathbf{x}_{i-1}$.

We propose a *multi-scale joint sampling* approach and a corresponding *zoom stack* representation that gets updated in the diffusion-based sampling process. In Sec. 6.2.1, we introduce our zoom stack representation and the process that allows us to render it into an image at any given zoom level. In Sec. 6.2.2, we present an approach for consolidating multiple diffusion estimates into this representation in a consistent way. Finally, in Sec. 6.2.3,

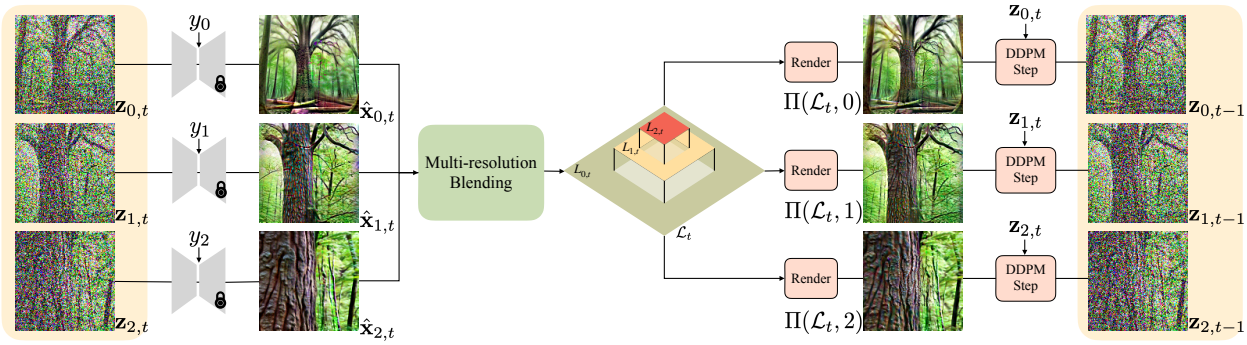


Figure 6.2: **Overview of a single sampling step.** (1) Noisy images $\mathbf{z}_{i,t}$ from each zoom level, along with the respective prompts y_i are simultaneously fed into the same pretrained diffusion model, returning estimates of the corresponding clean images $\hat{\mathbf{x}}_{i,t}$. These images may have inconsistent estimates for the overlapping regions that they all observe. We employ *multi-resolution blending* to fuse these regions into a consistent zoom stack $\mathcal{L}_t$ and re-render the different zoom levels from the consistent representation. These re-rendered images $\Pi_{\text{image}}(\mathcal{L}_t; i)$ are then used as the clean image estimates in the DDPM sampling step.

we show how these components are used in the complete sampling process.

6.2.1 Zoom Stack Representation

Our zoom stack representation, which we denote by $\mathcal{L} = (L_0, ..., L_{N-1})$, is designed to allow rendering images at any zoom level $p_0, ..., p_{N-1}$. The representation, illustrated in Fig. 6.3, contains N images of shape $H \times W$, one for each zoom level, where the ith image L_i stores the pixels corresponding to the ith zoom level p_i.

Image rendering. The rendering operator, which we denote by $\Pi_{\text{image}}(\mathcal{L}; i)$, takes a zoom stack $\mathcal{L}$ and returns the image at the ith zoom level $p_i = p^i$. We denote by $\mathcal{D}_i(\mathbf{x})$ the operator which downscales the image $\mathbf{x}$ by factor p_i, and zero-pads the image back to size $H \times W$; and we denote by M_i the corresponding $H \times W$ binary image which has value 1 at the center $H/p_i \times W/p_i$ patch and value 0 at padded pixels. The operator $\mathcal{D}_i$ operates

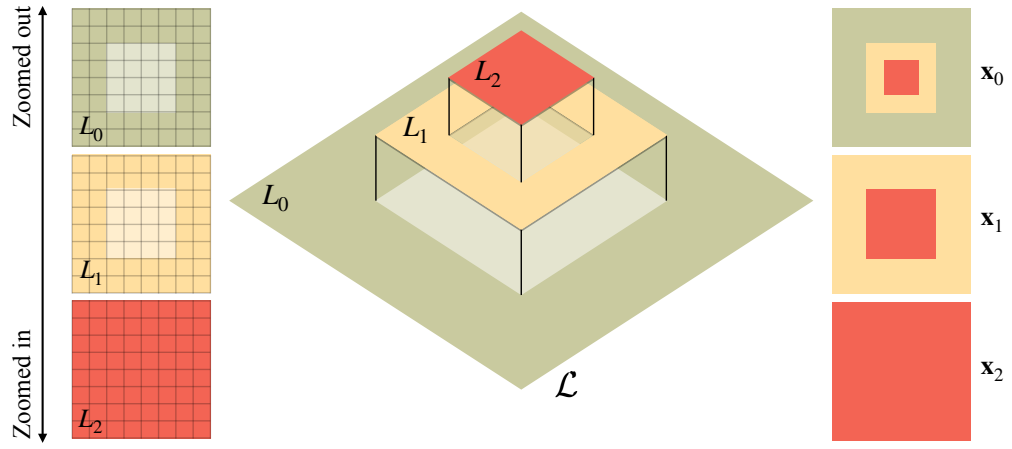


Figure 6.3: **Zoom stack.** Our representation consists of N layer images L_i of constant resolution (left). These layers are arranged in a pyramid-like structure, with layers representing finer details corresponding to a smaller spatial extent (middle). These layers are composited to form an image at any zoom level (right).

by prefiltering the image with a truncated Gaussian kernel of size $p_i \times p_i$ and resampling with a stride of p_i. As described in Alg. 3, an image $\mathbf{x}_i$ at the ith zoom level is rendered by starting with L_i, and iteratively replacing its central $H/p_j \times W/p_j$ crop with $\mathcal{D}_{j-i}(L_j)$, for $j = i+1, ..., N-1$. (In Alg. 3 we denote by $\odot$ the elementwise multiplication of a binary mask M with an image.) This process guarantees that rendering at different zoom levels will be consistent at overlapping central regions.

Noise rendering. At every denoising iteration of DDPM [42], each pixel is corrupted by globally-scaled i.i.d. Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Since we would like images rendered at different zoom levels to be consistent, it is essential to make sure the added noise is also consistent, with overlapping region across different zoom levels sharing the same noise structure. Therefore, we use a rendering operator similar to Π_{image} which converts a set of independent noise images, $\mathcal{E} = (E_0, ..., E_{N-1})$ into a single zoom-consistent noise $\boldsymbol{\epsilon}_i = \Pi_{\text{noise}}(\mathcal{E}; i)$. However, because downsampling involves prefiltering, which modifies the statistics of the resulting noise, we upscale the jth downscaled noise component by p_j/p_i to preserve the vari-

ance, ensuring that the noise satisfies the standard Gaussian distribution assumption, *i.e.*, that $\boldsymbol{\epsilon}_i = \Pi_{\text{noise}}(\mathcal{E}; i) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ for all levels i.

Algorithm 3 Image and noise rendering at scale i.

Set $\mathbf{x} \leftarrow L_i, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $j = i + 1, \ldots, N - 1$ **do**
    $\mathbf{x} \leftarrow M_{j-i} \odot \mathcal{D}_{j-i}(L_j) + (1 - M_{j-i}) \odot \mathbf{x}$
    $\boldsymbol{\epsilon} \leftarrow (p_j/p_i) M_{j-i} \odot \mathcal{D}_{j-i}(E_j) + (1 - M_{j-i}) \odot \boldsymbol{\epsilon}$
end for
return $\mathbf{x}, \boldsymbol{\epsilon}$

6.2.2 Multi-resolution blending

Equipped with a method for rendering a zoom stack and sampling noise at any given zoom level, we now describe a mechanism for integrating multiple observations of the same scene $\mathbf{x}_0, ..., \mathbf{x}_{N-1}$ at varying zoom levels $p_0, ..., p_{N-1}$ into a consistent zoom stack $\mathcal{L}$. This process is a necessary component of the consistent sampling process, as the diffusion model applied at various zoom levels will produce inconsistent content in the overlapping regions. Specifically, the jth zoom stack level L_j is used in rendering multiple images at all zoom levels $i \leq j$, and therefore its value should be consistent with multiple image observations (or diffusion model samples), namely $\{\mathbf{x}_i : i \leq j\}$. The simplest possible solution to this is to naïvely average the overlapping regions across all observations. This approach, however, results in blurry zoom stack images, since coarser-scale observations of overlapping regions contain fewer pixels, and therefore only lower-frequency information.

To solve this, we propose an approach we call *multi-resolution blending*, which uses Laplacian pyramids to selectively fuse the appropriate frequency bands of each observation level, which prevents aliasing as well as over-blurring. We show an outline of this process in Fig. 6.4. More concretely, to update the ith layer in the zoom stack, we begin by cropping all samples

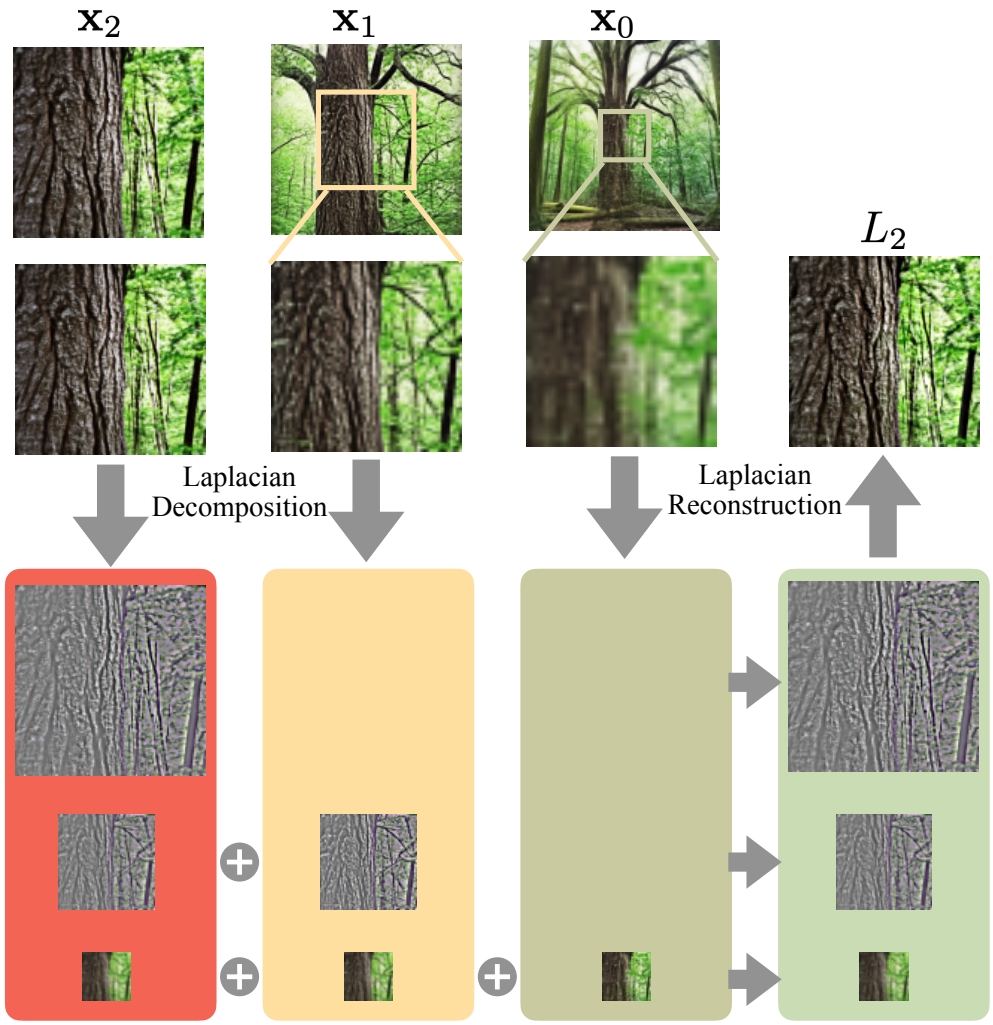


Figure 6.4: **Multi-resolution blending.** We produce a consistent estimate for Layer L_i in the zoom stack by merging the $H/p_j \times W/p_j$ central region of the corresponding zoomed out images $\mathbf{x}_j$ for $j \leq i$. This merging process involves (1) creating a Laplacian pyramid from each observation, and blending together the corresponding frequency bands to create a blended pyramid. This blended pyramid is recomposed into an image, which is used to update the layer L_i.

$j \geq i$ to match with the content of the ith level, and rescaling them back to $H \times W$. We then analyze each of these $N - i - 1$ images into a Laplacian pyramid [13], and average across corresponding frequency bands (see Fig. 6.4), resulting in an average Laplacian pyramid,

which can be recomposed into an image and assigned to the ith level of the zoom stack. This process is applied for each layer of the zoom stack L_i, collecting from all further zoomed-out levels $j \geq i$.

6.2.3 Multi-scale consistent sampling

Our complete *multi-scale joint sampling* process is shown in Alg. 4. Fig. 6.2 illustrates a single sampling step t: Noisy images $\mathbf{z}_{i,t}$ in each zoom level along with the respective prompt y_i are fed into the pretrained diffusion model in parallel to predict the noise $\hat{\boldsymbol{\epsilon}}_{i,t-1}$, and thus to compute the estimated clean images $\hat{\mathbf{x}}_{i,t}$. Equipped with our *multi-resolution blending* technique, the clean images are consolidated into a *zoom stack*, which is then rendered at all zoom levels, yielding consistent images $\Pi_{\text{image}}(\mathcal{L}_t; i)$. These images are then used in a DDPM update step along with the input $\mathbf{z}_t$ to compute the next $\mathbf{z}_{t-1}$.

6.2.4 Photograph-based Zoom

In addition to using text prompts to generate the entire zoom stack from scratch, our approach can also generate a sequence zooming into an existing photograph. Given the most zoomed-out input image $\boldsymbol{\xi}$, we still use Alg. 4, but we additionally update the denoised images to minimize the following loss function before every blending operation:

$$\ell(\hat{\mathbf{x}}_{0,t}, ..., \hat{\mathbf{x}}_{N-1,t}) = \sum_{i=0}^{N-1} \|\mathcal{D}_i(\hat{\mathbf{x}}_{i,t}) - M_i \odot \boldsymbol{\xi})\|_2^2, \tag{6.1}$$

where, as we defined in Sec. 6.2.1, $\mathcal{D}_i(\mathbf{x})$ downscales the image $\mathbf{x}$ by a factor p_i and pads the result back to $H \times W$, and M_i is a binary mask with 1 at the center $H/p_i \times W/p_i$ square and 0 otherwise. Before every blending operation we apply 5 Adam [66] steps at a learning rate of 0.1. This simple optimization-based strategy encourages the estimated clean images $\{\hat{\mathbf{x}}_{i,t-1}\}_{i=0}^{N-1}$ to match with the content provided in $\boldsymbol{\xi}$ in a zoom-consistent way. We show our generated photograph-based zoom sequences in Fig. 6.5.

Algorithm 4 Multi-scale joint sampling.

Set $\mathcal{L}_T \leftarrow \mathbf{0}$, $\mathbf{z}_{i,T} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\forall i = 0, ..., N-1$
for $t = T, \ldots, 1$ **do**
    $\mathcal{E} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    parfor $i = 0, \ldots, N-1$ **do**
        $\mathbf{x}_{i,t} = \Pi_{\text{image}}(\mathcal{L}_t; i)$
        $\boldsymbol{\epsilon}_i = \Pi_{\text{noise}}(\mathcal{E}; i)$
        $\mathbf{z}_{i,t-1} = \text{DDPM\_update}(\mathbf{z}_{i,t}, \mathbf{x}_{i,t}, \boldsymbol{\epsilon}_i)$
        $\hat{\boldsymbol{\epsilon}}_{i,t-1} = (1+\omega)\boldsymbol{\epsilon}_\theta(\mathbf{z}_{i,t-1}; t-1, y_i)$
                $-\omega\boldsymbol{\epsilon}_\theta(\mathbf{z}_{i,t-1}; t-1)$
        $\hat{\mathbf{x}}_{i,t-1} = (\mathbf{z}_{i,t-1} - \sigma_{t-1}\hat{\boldsymbol{\epsilon}}_{i,t-1})/\alpha_{t-1}$
    end parfor
    $\mathcal{L}_{t-1} \leftarrow \text{Blending}(\{\hat{\mathbf{x}}_{i,t-1}\}_{i=0}^{N-1})$
end for
return $\mathcal{L}_0$

6.2.5 Implementation Details

For the underlying text-to-image diffusion model, we use a version of Imagen [106] trained on internal data sources, which is a cascaded diffusion model consisting of (1) a base model conditioned on a text prompt embedding and (2) a super resolution model additionally conditioned the low resolution output from the base model. We use its default DDPM sampling procedure with 256 sampling steps, and we employ our *multi-scale joint sampling* to the base model only. We use the super resolution model to upsample each generated image independently.

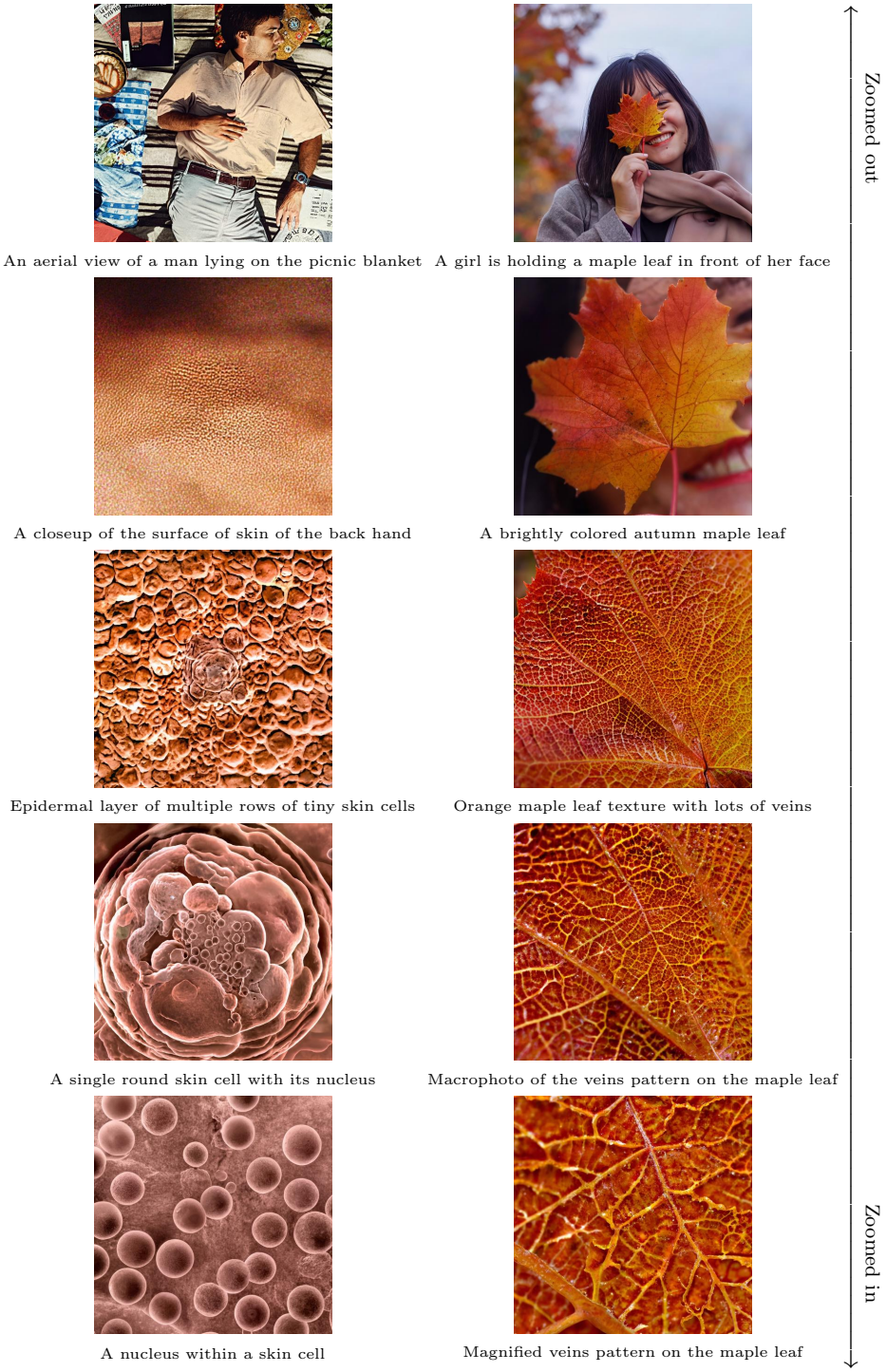


Figure 6.5: Selected images of our generated zoom sequences beginning with a provided real image. Left: Zoom from a man on a picnic blanket into the skin cells on his hand. Right: Zoom from a girl holding a leaf into the intricate vein patterns on the leaf.

Figure 6.6: Selected stills from our generated zoom videos (columns). Please refer to Sec. 6.3.1 for complete text prompts.

6.3 Experiments

In Figs. 6.5, 6.6, 6.7, 6.8, and 6.10, we demonstrate that our approach successfully generates consistent high quality zoom sequences for arbitrary relative zoom factors and a diverse set of scenes. Sec. 6.3.1 describes how we generate text prompts, Sec. 6.3.2 demonstrates how our method outperforms diffusion-based outpainting and super-resolution models, and Sec. 6.3.3 justifies our design decisions with an ablation study.

6.3.1 Text Prompt Generation

We generate a collection of text prompts that describe scenes at varying levels of scales using a combination of ChatGPT [89] and manual editing. For clarity, we use the GPT-4 model as deployed in ChatGPT. We start with prompting ChatGPT with a description of a scene, and asking it to formulate the sequence of prompts we might need for different zoom levels. While the results from this query are often plausible, they often (1) do not accurately match the corresponding requested scales, or (2) do not match the distribution of text prompts that the text-to-image model is able to most effectively generate. As such, we manually refine the prompts. We show a comparison of the prompts generated by ChatGPT and the corresponding manually refined prompts (which were used to generate our zooming videos) in Tab. 6.5 and Tab. 6.6. Some sequences were not generated automatically—these are shown in Tabs. 6.1, 6.2, 6.4, and 6.3.

In the future, we expect LLMs (and in particular, multimodal models) to automatically produce a sequence of prompts well suited for this application. In total, we collect a total of 10 examples, with the prompts sequence length varying form 6 to 16.

6.3.2 Baseline Comparisons

Fig. 6.7 compares zoom sequences generated with our method and without (*i.e.*, independently sampling each scale). When compared to our results, the independently-generated images similarly follow the text prompt, but clearly do not correspond to a single consistent

Figure 6.7: Generated zoom sequences with independent sampling (top) and our multi-scale sampling (bottom). Our method encourages different levels to depict a consistent underlying scene, while not compromising the image quality.

underlying scene.

Next, we compare our method to two autogressive generation approaches for generating zoom sequences: (1) Stable Diffusion's [1] outpainting model and (2) Stable Diffusion's "upscale" super-resolution model. We show representative qualitative results in Fig. 6.8. In Fig. 6.9, we compare with the super resolution model for photograph-based zoom.

Comparison to progressive outpainting. The outpainting baseline starts with generating the most zoomed-in image and progressively generates coarser scales by downsampling the previous generated image and outpainting the surrounding area. As in our method, the inpainting of each level is conditioned on the corresponding text prompt. In Fig. 6.8, we show that because of the causality of the autoregressive process, the outpainting approach suffers from gradually accumulating errors, *i.e.*, when a mistake is made at a given step, later outpainting iterations may struggle to produce a consistent image.

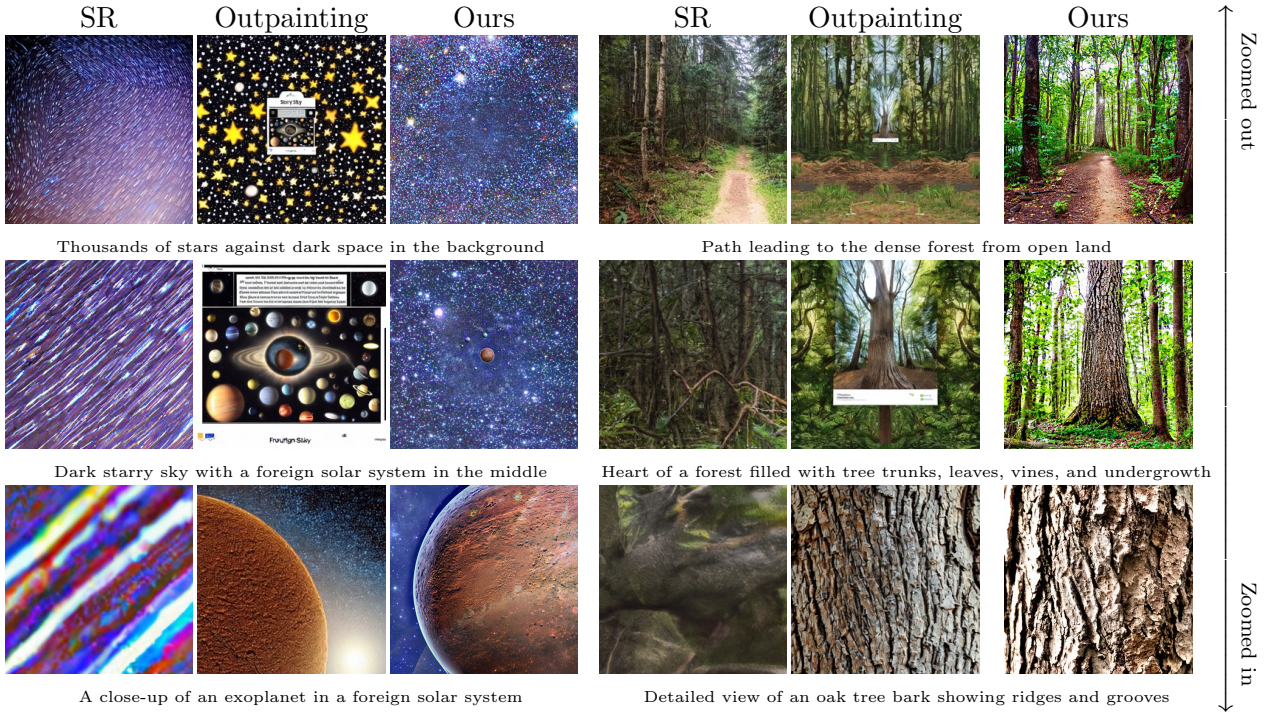


Figure 6.8: Comparisons with Stable Diffusion Outpainting and super-resolution (SR) models.

Comparison to progressive super-resolution. The super-resolution baseline starts with the most zoomed-out image and generates subsequent scales by super-resolving the upscaled central image region, conditioned on the corresponding text prompt. The low resolution input provides strong structural information which constrains the layout of the next zoomed-in image. As we can see in Fig. 6.8, this super-resolution baseline is not able to synthesize new objects that would only appear in the finer, zoomed-in scales.

6.3.3 Ablations

In Fig. 6.10, we show comparisons to simpler versions of our method to examine the effect of our design decisions.

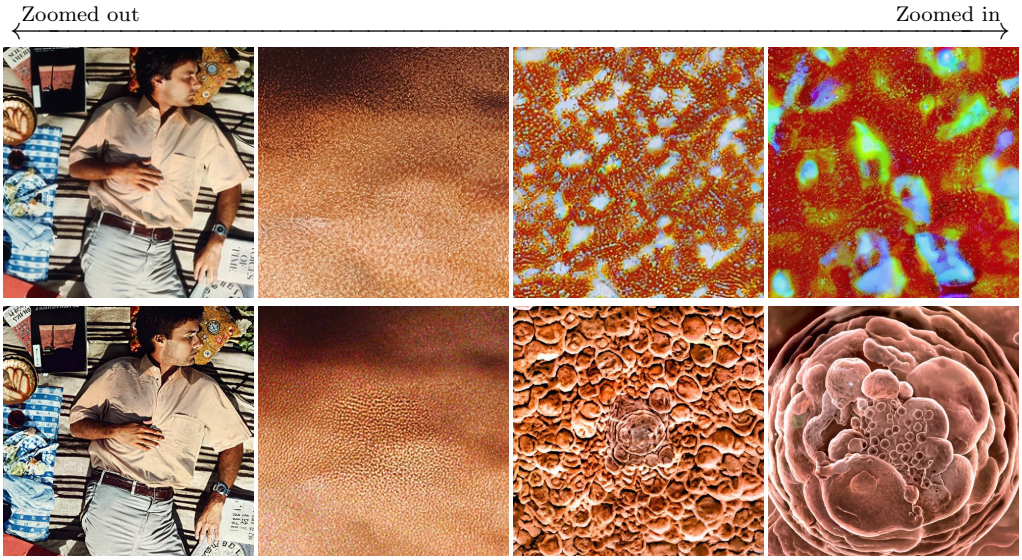


Figure 6.9: Comparison between the Stable Diffusion super-resolution (SR) model (top) and our method (bottom), zooming into a scene defined by a provided real input image (left).

Joint vs. Iterative update. Instead of performing multi-scale blending approach, we can instead iteratively cycle through the images in the zoom stack, and perform one sampling step at each level independently. Unlike fully independent sampling, this process does allow for sharing of information between scales, since the steps are still applied to renders from the zoom stack. We find that although this produces more consistent results than independent sampling, there remain inconsistencies at stack layer boundaries.

Shared vs. random noise Instead of using a shared noise Π_{noise}, noise can be sampled independently for each zoom level. We find that this leads to blur in the output samples.

Comparison with naïve blending. Instead of our multi-scale blending, we can instead naïvely blend the observations together, *e.g.*, as in MultiDiffusion [5]. We find that this leads to blurry outputs at deeper zoom levels.

6.4 Failure cases

Our method relies on the text-to-image diffusion model producing images of a scene at a particular set of scales from a particular viewpoint, and finding the exact set of text prompts that produce this can often be difficult. In Fig. 6.11, we show examples of cases where (1) the relative scale between a set of layers does not match the distribution of images that the model intends to create, and (2) the model intends to create images from different viewpoints across different zoom levels. As mentioned in the main paper, one possible improvement could be to optimize for suitable geometric transformations between successive zoom levels. These transformations could include translation, rotation, and even scale, to find better alignment between the zoom levels and the prompts.

6.5 Discussion

A significant challenge in our work is discovering the appropriate set of text prompts that (1) agree with each other across a set of fixed scales, and (2) can be effectively generated consistently by a given text-to-image model. One possible avenue of improvement could be to, along with sampling, optimize for suitable geometric transformations between successive zoom levels. These transformations could include translation, rotation, and even scale, to find better alignment between the zoom levels and the prompts.

Alternatively, one can optimize the text embeddings, to find better descriptions that correspond to subsequent zoom levels. Or, instead, use the LLM for in-the-loop generation, *i.e.*, by giving LLM the generated image content, and asking it to refine its prompts to produce images which are closer in correspondence given the set of pre-defined scales.

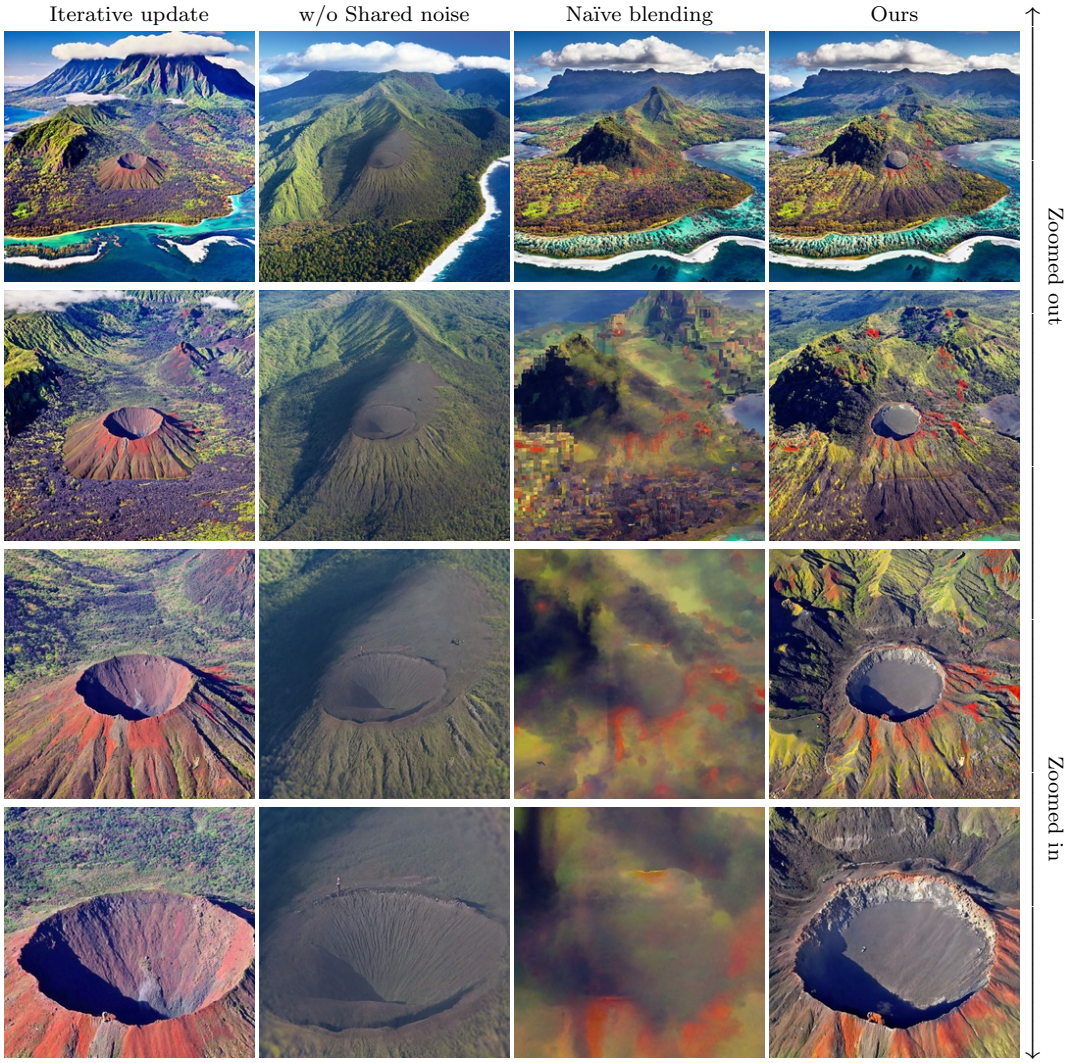


Figure 6.10: **Ablations.** We evaluate other options for multi-scale consistency: (1) iteratively updating each level separately, (2) naïve multi-scale blending, (3) removing the shared noise.

Figure 6.11: **Failure cases.** *Left*: an example where the predicted images from different levels observe the scene from different viewpoints (initially from a nearly horizontal view, but finally from an oblique upward-facing view). Right: an example where image priors do not correspond to the relative scale between zoom levels, as seen in the fact that multiple scales of the bark texture exist at a single zoom level.

• *A straight road in the middle with alpine forests on the sides under the blue sky with clouds; autumn season*
• *A photo capturing the tranquil serenity of a secluded alpine forest road with Mount Rainier in the far end; blue sky; autumn season*
• *A photo of serene alpine meadows against the massive Mount Rainier*
• *Extreme close-up of the steep cliffs and rocky outcrops of a snow mountain occupying the entire image; tight framing*
• *Extreme close-up of the steep cliffs and rocky outcrops of a snow mountain occupying the entire image; tight framing*
• *A team of climbers with red clothes climbing on the rugged cliffs; low camera angle*

Table 6.1: Complete prompts for the **Mount Rainier** example (column 4 in Fig. 7) with relative scale $p = 2$.

- *Small galaxy far away surrounded by large starry dark sky, millions of sparkling stars against dark background and vast emptiness*
- *Beautiful, high quality photo of Andromeda Galaxy*
- *Galactic core, tight framing*
- *Galactic core, tight framing*
- *Thousands of stars against dark space in the background*
- *Dark starry sky*
- *Dark starry sky with a foreign solar system in the middle*
- *Far view of alien solar system with a star and multiple exoplanets. Smaller stars in the background*
- *Alien solar system with one of the exoplanets in the center*
- *An exoplanet of a foreign solar system*
- *A close-up of an exoplanet in a foreign solar system, revealing a dry and arid climate*
- *Very high up top-down aerial image of deserted continents with reddish-hued soil in an alien planet revealing a dry and arid climate*
- *High up top-down aerial image of deserted continents with reddish-hued soil in an alien planet revealing a dry and arid climate*
- *Top-down photorealistic aerial image of a continent with a lot of deserts in an alien planet*
- *Top-down photorealistic aerial image of a desert with an alien outpost in the middle*
- *Top-down view of an alien outpost as seen directly above*

Table 6.2: Complete prompts for the **Galaxy** example (column 1 in Fig. 7) with relative scale $p = 2$.

- *A girl is holding a maple leaf in front of her face, partially obscuring it*
- *A brightly colored autumn maple leaf. The leaf is a rich blend of red and yellow hue and partially covering the face behind it; tight framing*
- *A brightly colored autumn maple leaf*
- *Orange maple leaf texture with lots of veins; macrophotography*
- *Macrophotograph showing the magnified veins pattern on the orange maple leaf texture; macrophotography*
- *High resolution macrophotograph showing the magnified veins pattern on the orange maple leaf texture; macrophotography*

Table 6.3: Complete prompts for the **Maple Leaf** example (column 2 in Fig. 6) with relative scale $p = 2$.

- *An aerial view of a man lying on the picnic blanket with his hand in the center of the image*
- *A close-up realistic photo showing the back side of a men's hand; uniform lighting; this lying person's hand should be put on top of light faded white shirt*
- *A close-up photo capturing the surface of skin of the back hand; uniform lighting*
- *Photo taken through a light microscope of skin's epidermal layer. The outermost layer, the stratum corneum, becomes apparent; Multiple rows of dense tiny skin cells becomes visible in the middle.*
- *Photo taken through a light microscope of a close up of skin's epidermal layer consisting multiple rows of dense tiny skin cells*
- *Photo taken through a light microscope showcasing several skin cells with similar sizes; with one cell in the center*
- *Photo taken through a light microscope of a single round skin cell with its nucleus in the center*
- *Photo taken through a light microscope of a nucleus within a single cell*

Table 6.4: Complete prompts for the **Hand** example (column 1 in Fig. 6) with relative scale $p = 4$.

ChatGPT generated	Manually refined
Forest, $p = 2$	
• *View of a vast forest from a hilltop*	<level removed in refinement>
• *Path leading to the dense forest from open land*	• *Path leading to the dense forest from open land*
• *Entrance of a forest with sunlight filtering through the trees*	• *Entrance of a forest leading into an oak tree in the middle with sunlight filtering through the trees*
• *Heart of a forest filled with tree trunks, leaves, vines, and undergrowth*	• *Heart of a forest with a tall oak tree in the middle, filled with tree trunks, leaves, vines, and undergrowth*
• *Single oak tree towering above the rest of the forest*	• *Textured tree trunk of a tall oak tree in the middle of a forest*
• *Close-up of a textured oak tree trunk and branches*	• *Close-up of a textured oak tree trunk in a forest*
<level added in refinement>	• *Close-up of a textured oak tree trunk in a forest*
• *Detailed view of an oak tree bark showing ridges and groove*	• *Detailed view of an oak tree bark showing ridges and groove*
• *Close-up of tree bark showing small cracks, lichen, and insects*	• *Close-up of tree bark showing small cracks, lichen, and insects*
Hawaii, $p = 2$	
• *An aerial photo capturing Hawaii's islands surrounded by the vast Pacific Ocean from above*	• *A aerial photo capturing Hawaii's islands surrounded by the vast Pacific Ocean from above*
• *An aerial photo showcasing Hawaii's rugged coastlines and pristine beaches*	• *An aerial photo showcasing Hawaii's rugged coastlines and pristine beaches*
• *An aerial photo revealing Hawaii's majestic mountains and lush rainforests*	• *An aerial photo revealing Hawaii's majestic mountains and lush rainforests*
• *An aerial shot of Hawaii's dramatic crater ridges and expansive lava fields*	• *An aerial shot of Hawaii's dramatic crater ridges and expansive lava fields*
• *Aerial view of surreal steam vents and sulphuric fumaroles within Hawaii's volcanic landscape*	• *An aerial close-up photo of the volcano's caldera*
• *Aerial perspective capturing the raw power and natural beauty of the volcano's caldera*	• *An aerial close-up photo of the rim of a volcano's caldera, with a man standing on the edge.*
<level added in refinement>	• *A top down shot of a man standing on the edge of a volcano's caldera, waving at the camera.*

Table 6.5: Generated prompts from ChatGPT vs. our manually refined prompts. We (1) removed prompts which are view inconsistent with others, (2) add more levels to make the relative scale correct, (3) add description to give more context about the entire scene.

ChatGPT generated	Manually refined
Sunflowers, $p = 2$	
• *A sunflower field from afar*	• *A sunflower field from afar*
<level added in refinement>	• *A sunflower field*
• *Move closer to the sunflower field; individual sunflowers becoming more defined, swaying gently in the breeze*	• *Close-up of rows of sunflowers of the same size facing front and swaying gently in the breeze; with one in the center*
• *Zooms in on a specific sunflower at the field's edge*	• *Zooms in on a single front-facing sunflower in the center at the field's edge*
• *Closer view of the sunflower. Emphasize the sunflower's golden petals and the intricate details*	• *Closer view of the sunflower in the center. Emphasize the sunflower's golden petals and the intricate details*
• *An image focusing solely on the center of the sunflower Showcase the dark, velvety disc florets, and capture the honey bee sipping nectar and transferring pollen*	• *An extreme close-up of the center of the sunflower Showcase the dark, velvety disc florets, and capture the honey bee sipping nectar and transferring pollen*
Earth, $p = 4$	
• *A distant view of Earth, showing continents and oceans*	• *Satellite image of the Earth's surface showing a landmass in the middle as seen from space*
• *Zooming in on a continent, with major geographical features visible*	• *Satellite image of landmass of the Earth's foggy surface*
• *A focused view on a specific region, highlighting rivers and landscapes*	• *Satellite image of a state in the U.S., showing the state's natural beauty with rivers, forests, and towns scattered across*
• *Narrowing down to a dense forest area, showcasing the canopy and terrain*	• *Satellite image of a quaint American countryside surrounded by forests and rivers in a foggy morning*
• *Zooming in on a specific lake, surrounded by the forest.*	• *Satellite image of a foggy forest with a lake in the middle shoot directly from above*
• *Close-up of the lake's surface, with surrounding vegetation*	• *Satellite image of a lake surrounded by a forest shoot directly from above*
• *Top-down view of a person kayaking in the lake, amidst the forest.*	• *Top down view of a lake with a person kayaking shoot directly from above*

Table 6.6: Generated prompts from ChatGPT vs. our manually refined prompts. We (1) removed prompts which are view inconsistent with others, (2) add more levels to make the relative scale correct, (3) add description to give more context about the entire scene.

Chapter 7

CHOREOGRAPHY-AWARE KEYFRAMING FOR ANIMAL DANCE GENERATION

This chapter presents the research project *How Animals Dance (When You're Not Looking)* [136][1] with Aleksander Holynski, Brian Curlss, Ira Kemelmacher-Shlizerman, and Steven M. Seitz.

> Everything in the universe has a rhythm; everything dances.
>
> —*Maya Angelou*

Humans dance spontaneously to music—just picture a toddler cheerfully bouncing to the beats at a birthday party. Animals can dance too; *Snowball the cockatoo*—can perform up to 14 distinct dance movements in response to different musical cues [62]. In fact, our animal friends are probably dancing all the time when we're not looking. In this paper, we capture this hidden world of animal dance, and expose it *for the first time* to the human eyes.

While we happen to be particularly obsessed with dancing animals, this paper introduces a new framework for generating music-synchronized, highly structured, up to 30 seconds long dance videos. Such capabilities are challenging for current state of the art generative models [8, 4, 154, 151], most of which are limited to short clips of a few seconds, do not produced synchronized audio and video, and lack intuitive controls for long range motion. Beyond text prompting, most controls for video generation are fine-grained and operate on a single frame at a time [142, 32], *e.g.*, body pose, camera pose, motion brushes, etc. In contrast, we introduce *choreography patterns* as a new control for video generation. Specifically, we allow the user to specify a structured sequence of dance moves, or "beats", *e.g.*, A-B-A-B-C-D-A,

[1] https://how-animals-dance.github.io/

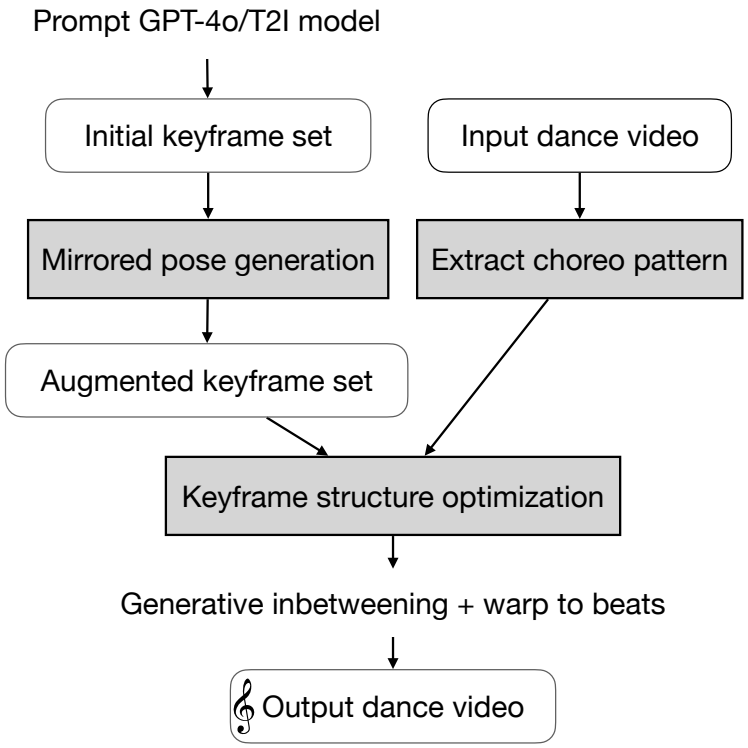


Figure 7.1: **System overview.** Given a few initially generated keyframes as input, we generate mirrored counterparts, extract choreography pattern from a dance video, and optimize the keyframe structure accordingly. The final dance is synthesized by generating in-between frames with a video diffusion model and warped to the musical beats. Our method is highlighted in gray.

where each letter corresponds to a particular move, and constrain the motion in the video to follow that choreography. Furthermore, we show how these choreography patterns can be automatically estimated from existing (human) dance videos.

Our use of choreography patterns as a control is inspired by how real dances are organized. A well-formed dance follows basic choreographic rules [14], which structure the movements to align with the rhythmic flow of the accompanying music, and often involve recurring patterns such as mirroring and repetition to help reinforce the musical structure [65, 64]. We leverage this inherent structure of dance to make the generation task more tractable. As input, we

use a collection of initially generated keyframes, each representing a distinct pose. We then formulate the dance synthesis as a graph optimization problem, *i.e.*, find the optimal walk path through the keyframes where the underlying motion satisfy a specified choreography pattern of beats. Each selected keyframe in the path is aligned to a musical beat. The final dance video is produced by synthesizing in-between frames between the keyframes using a generative video inbetweening model [139, 134] (Fig. 7.1).

Beyond 1) introducing a new type of generative video control (choreography patterns), and 2) a practical system for generating music-synchronized dance videos, this paper makes the following technical contributions. First, we introduce a technique for inferring choreography patterns from human dance videos, such as those found on Youtube and TikTok. Second, we formulate the satisfaction of these constraints as a graph optimization problem and solve it. Finally, we demonstrate an approach for pose-mirroring in the image domain, while retaining asymmetries in foreground and background features.

We demonstrate the effectiveness of our method by generating dance videos up to 30 seconds long across approximately 25 animal instances across 10 classes—including marmots, sea otters, hedgehogs, and cats—paired with various songs. These videos represent the first-ever recorded demonstrations of these animals performing such complex musical dance routines and will be studied by generations of zoologists.

7.1 Prior work

Music-driven generative dance synthesis. Earlier learning-based approaches developed neural networks that synthesize human dance motion directly from music input [73, 71, 49, 75, 157, 120]. Recent advances have shifted toward diffusion-based methods [97, 69, 129], which also focus primarily on generating skeletal motions from music. More recently, some works have begun exploring direct dance video generation using video diffusion models [120, 102, 45]. However, directly enforcing choreography structure within these frameworks remains challenging. In addition, these learning-based approaches typically require training data—an issue in our case, as dance videos featuring animals are extremely scarce.

Graph-based human dance motion synthesis. In contrast to learning-based approaches, graph-based frameworks [65, 64, 88, 85, 14] synthesize new motions from an existing dance motion segments database, and cast the dance synthesis as a graph optimization problem: finding an optimal path in the constructed motion graph that aligns with the input music. For example, Kim et al. [65] introduced rhythmic and beat-based constraints to guide the path search, while more recent work *ChoreoMaster* [14] incorporated richer choreography rules, requiring not only structural alignment with music but also stylistic compatibility.

Our approach follows this paradigm but differs in key ways. First, instead of relying on a motion capture database, we take a small set of keyframes of an animal or subject as input. We augment this set by generating mirrored pose images, creating a complete keyframe set for dance synthesis. The graph is constructed over these keyframes, and a video diffusion model is applied to generate realistic in-between frames along the optimized walk path, producing the final dance video. Second, while basic choreography rules can be inferred from the musical structure as done in [14], different performers may interpret the same piece differently. As such, we propose a way to extract choreography patterns directly from a reference dance video and use it as the control.

Anthropomorphic character animation. Given a reference image, character image animation, generates videos following a per-frame target human skeletal pose sequence. While existing methods [47, 48] are primarily designed for human figures, recent work [122] extends to anthropomorphic characters by learning generalized motion representation. However, it still favor human-like anatomy, often producing animals with features like elongated limbs and human-style body proportions. They also struggle to generate high-fidelity videos from a single image when handling long and diverse sequences, such as a 30-second dance. In contrast, our method does not rely on per-frame human skeleton pose as guidance and uses choreography pattern as higher-level control, letting the video diffusion model generate in-between motions so that the final dance follows the choreographic structure, not human motion itself.

7.2 Approach

We begin by generating a small set of keyframes $\{I_k\}_{k=0}^{K-1}$, each depicting the subject, *e.g.*, a marmot, in different poses while maintaining a consistent background and static camera view (see Section 7.3.1 for details). Our goal is to synthesize a dance video of the input animal from the provided keyframes, synchronized to the beats and following the choreography pattern extracted from a reference dance video.

We first introduce how we extract the choreography pattern directly from a human dance video in Section 7.2.1. Since motion mirroring is an essential component of dance, we then present our approach for generating a mirrored pose counterpart for each keyframe to augment the keyframe set in Section 7.2.2. Finally, in Section 7.2.3, we present how to synthesize the full dance video using the complete set of keyframes, including mirrored ones, to follow the choreography pattern.

7.2.1 Choreography Pattern Labeling

Choreography are closely tied to the rhythmic structure of music. In music theory, a *beat* is the basic temporal unit, while a *bar* (or measure) groups a fixed number of beats. The *meter* defines how beats are grouped and emphasized within the bar, and is indicated by a time signature, *e.g.*, 2/4, 3/4, 4/4, where the upper number specifies beats per bar, and the lower number denotes the note value that receives one beat. In this work, we focus on music with a 4/4 time signature—each bar contains four quarter note beats—the most common structure in popular music.

Problem definition. Given a 4/4 music track with a synchronized dance video, we begin by detecting the beat times $\mathcal{B} = \{t_0, t_1, ..., t_{N-1}\}$, assuming a total of N beats, corresponding to $\frac{N}{4}$ bars. Based on the beat times, we construct a sequence of motion segments $\mathcal{S} = \{s_0, s_1, ..., s_{\frac{N}{2}-1}\}$ where each segment s_i spans from beat t_{2i} to beat t_{2i+1}. Each bar thus yields two motion segments: one from the first to the second beat, and another from the third to the fourth beat, aligning with the 4/4 music structure where major movements

typically begin on accented beats and end on weak ones, whereas transitions occur across weak-to-accented intervals. The "choreo pattern" labeling task outputs a sequence of labels $\mathcal{L} = \{l_0, l_1, ..., l_{\frac{N}{2}-1}\}$, *e.g.*, A-A'-B-C-D-D, where each l_i corresponds to motion segment s_i. Distinct motions receive different labels, identical motions share the same one, and mirrored motions are indicated by prime-labeled counterparts (*e.g.*, A and A').

Motion segments quantization. We formulate motion segment sequence labeling as a quantization problem: clustering similar motion segments and assigning each a cluster ID as its label. Each segment s_i of length T_i is represented by the SMPL-X [93] pose sequence recovered from the video: $s_i = \{(\theta_{t_i} \in \mathbb{R}^{3\times(J+1)}, \tau_{t_i} \in \mathbb{R}^3)\}_{t_i=0}^{T_i}$, where θ_{t_i} contains per-joint axis-angle rotation for $J = 21$ body joints in addition to a joint for global rotation (the 0-th joint), and τ_{t_i} denotes the global translation in 3D space.

For clustering, we focus solely on poses—ignoring global translations—to capture distinctive motion patterns. The distance between two SMPL-X poses is defined as the average geodesic distance across joints:

$$d_\theta(\theta_1, \theta_2) = \frac{1}{J}\sum_{j=0}^{J} ||\log(R(\theta_1^j)^T R(\theta_2^j))||_F \tag{7.1}$$

where $R(\theta^j)$ converts the axis-angle representation of joint j into a rotation matrix. To account for slight temporal offsets between beats, we compute the distance between two motion segments using dynamic time warping (DTW), with d_θ as the local cost metric between poses. The clustering function $\mathcal{C}$ is then defined as:

$$\mathcal{C}(\mathcal{S}; \text{DTW}_{d_\theta}, \epsilon_\theta) \rightarrow \{\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_C\} \tag{7.2}$$

where $\bigcup_{c=1}^{C} \mathcal{C}_c = \mathcal{S}$, with $\mathcal{C}_i \bigcap \mathcal{C}_j = \emptyset$, for $i \neq j$. Segments within the same cluster satisfy: $\text{DTW}_{d_\theta}(s_i, s_j) < \epsilon_\theta,\ \forall s_i, s_j \in \mathcal{C}_c$.

Mirrored motion segments detection. After the quantization stage, we identify mirrored motion segments in two steps.

Mirrored pose clusters. A mirrored joint rotation is defined by reflecting the axis-angle vector across the sagittal (YZ) plane: $\mathcal{F}(\theta^j) = (\omega_x, -\omega_y, -\omega_z)$, where $\theta^j = (\omega_x, \omega_y, \omega_z)$. Then

a mirrored pose θ' is obtained by applying this reflection to each joint after left-right joint swapping:

$$\theta'^{j} = \mathcal{F}(\theta^{\pi(j)}) \quad \text{for } j = 0, \ldots, J \tag{7.3}$$

here $\pi(j)$ denotes the left-right joint permutation (*e.g.* swapping left/right arms, legs, and shoulders), and for central joints (*e.g.*, spine, neck, head), $\pi(j) = j$, so only the reflection is applied.

Two clusters $\mathcal{C}_a$ and $\mathcal{C}_b$ are considered mirrored if there exists at least one pair of segments $s_i \in \mathcal{C}_a, s_j \in \mathcal{C}_b$, such that the mirrored version of s_i, denoted by $s'_i = \{\theta'_{t_i}\}_{t_i=0}^{T_i}$, is similar to s_j under dynamic time warping: $\text{DTW}_{d_\theta}(s'_i, s_j) < \epsilon_{\theta'}$. The resulting set of mirrored cluster pairs is: $\mathcal{M} = \{(\mathcal{C}_a, \mathcal{C}_b) |\ \exists s_i \in \mathcal{C}_a, s_j \in \mathcal{C}_b, \text{DTW}_{d_\theta}(s'_i, s_j) < \epsilon_{\theta'}\}$.

Mirrored motion directions within a cluster. For clusters without a mirrored counterpart, we further check whether they can be internally partitioned into two directionally mirrored groups. We first extract the overall motion direction $\vec{d}_{s_i}$ of each motion segment s_i using its global translation: $\vec{d}_{s_i} = (\tau_{t_i}^{T_i} - \tau_{t_i}^{0})/||\tau_{t_i}^{T_i} - \tau_{t_i}^{0}||$, where $\tau_{t_i}^{0}$and $\tau_{t_i}^{T_i}$ denote the segment's start and end positions, respectively.

To identify mirrored directions, each motion direction $\vec{d}_{s_i}$ is reflected across the YZ plane as $\vec{d}'_{s_i} = \text{diag}([-1, 1, 1])\vec{d}_{s_i}$. We then perform bipartite matching to find mirror pairs (s_i, s_j) that satisfy $\|\vec{d}'_{s_i} - \vec{d}_{s_j}\| < \epsilon_d$. If valid pairs are found, we assign all matched segments into two directionally consistent groups based on their directional similarity. The original cluster $\mathcal{C}_i$ is then partitioned into two mirrored subgroups $(\mathcal{C}_i^0, \mathcal{C}_i^1)$, which are then added to the mirrored cluster set $\mathcal{M}$.

Finally, we assign a unique label to each cluster. For each mirrored pair $(\mathcal{C}_a, \mathcal{C}_b) \in \mathcal{M}$, we assign a base label l_a (*e.g.* A) to segments in $\mathcal{C}_a$, and its mirrored label l'_a (*e.g.* A') to segments in $\mathcal{C}_b$. Clusters without a mirrored counterpart are assigned a distinct label without a prime.

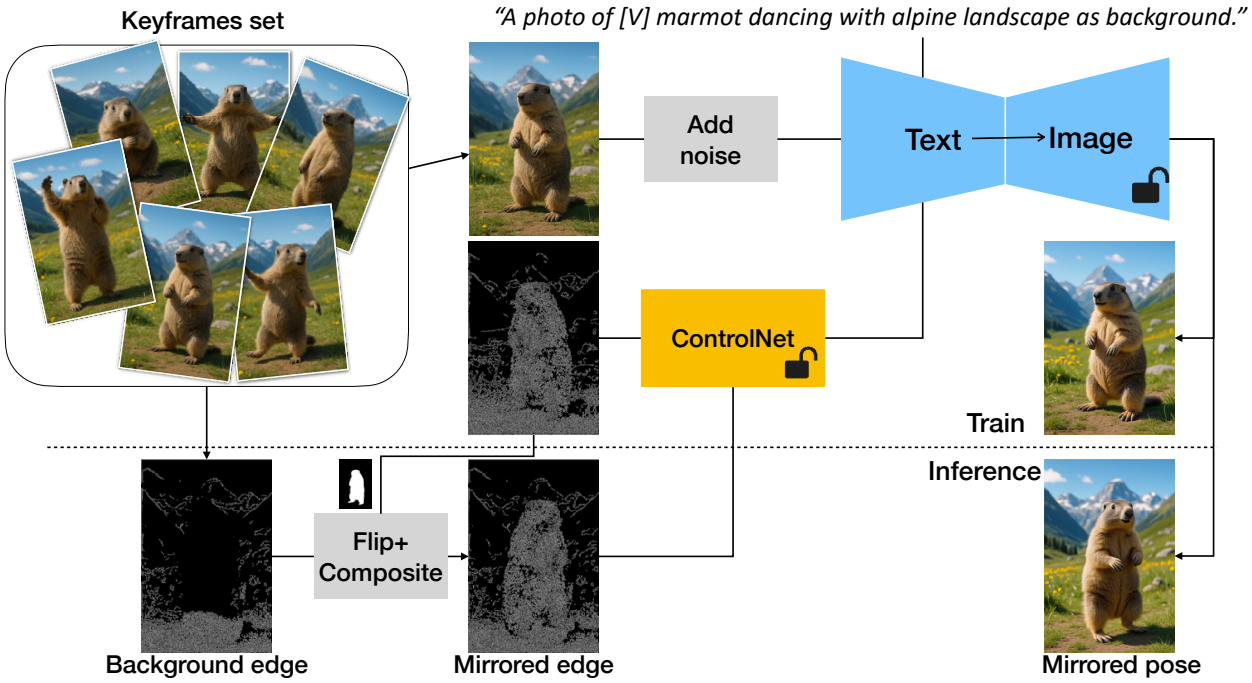


Figure 7.2: **Mirrored pose generation.** We fine-tune a text-to-image model with ControlNet using the canny edges extracted from each keyframe as conditioning. During inference, mirrored pose images are generated by flipping only the subject edges and using an inpainted background edge composed from the keyframe set.

7.2.2 Mirrored Pose Image Generation

To augment the keyframe set with mirrored counterparts, we generate visually consistent keyframe pairs for each input pose. This process (Fig. 7.2) involves fine-tuning a text-to-image model with ControlNet, generating mirrored edge maps, and re-generating the original keyframes for consistency. In the end, we get a complete set of consistent keyframes $\mathcal{I} = \{I_0, \ldots, I_{K-1}, I'_0, \ldots, I'_{K-1}\}$, where I'_k is the mirrored version of I_k.

Fine-tuning. We fine-tune a pretrained text-to-image model on the input keyframes set, overfitting it to capture the appearance of the specific input subject instance and the background. To provide structural guidance, we incorporate ControlNet [156] using the canny

edge maps extracted from the input images as a conditional input. We use the prompt format: "A photo of [V] [*subject class name*] dancing [*background description*].", where [V] is a unique token for identifying the subject **instance** rather than **class**. The placeholders [*subject class name*] and [*background description*] are replaced with the actual class name of the subject and background description. For example, "A photo of [V] marmot dancing with alpine landscape as background."

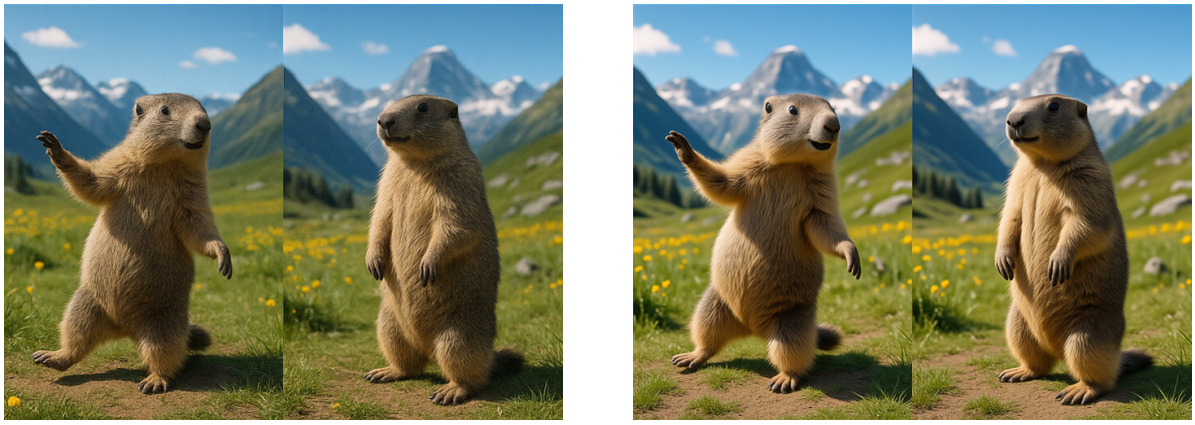
Mirrored edge generation. To generate mirrored pose images, We first extract the subject mask using SAM [68]. We then construct a unified background canny edge map by inpainting and stitching the background edges from all input keyframes. For each keyframe, we extract the subject's edge map and horizontally flip it to create a mirrored subject edge map. This flipped edge map is then composited with the shared background edge map to generate a full mirrored edge map. which is used as input to the fine-tuned model to generate the corresponding mirrored image.

Keyframes re-generation. The original keyframes may contain slight inconsistencies in the background due to generation instability. Additionally, the fine-tuned model may introduce subtle color shifts during inference. To ensure visual consistency among the augmented keyframes set, we regenerate the original keyframes using the same model and shared background edge map (see Fig. 7.3 for an example).

Implementation details. We use FLUX.1-dev and Xlabs-AI/flux-controlnet-canny as the pretrained text-to-image and controlnet model. We fine-tune them jointly with a LoRA rank of 16 for 500 epochs, Training on 6 keyframes takes around 90 minutes on a single A100 GPU. Canny edges are extracted using threshold values of (50, 100).

7.2.3 Choreography Pattern Driven Dance Synthesis

Given the augmented keyframe set $\mathcal{I} = \{I_0, \ldots, I_{K-1}, I'_0, \ldots, I'_{K-1}\}$, where I'_k denotes the mirrored counterpart of keyframe I_k, and the choreography pattern label sequence $\mathcal{L} = \{l_0, l_1, ..., l_{\frac{N}{2}-1}\}$, the goal is to find an optimal walk path $\mathcal{P} = \{I_{p_0}, I_{p_1}, \ldots, I_{p_{N-1}}\}$ through



(a) raw keyframes (b) refined keyframes

Figure 7.3: Improving visual consistency by re-generating keyframes with shared background edges.

the keyframe set, and the i-th keyframe I_{p_i} in the path corresponds to the i-th beat. We then apply video diffusion model to generate in-between frames, finally producing the final dance video.

Since each label l_i corresponds to a motion segment between keyframe pairs $(I_{p_{2i}}, I_{p_{2i+1}})$, we cast path planning as a graph optimization, where each node represents a candidate keyframe pair. The choreography label sequence $\mathcal{L}$ specifies assignment constraints: same labels map to the same pair, distinct labels to distinct pairs, and mirrored labels to mirrored pairs. The object is to assign each label l_i to a node such that these constraints are met while minimizing the total transition cost along the path.

Keyframe graph construction. We construct the keyframe graph $G = (V, E)$ as a directed graph, where each node $(I_u, I_v) \in V$, with $I_u \neq I_v$, represents an ordered pair of keyframes from the augmented keyframe set $\mathcal{I}$. Note $(I_u, I_v) \neq (I_v, I_u)$. Each node corresponds to a potential dance segment from I_u to I_v, and each edge $(I_u, I_v) \to (I_w, I_x) \in E$, with $I_v \neq I_w$, represents a valid transition between segments.

To ensure both expressive motion and synthesis feasibility, we filter nodes based on the average flow magnitude $|F(I_u, I_v)|$ from I_u to I_v, computed over the foreground region of the subject. We use RAFT [125] to compute the optical flow. Nodes with flow that is too small or too large are discarded. Only node pairs with acceptable motion range are retained:

$$V = \{(I_u, I_v) \mid M_{\text{low}} < |F(I_u, I_v)| < M_{\text{high}}\} \tag{7.4}$$

To make the synthesized dance smooth and fluid, we define the edge cost between two nodes as the flow magnitude between the end keyframe of the first node and start one of the next node: $\mathcal{T}((I_u, I_v) \to (I_w, I_x)) = |F(I_v, I_w)|$. We prune high-cost transitions by including only edges with flow below a threshold:

$$E = \{((I_u, I_v) \to (I_w, I_x)) \mid |F(I_v, I_w)| < M_{\text{high}}\} \tag{7.5}$$

We also define a mirroring function $\mu : V \to V$ over graph node such that two nodes are mirrored if and only their respective keyframes are mirrored: $\mu((I_u, I_v)) = (I'_u, I'_v)$.

Graph optimization. We define a node assignment function:$\phi : \mathcal{L} \to V$, which maps each choreography label $l_i \in \mathcal{L}$ to a graph node $(I_u, I_v) \in V$, forming a walk path through the keyframe graph. The goal is to find the optimal assignment ϕ^* that minimizes the total transition cost across the sequence:

$$\phi^* = \text{argmin} \quad \sum_{i=0}^{\frac{N}{2}-2} \mathcal{T}(\phi(l_i), \phi(l_{i+1})) \tag{7.6}$$

subject to the following constraints:

$$l_i = l_j \;\Leftrightarrow\; \phi(l_i) = \phi(l_j),\; l'_i = l_j \;\Leftrightarrow\; \phi(l_i) = \mu(\phi(l_j)) \tag{7.7}$$

To ensure visual variety and avoid redundancy, we introduce two additional constraints: (1) Different labels cannot map to partially mirrored node pairs—defined as nodes sharing one keyframe (in any order), with the other keyframes mirrored:

$$l_i \neq l_j, \Rightarrow (I_a, I_b) \not\approx (I_c, I_d) \tag{7.8}$$

where $\phi(l_i) = (I_a, I_b), \phi(l_j) = (I_c, I_d)$.

(2) Consecutive, distinct, and non-mirrored labels must not be assigned to nodes that share a keyframe, to prevent unnecessary single keyframe repetition.

$$l_i \neq l_{i+1},\ l'_i \neq l_{i+1} \ \Rightarrow\ (I_a \neq I_c \wedge I_b \neq I_d) \tag{7.9}$$

where $\phi(l_i) = (I_a, I_b), \phi(l_{i+1}) = (I_c, I_d)$.

7.2.4 Warp to Music

We generate the final dance video by applying a video diffusion model to synthesize in-between frames along the optimized keyframe walk path $\mathcal{P} = \{I_{p_0}, I_{p_1}, \ldots, I_{p_{N-1}}\}$, where each keyframe I_{p_i} corresponds to beat position i. Note that since there are motion repetition in the choreography, we only have to synthesize videos between unique keyframe pairs. In practice, we use Framer [134], which generates 14 in-between frames, and we assume a fps of 25. To synchronize with the music, we warp the video timeline such that the timing of every keyframe in $\mathcal{P}$ align with the corresponding beat time in the audio. Following the visual rhythm strategy from [17], we accelerate the warping rate into beat points and decelerate before and after to preserve beat saliency while ensuring temporal smoothness.

7.3 Experiments

We generate a collection of input keyframe grids featuring approximately 25 animal instances, some captured as half-body views. The animal classes include *marmot, capybara, hedgehog, meerkat, penguin, sea otter, cat, quokka, beaver* and others. We also incorporate characters such as *Elmo*. For the video results, we generate dance videos for these instances using five popular song clips, ranging from 16 to 28 seconds in length, with choreography patterns extracted from the corresponding YouTube video clips. While we showcase our method using these examples, it can adapt to any choreography patterns paired with music. Figs. 7.6, 1.5 shows selected examples of the final keyframe pairs assigned for each choreography label, arranged in the order specified by the choreography pattern.

7.3.1 Keyframes Generation

We generate initial keyframes set by prompting text-to-image model FLUX to generate an image grid with consistent keyframes using prompt template like "a 3x2 grid of frames, showing 6 consecutive frames from a video clip of [...]". For example, the description prompt might be *"a marmot dancing wildly in the wild alpine landscape, striking a variety of fun and energetic poses"*. The same prompt format can also be used with GPT-4o, which supports even finer specifications. For instance, one can specify: *"Generate a grid with 2 rows and 3 columns. Depict a quokka with distinct poses with a wild background. The postures should feel natural for the quokka's anatomy...."*. In all of our results, we use a total of 6 input keyframes.

7.3.2 Choreography Pattern Labeling

We collect a total of 20 dance video clips featuring various 4/4 music tracks from Youtube and TikTok, ranging from 12s to 28s in length. To create ground truth, we manually annotate the choreography pattern label sequence. We then evaluate our method in Section 7.2.1 by comparing our extracted label sequences against the ground truth ones. Beat times are detected using Librosa, and SMPL-X pose sequences are recovered from the videos using GVHMR [109]. We set the threshold values as follows: $\epsilon_\theta = 0.21, \epsilon_{\theta'} = 0.25$ and $\epsilon_d = 0.1$. Specifically, we evaluate two aspects: (1) Clustering accuracy, where each unique label—including mirrored variants—is treated as a distinct cluster. We assess the clustering results using standard metrics: Adjusted Rand Index (ARI) and Normalized Mutual Information (NMI); (2) Mirror detection accuracy, where we compute precision and recall based on the correctly identified mirrored motion segments pairs.

We report the results in Table 7.1. Our training-free extraction method achieves overall strong quantization accuracy, effectively differentiating different motion patterns. For mirroring, our prediction occasionally misses mirrored pairs, typically in cases where the poses are symmetrical but exhibit subtle mirroring in head or body orientation. Since our method

Clustering		Mirroring		
ARI↑	NMI↑	Prec. ↑	Recall↑	F1↑
0.94	0.98	0.93	0.91	0.92

Table 7.1: Evaluation on choreography pattern labeling.

also can output representative motions for each choreography label, users can more easily visualize the structure and manually correct annotation errors if needed.

7.3.3 Baseline Comparisons

Baselines. A straightforward baseline is music-conditioned video generation, which generates videos directly from audio and text prompts. However, methods such as MusicInfuser [45] are trained on human dance videos and typically produce only short clips, *e.g.*, 5s. So they cannot generalize to long animal dance videos synchronized to the input music. Several examples are provided in the supplementary videos.

Next we compare our method to human pose driven single image animation method using Animate-X [122], which animates an input image according to a sequence of human skeleton poses and works with anthromorphic characters. For each of our generated dance videos, we extract the pose sequence from the same reference video used to extract the choreography pattern and use it as the driving sequence for Animate-X.

User study. Since there are no existing long, structured animal dance videos for direct reference, we conducted a perceptual user study to evaluate the generated dance videos. We used 40 generated dance video pairs across 5 different songs, and invited 31 participants. Each participant was presented a random set of 8 pairwise comparisons of our results and Animate-X. The order of the videos within each pair was randomized. For each pair, participants were asked to choose which video they judged better on each of four criteria, or select "similar" if they found no difference: (1) *Beat accuracy*—are the dances synchronized with

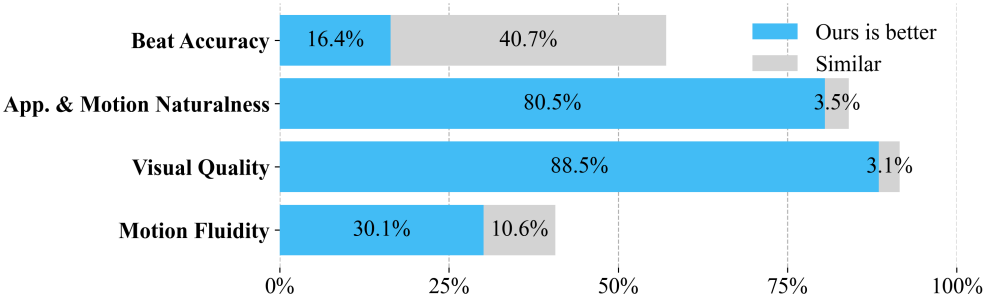


Figure 7.4: User ratings of our approach compared to Animate-X on various criteria.

the music beats? (2) *Appearance & motion naturalness* —do the animals' body proportions and movements feel natural for them? (3) *Visual quality*—do the videos have high overall visual quality (*e.g.* sharpness, clarity, and fewer artifacts)? (4) *Motion fluidity*—are the dance movements smooth and fluid? The responses are shown in Fig. 7.4, and example qualitative comparisons are shown in Fig. 7.5.

Discussions. While beat accuracy was rated similarly for both methods, participants found our results more natural-looking for animals and of higher visual fidelity than those from Animate-X. Specifically, 80.5% of responses rated our videos superior in appearance and motion naturalness, and 88.5% rated them higher in overall visual quality. Animate-X was preferred for motion fluidity (59.3%).

These results are in line with the different setups of these two methods. Animate-X generates animal dances by following fine-grained per-frame human pose sequences, which naturally leads to human-like figures and dance motions—resulting in more fluid and richer movement compared to our method. Yet transferring human poses to animal bodies is inherently difficult: it requires to solve complex correspondence across different body morphologies, and becomes even more challenging when handling the long and diverse pose sequences of real dances. For example, in Fig. 7.5, Animate-X maps the human arms to the penguin's wings, causing the wings to move like human arms; the penguin's differently shaped head further introduces blurry artifacts. Our method instead uses choreography pat-

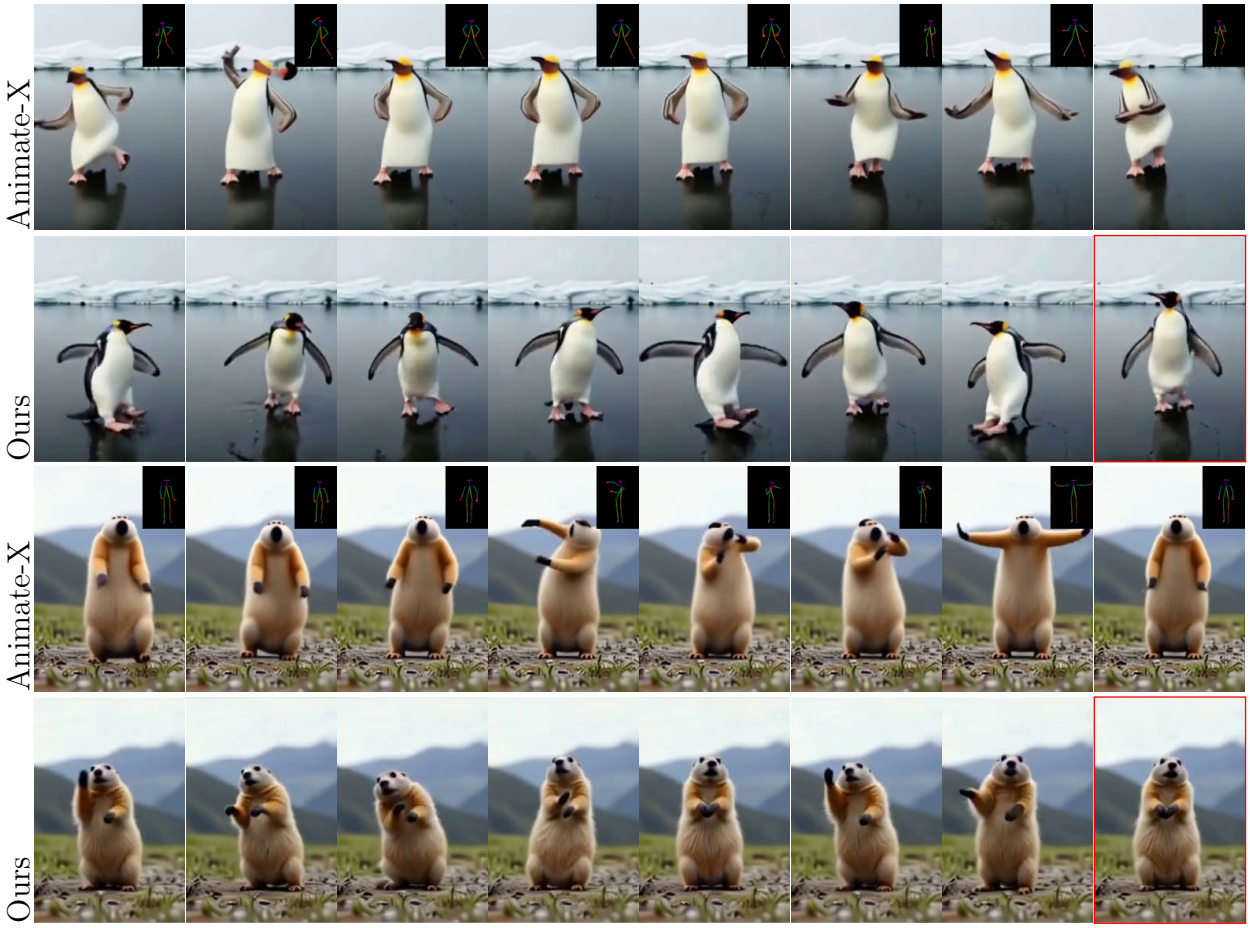
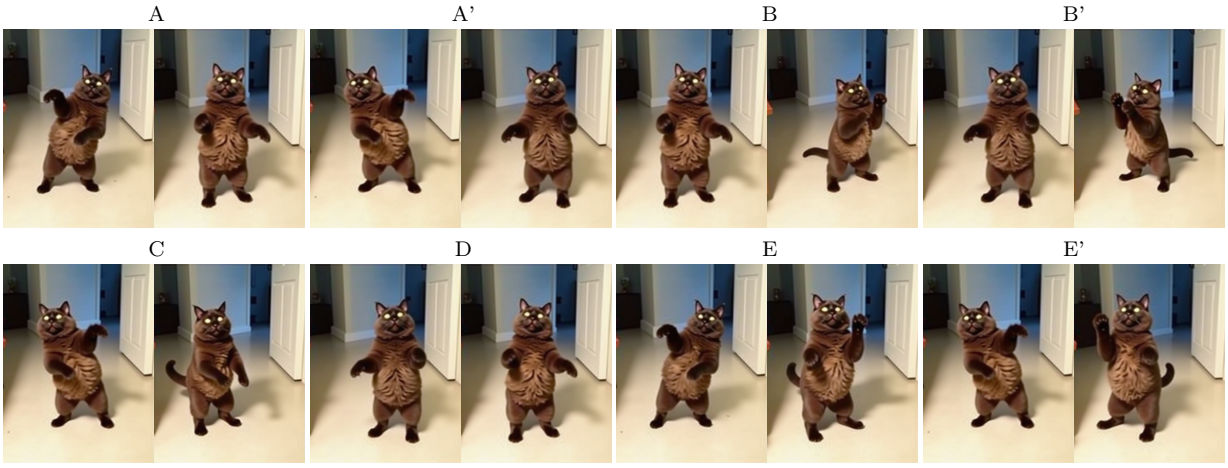
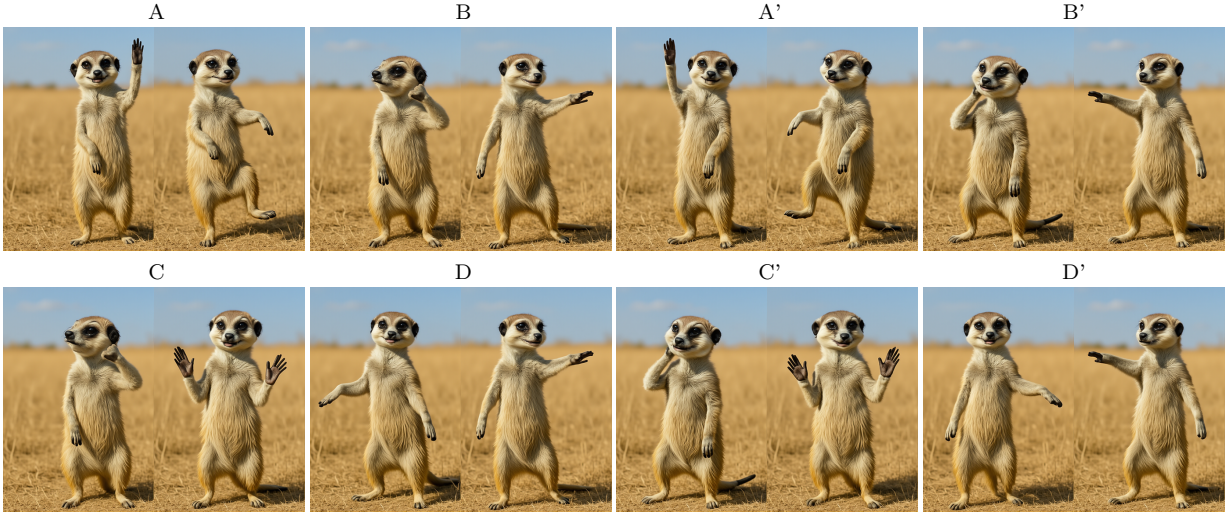


Figure 7.5: Sample frames from both our results (cropped for visualization) and Animate-X. Top: *APT* (Youtube [DJz1zlm73HI]); Bottom: *Cann't Stop Feeling* (Youtube [xyMBnn3dzdU]) used as the reference dance videos. The red box marks the input image for Animate-X.

tern as higher-level control, letting the video diffusion model generate in-between motions so that the final dance follows the intended dance structure rather than fined-grained poses.



(a) Dance to *Uptown Funk* following the choreography pattern extracted from Youtube [U9Zj1BaH01c] (16.0s to 36.0s): A-A'-A-A'-A-A'-A-A'-B-B'-C-D-E-E'-E-E'.



(b) Dance to *Bumblebee* following the choreography pattern extracted from Youtube [GIq7ZgmxE2w] (15.5s to 45.5s): A-B-A'-B'-C-D-C'-D'-A-B-A'-B'-C-D-C'-D'-E-F-G-G-G-G-H-H-H'-H'-G-G-G-G-H-H.

Figure 7.6: Selected examples from our generated dances. Keyframe pairs are labeled by the choreography pattern label, arranged in the order specified by the choreography pattern.

7.3.4 User Control

Given the same choreography pattern for the dance, the user can use pose-grid template to guide the input keyframe poses, control the allowed motion range in the graph, and define custom constraints during graph optimization.

Pose-grid template control. Given a keyframe pose grid as template, we prompt GPT-4o to generate a new grid in which another animal "mimics" each pose from the original, though the poses are not expected to be exact same since different animals have different anatomical structures. This template may come from a previously generated grid (see Fig. 7.7 for an example) or be extracted from a human dance video by identifying distinct poses. This provides a way to guide or customize the input poses, which allows to generate dance videos where different animals dance alike (see supplementary video for examples).

Motion range control. The threshold parameters M_{low} and M_{high} control the range of allowed motion magnitudes between keyframes. A typical setting is $M_{\text{low}} \geq 12.0$ and $M_{\text{high}} \leq 60.0$ at resolution 1024×576. Within this range, lowering M_{low} and increasing M_{high} introduces more candidate nodes and transitions, potentially resulting in richer and more expressive dances.

User custom constraints control. During graph optimization, users can specify hard constraints on node assignments—for instance, enforcing preferred keyframe pair(s) for specific label(s). Additionally, for some dance, mirrored poses happen at the start and end of a choreography label. When such mirrored pose pairs are detected for a specific label, we can enforce corresponding node assignments during optimization. For example, such labels have to be assigned to nodes (I_u, I_v) where $I_v = I'_u$, This allows closer alignment with the reference choreography.

7.3.5 Failure Cases

Motion intensity estimation. Within the keyframe graph, we estimate the underlying motion strength between keyframe pairs using the average flow magnitude. This measure can be unreliable in certain cases. For example, when two keyframes depict mirrored side views, the flow fails to capture the true motion complexity between poses. In Fig. 7.8, the average flow magnitude is 36.81 (image size is 1024×576), which appears moderate due to incorrect correspondences between opposite sides, but the sea otter must rotate from one side view to the other, and this motion is more complex and sometimes challenging for the video model to synthesize. Establishing a reliable link between keyframe flow and the generated motion strength remains an open problem.

Background consistency. As described in Sec. 7.2.2, we re-generate the keyframes with the fine-tuned model to improve visual consistency in the initial keyframe set using a shared background Canny edge map which is inpainted from the original keyframes as control. However, slight background consistency can remain when the original keyframes have shallow depth of field, which limits the Canny edge extraction. For example, in Fig. 7.9, the grass background behind the capybara is blurred in the original keyframe, leaving no detectable edges in that region. As a result, the refined keyframes show slight background inconsistencies in the grass (see bottom row). However, for scenes with clear and sharp backgrounds, our method maintains consistency well.

7.4 Discussion

We present a paradigm for generating music-synchronized, choreography-aware animal dance videos by using choreography pattern as a novel control to impose a structure on the keyframes input. Our work opens up exciting opportunities for creative and fun applications of dancing animals in entertainment and social media. Below we discuss limitations and future works.

Limitations. We use an offline video diffusion model to generate short motion segments between keyframes (*e.g.*, 0.5s for a 120 BPM song). The motion can sometimes look unrealistic: animals may appear to slide or morph between poses rather than moving in a physically plausible way. This stems from the limitations of current video diffusion models in producing naturalistic motion for articulated subjects. However, we are optimistic that these issues can be addressed with continued advances in large-scale video diffusion models,

Future works. To generate more advanced and musically aligned animal dances, two directions can be explored: (1) *dance motion realism*: the motions generated by the video diffusion model may not always reflect plausible or expressive dance motion. Incorporating priors that favor natural, dance-like movement could improve alignment with musical context. (2) *style compatibility*: although our method follows the choreography pattern, it does not consider musical style. Modeling genre-specific movement characteristics could enhance the stylistic coherence of generated dances.

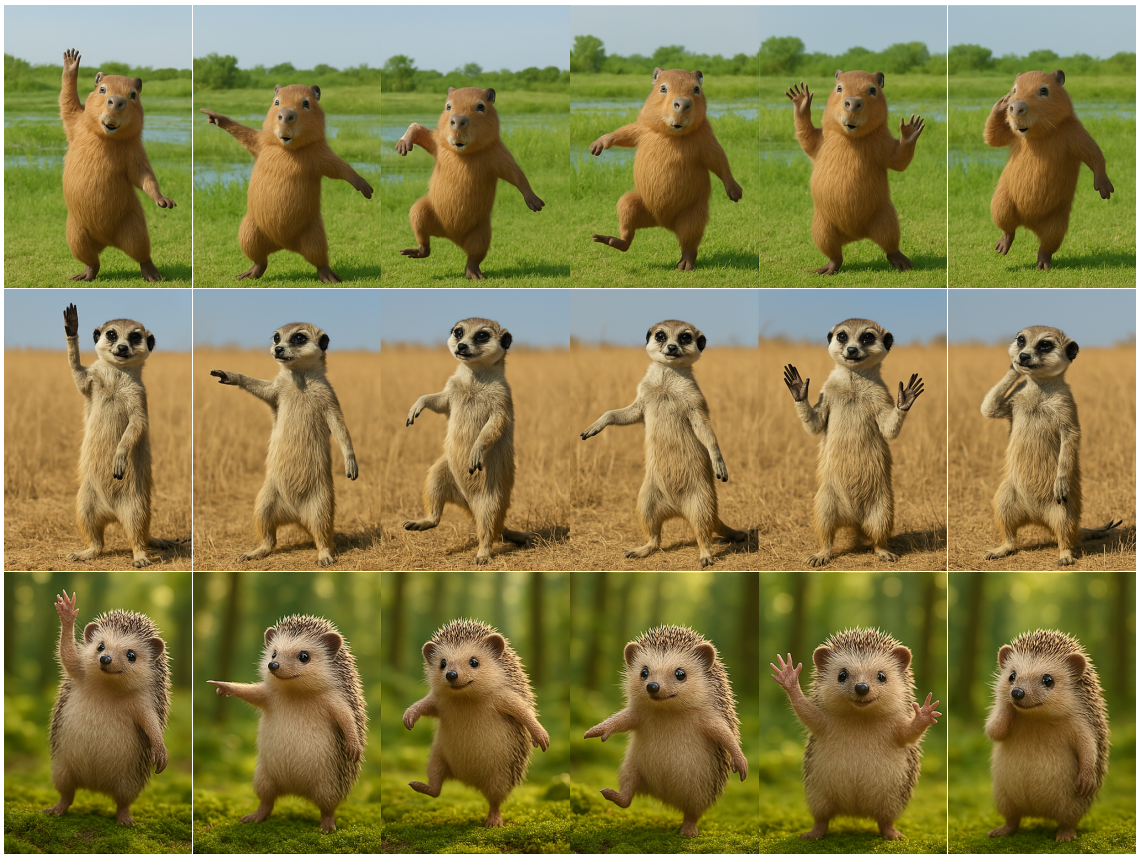


Figure 7.7: *Keyframe pose grid mimicking.* Top row: A keyframe pose grid template showing six distinct poses of a capybara. Middle and bottom rows: A meerkat and a hedgehog mimicking the capybara's poses, while preserving their own body structure, generated using GPT-4o.

Figure 7.8: *Failures.* The average flow magnitude from left to the right is not large, but the underlying motion intensity is much higher.

Figure 7.9: Background inconsistency caused by missing background Canny edges in the original keyframes. Top row: original keyframes generated by GPT-4o. Second row: Canny edge maps used in generating the refined keyframes. Third row: our refined keyframes. Bottom row: zoom-in of the top-left corners (red rectangles) of the refined keyframes, highlighting slight background inconsistencies where Canny edges are absent.

Chapter 8

DISCUSSIONS AND CONCLUSIONS

8.1 Forms of control in generative keyframing

Traditional keyframing relies heavily on the creators to design the characters by specifying control parameters, adjust control points, and tune the interpolation curves for each parameter to achieve motion realism and their creative intent, especially when injecting personality into the created characters. Though generative keyframing extends beyond these parametric controls by automatically generating realistic motions directly from images, it remains essential for the generative model to produce content that aligns with the creator's expressive intent. In other words, control is still necessary for content creation, even in generative systems.

Prompt engineering. Current state-of-the-art image and video generation models are almost always conditioned on text prompts, where user describe the desired content in the generated image or videos. This naturally leads to the practice of prompt engineering, the process of crafting and refining instructions to guide AI models, particularly large language models (LLMs), toward producing the indented outputs. Effective prompt engineering involves a range of techniques, from providing examples (few-shot prompting) to breaking tasks into steps (chain-of-thought prompting [143]) to writing clear and specific instructions. For example, in the animal dance generation work in Chapter 7, creators can provide pose templates to the image generation models such as GPT-4o or Nano Banana[1], and ask the model to describe the poses in the template, helping steer the generation toward characters that follow the desired poses as closely as possible.

[1]https://gemini.google/overview/image-generation/

Motion trajectory control. Beyond text-based prompting, there are also fined-grained visual controls that operate directly on image and videos. Such research has primarily focused on enabling explicit control over elements such as camera motion and subject motion trajectories in the generated videos. For example, *Motion Prompting* [32] trains a video generation model conditioned on spatio-temporally sparse or dense motion trajectories, *i.e.*, point trajectories. *Motion Ctrl* [142] is designed to independently and effectively manage both camera and object motion in the generated videos by taking a desired trajectory as input. *Framer* [134] provides interactive keyframe interpolation that allows users to customize the trajectory of selected keypoints.

Model personalization. Another way to steer a generative model toward a desired visual or motion style is to fine-tune it on a small set of example samples that represent the target style. For example, the pioneering work *Dreambooth* [103] introduced a novel fine-tuning technique that adapts a pretrained text-to-image model using just a few images of a subject, enabling the model to generate that same subject across diverse prompts. Other works [37] further explore how to fine-tune base generative models while preserving their generalization capability, preventing overfitting to the personalized samples.

Timing control. In traditional keyframing, creators control the timing for keyframes based on the desired motion duration and expressive intent: some transitions are intrinsically short or intentionally squashed, while others are longer or stretched, depending on the action and the creator's goals. In contrast, diffusion-based generative models typically generate a fixed number of in-between frames. Regardless of whether the input keyframes are close together or far apart, the model outputs the same fixed-length sequence, which limits the ability to control motion timing and pacing. Enabling adaptive, duration-aware generative inbetweening remains an open challenge and a promising direction for future research.

In summary, each form of control comes with its own advantages and limitations. Text prompting offers high-level semantic control without requiring users to build detailed visual

inputs. Besides, by using semantics, it also solves the correspondence problem which is difficult to do for fine grained visual control when dealing with different modalities. For instance, in the animal pose generation task in Chapter 7, semantic descriptions transfer poses across different animals and even humans, which is much more effectively than purely visual controls, as these subjects have very different anatomies. However, text prompting can be imprecise; for example, models still struggle with attributes like left versus right. Visual controls, on the other hand, provide much more fine-grained guidance but require well-designed, user-friendly interfaces for creators to specify trajectories, poses, or other detailed constraints. Finally, model personalization offers strong control over personalized styles but requires users to curate training examples and access GPU resources to fine-tune the model, making it less accessible in practice. Looking ahead, an important direction for future work is to unify these control modalities into a cohesive and intuitive framework.

8.2 Keyframes for video storytelling

In Chapters 6 and 7, I explored using keyframing for creative applications such as generating deep zooms and long animal dance videos. With the emergence of advanced models like Veo 3.1[2] that can already generate several seconds of high-quality video with synchronized audio, one might ask: what is the role of keyframes in this new landscape?

As in traditional animation, keyframes act as structural anchors that define the essential poses, compositions of an animation. This allows the creator to break the creation process into meaningful sub-tasks: designing keyframes, arranging them along a timeline, and producing transitions between them. More importantly, despite recent progress, current video generation models still struggle with long-range temporal consistency, maintaining identity across shots, and producing coherent multi-shot narratives. They also lack fine control over pacing, scene composition, and high-level story structure.

Therefore, to generate long videos composed of consistent shots, it remains necessary to

[2]https://deepmind.google/models/veo/

generate a coherent set of keyframes to help maintain multi-shot consistency throughout the video. Keyframes provide the global structure and creative intent that today's generative models cannot reliably preserve on their own, especially over extended durations.

8.3 Turning photo album into coherent video

Imagine a situation where we extract random frames at random intervals from a long video, and this video was captured under a long period of time and thus contains diverse subject motions, camera motions, lighting changes and so on. These frames are then randomly shuffled. Could a generative model reconstruct the original video or generate a plausible new video out of these frames?

This thought experiment mirrors a familiar real world situation, *i.e.*, our photo album: after a long trip, we often have hundreds even thousands of photos capturing different moments, places, and events. Together, these images tell a story that could, in principle, be expressed as a video. This raises an intriguing question: can we automatically synthesize a coherent video from such unordered photos, essentially creating a personalized travel movie?

Achieving this would require identifying distinct and meaningful images, organizing them both temporally and semantically, and determining whether smooth transitions between them are possible—given the motion priors of the generative video inbetweening model being used. Chapter 7 provides a focused example of this idea, showing how a small set of generated images can be structurally optimized to form a choreography-aware dance video.

8.4 The inverse process: identifying keyframes from a video

So far, this thesis has examined keyframing for the purpose of editing and creation, where the specification of keyframes is largely determined by the creator, even in generative settings, where users can define them through text prompts or other conditioning signals. These keyframes serve as structural anchors for controlling how a video unfolds.

An intriguing inverse problem is: given an existing video, how can we identify its keyframes? In this context, the definition of "keyframe" becomes task-dependent. For video understand-

ing or summarization, keyframes may represent the most informative or semantically rich frames, where advanced vision language models can help analyze video content and extract representative moments. For compression or reconstruction, keyframes take on a different meaning: can we identify the minimum set of frames from which a generative model could reconstruct the full video? Extending this idea further, one could imagine using generative models not only to reconstruct but also to reimagine videos from these extracted keyframes, enabling personalized or creative reinterpretations of existing footage.

8.5 Conclusions

Chapter 4 and 5 explored how generative models can automatically generate in-between transition frames and even realistic motions. These methods offer new possibilities for reimagining a wide range of existing video editing and creation workflows that rely on inbetweening.

Chapter 6 and 7 extends keyframing to a broader context and explores creative generative applications built upon keyframe-based structure. Chapter 6 focuses on deep zoom generation by generating consistent keyframes across multiple spatial scales. Chapter 7 investigates choreography-aware, music-synchronized animal dance video generation, which takes a small set of keyframes generated by an image model as input. Each keyframe represents a distinct animal pose, and a graph-based optimization determines their optimal temporal ordering in the final dance video so as to produce a smooth, coherent dance that follows a specified choreographic pattern.

Together, these contributions advance the frontier of video creation and and open several promising avenues for future work. For generative keyframing to be used at broader scale, developing new forms of control or integrating existing controls remains essential. Another important direction is the consistent generation of keyframes to maintain multi-shot coherence in long-form video generation. Finally, as generative models continue to advance, they will unlock new creative possibilities and inspire applications that may meaningfully transform how people create and interact with video content.

BIBLIOGRAPHY

[1] Stability AI. Stable-diffusion-2-inpainting. *https://huggingface.co/stabilityai/stable-diffusion-2-inpainting*.

[2] Arpit Bansal, Hong-Min Chu, Avi Schwarzschild, Soumyadip Sengupta, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Universal guidance for diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 843–852, 2023.

[3] Wenbo Bao, Wei-Sheng Lai, Chao Ma, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. Depth-aware video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3703–3712, 2019.

[4] Omer Bar-Tal, Hila Chefer, Omer Tov, Charles Herrmann, Roni Paiss, Shiran Zada, Ariel Ephrat, Junhwa Hur, Yuanzhen Li, Tomer Michaeli, et al. Lumiere: A space-time diffusion model for video generation. *arXiv preprint arXiv:2401.12945*, 2024.

[5] Omer Bar-Tal, Lior Yariv, Yaron Lipman, and Tali Dekel. Multidiffusion: Fusing diffusion paths for controlled image generation. *arXiv preprint arXiv:2302.08113*, 2023.

[6] Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, Sumith Kulal, et al. Flux. 1 kontext: Flow matching for in-context image generation and editing in latent space. *arXiv e-prints*, pages arXiv–2506, 2025.

[7] Floraine Berthouzoz, Wilmot Li, and Maneesh Agrawala. Tools for placing cuts and transitions in interview video. *ACM Transactions on Graphics (TOG)*, 31(4):1–8, 2012.

[8] 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.

[9] 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.

[10] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[11] Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18392–18402, 2023.

[12] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[13] Peter J Burt and Edward H Adelson. The laplacian pyramid as a compact image code. In *Readings in computer vision*, pages 671–679. Elsevier, 1987.

[14] Kang Chen, Zhipeng Tan, Jin Lei, Song-Hai Zhang, Yuan-Chen Guo, Weidong Zhang, and Shi-Min Hu. Choreomaster : Choreography-oriented music-driven dance synthesis. *ACM Transactions on Graphics (TOG)*, 40(4), 2021.

[15] Xi Chen, Xiao Wang, Soravit Changpinyo, AJ Piergiovanni, Piotr Padlewski, Daniel Salz, Sebastian Goodman, Adam Grycner, Basil Mustafa, Lucas Beyer, et al. Pali: A jointly-scaled multilingual language-image model. *arXiv preprint arXiv:2209.06794*, 2022.

[16] Duolikun Danier, Fan Zhang, and David Bull. Ldmvfi: Video frame interpolation with latent diffusion models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 1472–1480, 2024.

[17] Abe Davis and Maneesh Agrawala. Visual rhythm and beat. *ACM Transactions on Graphics (TOG)*, 37(4):1–11, 2018.

[18] Haoge Deng, Ting Pan, Haiwen Diao, Zhengxiong Luo, Yufeng Cui, Huchuan Lu, Shiguang Shan, Yonggang Qi, and Xinlong Wang. Autoregressive video generation without vector quantization. *arXiv preprint arXiv:2412.14169*, 2024.

[19] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4690–4699, 2019.

[20] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[21] Tianyu Ding, Luming Liang, Zhihui Zhu, and Ilya Zharkov. Cdfi: Compression-driven network design for frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8001–8011, 2021.

[22] Jiong Dong, Kaoru Ota, and Mianxiong Dong. Video frame interpolation: A comprehensive survey. *ACM Transactions on Multimedia Computing, Communications and Applications*, 19(2s):1–31, 2023.

[23] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[24] Michail Christos Doukas, Stefanos Zafeiriou, and Viktoriia Sharmanska. Headgan: One-shot neural head synthesis and editing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14398–14407, 2021.

[25] Nikita Drobyshev, Jenya Chelishev, Taras Khakhulin, Aleksei Ivakhnenko, Victor Lempitsky, and Egor Zakharov. Megaportraits: One-shot megapixel neural head avatars. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 2663–2671, 2022.

[26] Ariel Ephrat, Inbar Mosseri, Oran Lang, Tali Dekel, Kevin Wilson, Avinatan Hassidim, William T Freeman, and Michael Rubinstein. Looking to listen at the cocktail party: A speaker-independent audio-visual model for speech separation. *arXiv preprint arXiv:1804.03619*, 2018.

[27] Dave Epstein, Allan Jabri, Ben Poole, Alexei A Efros, and Aleksander Holynski. Diffusion self-guidance for controllable image generation. *arXiv preprint arXiv:2306.00986*, 2023.

[28] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

[29] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883, 2021.

[30] Haiwen Feng, Zheng Ding, Zhihao Xia, Simon Niklaus, Victoria Abrevaya, Michael J Black, and Xuaner Zhang. Explorative inbetweening of time and space. *arXiv preprint arXiv:2403.14611*, 2024.

[31] Songwei Ge, Aniruddha Mahapatra, Gaurav Parmar, Jun-Yan Zhu, and Jia-Bin Huang. On the content bias in fréchet video distance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7277–7288, 2024.

[32] Daniel Geng, Charles Herrmann, Junhwa Hur, Forrester Cole, Serena Zhang, Tobias Pfaff, Tatiana Lopez-Guevara, Carl Doersch, Yusuf Aytar, Michael Rubinstein, Chen Sun, Oliver Wang, Andrew Owens, and Deqing Sun. Motion prompting: Controlling video generation with motion trajectories. *arXiv preprint arXiv:2412.02700*, 2024.

[33] Daniel Geng, Inbum Park, and Andrew Owens. Visual anagrams: Generating multi-view optical illusions with diffusion models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[34] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[35] Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. Densepose: Dense human pose estimation in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7297–7306, 2018.

[36] Yuwei Guo, Ceyuan Yang, Anyi Rao, Maneesh Agrawala, Dahua Lin, and Bo Dai. Sparsectrl: Adding sparse controls to text-to-video diffusion models. *arXiv preprint arXiv:2311.16933*, 2023.

[37] Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*, 2024.

[38] Donald Hearn and M. Pauline Baker. *Computer Graphics with OpenGL*. Pearson Prentice Hall, Upper Saddle River, NJ, 3rd edition, 2004.

[39] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. *arXiv preprint arXiv:2208.01626*, 2022.

[40] 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. *Advances in neural information processing systems*, 30, 2017.

[41] 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.

[42] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[43] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[44] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646, 2022.

[45] Susung Hong, Ira Kemelmacher-Shlizerman, Brian Curless, and Steven M Seitz. Musicinfuser: Making video diffusion listen and dance. *arXiv preprint arXiv:2503.14505*, 2025.

[46] Wenyi Hong, Ming Ding, Wendi Zheng, Xinghan Liu, and Jie Tang. Cogvideo: Large-scale pretraining for text-to-video generation via transformers. *arXiv preprint arXiv:2205.15868*, 2022.

[47] Li Hu. Animate anyone: Consistent and controllable image-to-video synthesis for character animation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8153–8163, 2024.

[48] Li Hu, Guangyuan Wang, Zhen Shen, Xin Gao, Dechao Meng, Lian Zhuo, Peng Zhang, Bang Zhang, and Liefeng Bo. Animate anyone 2: High-fidelity character image animation with environment affordance. *arXiv preprint arXiv:2502.06145*, 2025.

[49] Ruozi Huang, Huang Hu, Wei Wu, Kei Sawada, Mi Zhang, and Daxin Jiang. Dance revolution: Long-term dance generation with music via curriculum learning. In *International conference on learning representations*, 2020.

[50] Zhewei Huang, Tianyuan Zhang, Wen Heng, Boxin Shi, and Shuchang Zhou. Rife: Real-time intermediate flow estimation for video frame interpolation. *arXiv preprint arXiv:2011.06294*, 2020.

[51] Drew A Hudson and Larry Zitnick. Generative adversarial transformers. In *International conference on machine learning*, pages 4487–4499. PMLR, 2021.

[52] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, 2017.

[53] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. Perceiver io: A general architecture for structured inputs & outputs. *arXiv preprint arXiv:2107.14795*, 2021.

[54] Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *International conference on machine learning*, pages 4651–4664. PMLR, 2021.

[55] Siddhant Jain, Daniel Watson, Eric Tabellion, Aleksander Hołyński, Ben Poole, and Janne Kontkanen. Video interpolation with diffusion models. *arXiv preprint arXiv:2404.01203*, 2024.

[56] Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9000–9008, 2018.

[57] Ollie Johnston and Frank Thomas. *The illusion of life: Disney animation*. Disney Editions New York, 1981.

[58] Tarun Kalluri, Deepak Pathak, Manmohan Chandraker, and Du Tran. Flavr: Flow-agnostic video representations for fast frame interpolation. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 2071–2082, 2023.

[59] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35:26565–26577, 2022.

[60] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[61] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119, 2020.

[62] R Joanne Jao Keehn, John R Iversen, Irena Schulz, and Aniruddh D Patel. Spontaneity and diversity of movement to music are not uniquely human. *Current Biology*, 29(13):R621–R622, 2019.

[63] Ira Kemelmacher-Shlizerman, Eli Shechtman, Rahul Garg, and Steven M Seitz. Exploring photobios. *ACM Transactions on Graphics (TOG)*, 30(4):1–10, 2011.

[64] Jae Woo Kim, Hesham Fouad, and James K Hahn. Making them dance. In *AAAI Fall Symposium: Aurally Informed Performance*, volume 2, page 2, 2006.

[65] Tae-hoon Kim, Sang Il Park, and Sung Yong Shin. Rhythmic-motion synthesis based on motion-beat analysis. *ACM Transactions on Graphics (TOG)*, 22(3):392–401, 2003.

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

[67] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018.

[68] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv:2304.02643*, 2023.

[69] Nhat Le, Tuong Do, Khoa Do, Hien Nguyen, Erman Tjiputra, Quang D Tran, and Anh Nguyen. Controllable group choreography using contrastive diffusion. *ACM Transactions on Graphics (TOG)*, 42(6):1–14, 2023.

[70] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 2002.

[71] Hsin-Ying Lee, Xiaodong Yang, Ming-Yu Liu, Ting-Chun Wang, Yu-Ding Lu, Ming-Hsuan Yang, and Jan Kautz. Dancing to music. *Advances in neural information processing systems*, 32, 2019.

[72] Hyeongmin Lee, Taeoh Kim, Tae-young Chung, Daehyun Pak, Yuseok Ban, and Sangyoun Lee. Adacof: Adaptive collaboration of flows for video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5316–5325, 2020.

[73] Juheon Lee, Seohyun Kim, and Kyogu Lee. Listen to dance: Music-driven choreography generation using autoregressive encoder-decoder network. *arXiv preprint arXiv:1811.00818*, 2018.

[74] Yuseung Lee, Kunho Kim, Hyunjin Kim, and Minhyuk Sung. Syncdiffusion: Coherent montage via synchronized joint diffusions. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[75] Ruilong Li, Shan Yang, David A Ross, and Angjoo Kanazawa. Ai choreographer: Music conditioned 3d dance generation with aist++. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 13401–13412, 2021.

[76] Zhengqi Li, Qianqian Wang, Noah Snavely, and Angjoo Kanazawa. Infinitenature-zero: Learning perpetual view generation of natural scenes from single images. In *European Conference on Computer Vision*, pages 515–534. Springer, 2022.

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

[78] Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. *arXiv preprint arXiv:2202.09778*, 2022.

[79] Yihao Liu, Liangbin Xie, Li Siyao, Wenxiu Sun, Yu Qiao, and Chao Dong. Enhanced quadratic video interpolation. In *Computer Vision–ECCV 2020 Workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part IV 16*, pages 41–56. Springer, 2020.

[80] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

[81] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, October 2015.

[82] Grace Luo, Lisa Dunlap, Dong Huk Park, Aleksander Holynski, and Trevor Darrell. Diffusion hyperfeatures: Searching through time and space for semantic correspondence. *arXiv preprint arXiv:2305.14334*, 2023.

[83] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[84] Arun Mallya, Ting-Chun Wang, and Ming-Yu Liu. Implicit warping for animation with image sets. In *NeurIPS*, 2022.

[85] Adriano Manfrè, Ignazio Infantino, Filippo Vella, and Salvatore Gaglio. An automatic system for humanoid dance creation. *Biologically Inspired Cognitive Architectures*, 15:1–9, 2016.

[86] Simon Niklaus and Feng Liu. Softmax splatting for video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5437–5446, 2020.

[87] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive separable convolution. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 261–270, 2017.

[88] Ferda Ofli, Yasemin Demir, Yücel Yemez, Engin Erzin, A Murat Tekalp, Koray Balcı, İdil Kızoğlu, Lale Akarun, Cristian Canton-Ferrer, Joëlle Tilmanne, et al. An audio-driven dancing avatar. *Journal on Multimodal User Interfaces*, 2:93–103, 2008.

[89] OpenAI. Chatgpt [large language model]. *https://chat.openai.com/chat*.

[90] Junheum Park, Keunsoo Ko, Chul Lee, and Chang-Su Kim. Bmbc: Bilateral motion estimation with bilateral cost volume for video interpolation. In *European Conference on Computer Vision*, pages 109–125. Springer, 2020.

[91] Junheum Park, Chul Lee, and Chang-Su Kim. Asymmetric bilateral motion estimation for video frame interpolation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14539–14548, 2021.

[92] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International conference on machine learning*, pages 4055–4064. PMLR, 2018.

[93] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive body capture: 3D hands, face, and body from a single image. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 10975–10985, 2019.

[94] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

[95] Ryan Po, Wang Yifan, Vladislav Golyanik, Kfir Aberman, Jonathan T Barron, Amit H Bermano, Eric Ryan Chan, Tali Dekel, Aleksander Holynski, Angjoo Kanazawa, et al. State of the art on diffusion models for visual computing. *arXiv preprint arXiv:2310.07204*, 2023.

[96] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2017.

[97] Qiaosong Qi, Le Zhuo, Aixi Zhang, Yue Liao, Fei Fang, Si Liu, and Shuicheng Yan. Diffdance: Cascaded human motion diffusion model for dance generation. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 1374–1382, 2023.

[98] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022.

[99] Fitsum Reda, Janne Kontkanen, Eric Tabellion, Deqing Sun, Caroline Pantofaru, and Brian Curless. Film: Frame interpolation for large motion. In *European Conference on Computer Vision*, pages 250–266. Springer, 2022.

[100] Fitsum A Reda, Deqing Sun, Aysegul Dundar, Mohammad Shoeybi, Guilin Liu, Kevin J Shih, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Unsupervised video interpolation using cycle consistency. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 892–900, 2019.

[101] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[102] Ludan Ruan, Yiyang Ma, Huan Yang, Huiguo He, Bei Liu, Jianlong Fu, Nicholas Jing Yuan, Qin Jin, and Baining Guo. Mm-diffusion: Learning multi-modal diffusion models for joint audio and video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10219–10228, 2023.

[103] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22500–22510, 2023.

[104] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Wei Wei, Tingbo Hou, Yael Pritch, Neal Wadhwa, Michael Rubinstein, and Kfir Aberman. Hyperdreambooth: Hypernetworks for fast personalization of text-to-image models. *arXiv preprint arXiv:2307.06949*, 2023.

[105] Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–10, 2022.

[106] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022.

[107] Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4713–4726, 2022.

[108] Sand-AI. Magi-1: Autoregressive video generation at scale, 2025.

[109] Zehong Shen, Huaijin Pi, Yan Xia, Zhi Cen, Sida Peng, Zechen Hu, Hujun Bao, Ruizhen Hu, and Xiaowei Zhou. World-grounded human motion recovery via gravity-view coordinates. In *SIGGRAPH Asia Conference Proceedings*, 2024.

[110] Zhihao Shi, Xiangyu Xu, Xiaohong Liu, Jun Chen, and Ming-Hsuan Yang. Video frame interpolation transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17482–17491, 2022.

[111] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. Animating arbitrary objects via deep motion transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2377–2386, 2019.

[112] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. First order motion model for image animation. *Advances in Neural Information Processing Systems*, 32, 2019.

[113] Aliaksandr Siarohin, Oliver Woodford, Jian Ren, Menglei Chai, and Sergey Tulyakov. Motion representations for articulated animation. In *CVPR*, 2021.

[114] Hyeonjun Sim, Jihyong Oh, and Munchurl Kim. Xvfi: extreme video frame interpolation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 14489–14498, 2021.

[115] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.

[116] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[117] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

[118] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[119] Sora. Sora, video generation models as world simulators. *https://openai.com/index/video-generation-models-as-world-simulators/*.

[120] Guofei Sun, Yongkang Wong, Zhiyong Cheng, Mohan S Kankanhalli, Weidong Geng, and Xiangdong Li. Deepdance: music-to-dance motion choreography with adversarial learning. *IEEE Transactions on Multimedia*, 23:497–509, 2020.

[121] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5693–5703, 2019.

[122] 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.

[123] Luming Tang, Nataniel Ruiz, Chu Qinghao, Yuanzhen Li, Aleksander Holynski, David E Jacobs, Bharath Hariharan, Yael Pritch, Neal Wadhwa, Kfir Aberman, and Michael Rubinstein. Realfill: Reference-driven generation for authentic image completion. *arXiv preprint arXiv:2309.16668*, 2023.

[124] Shitao Tang, Fuyang Zhang, Jiacheng Chen, Peng Wang, and Yasutaka Furukawa. Mvdiffusion: Enabling holistic multi-view image generation with correspondence-aware diffusion. *arXiv preprint arXiv:2307.01097*, 2023.

[125] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pages 402–419. Springer, 2020.

[126] Yoad Tewel, Omri Kaduri, Rinon Gal, Yoni Kasten, Lior Wolf, Gal Chechik, and Yuval Atzmon. Training-free consistent text-to-image generation, 2024.

[127] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021.

[128] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[129] Jonathan Tseng, Rodrigo Castellon, and Karen Liu. Edge: Editable dance generation from music. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 448–458, 2023.

[130] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[131] Vikram Voleti, Alexia Jolicoeur-Martineau, and Chris Pal. Mcvd-masked conditional video diffusion for prediction, generation, and interpolation. *Advances in neural information processing systems*, 35:23371–23385, 2022.

[132] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[133] Ting-Chun Wang, Arun Mallya, and Ming-Yu Liu. One-shot free-view neural talking-head synthesis for video conferencing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10039–10049, 2021.

[134] Wen Wang, Qiuyu Wang, Kecheng Zheng, Hao Ouyang, Zhekai Chen, Biao Gong, Hao Chen, Yujun Shen, and Chunhua Shen. Framer: Interactive frame interpolation. *arXiv preprint arXiv:2410.18978*, 2024.

[135] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 568–578, 2021.

[136] Xiaojuan Wang, Aleksander Holynski, Brian Curless, Ira Kemelmacher, and Steve Seitz. How animals dance (when you're not looking). *arXiv preprint arXiv:2505.23738*, 2025.

[137] Xiaojuan Wang, Janne Kontkanen, Brian Curless, Steve Seitz, Ira Kemelmacher, Ben Mildenhall, Pratul Srinivasan, Dor Verbin, and Aleksander Holynski. Generative powers of ten. *arXiv preprint arXiv:2312.02149*, 2023.

[138] Xiaojuan Wang, Taesung Park, Yang Zhou, Eli Shechtman, and Richard Zhang. Jump cut smoothing for talking heads. *arXiv preprint arXiv:2401.04718*, 2024.

[139] Xiaojuan Wang, Boyang Zhou, Brian Curless, Ira Kemelmacher-Shlizerman, Aleksander Holynski, and Steven M Seitz. Generative inbetweening: Adapting image-to-video models for keyframe interpolation. *arXiv preprint arXiv:2408.15239*, 2024.

[140] Xinlong Wang, Xiaosong Zhang, Zhengxiong Luo, Quan Sun, Yufeng Cui, Jinsheng Wang, Fan Zhang, Yueze Wang, Zhen Li, Qiying Yu, et al. Emu3: Next-token prediction is all you need. *arXiv preprint arXiv:2409.18869*, 2024.

[141] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[142] Zhouxia Wang, Ziyang Yuan, Xintao Wang, Yaowei Li, Tianshui Chen, Menghan Xia, Ping Luo, and Ying Shan. Motionctrl: A unified and flexible motion controller for video generation. 2023.

[143] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[144] Wenming Weng, Ruoyu Feng, Yanhui Wang, Qi Dai, Chunyu Wang, Dacheng Yin, Zhiyuan Zhao, Kai Qiu, Jianmin Bao, Yuhui Yuan, et al. Art-v: Auto-regressive text-to-video generation with diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7395–7405, 2024.

[145] Jay Zhangjie Wu, Yixiao Ge, Xintao Wang, Stan Weixian Lei, Yuchao Gu, Yufei Shi, 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. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7623–7633, 2023.

[146] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

[147] Jinbo Xing, Menghan Xia, Yong Zhang, Haoxin Chen, Xintao Wang, Tien-Tsin Wong, and Ying Shan. Dynamicrafter: Animating open-domain images with video diffusion priors. *arXiv preprint arXiv:2310.12190*, 2023.

[148] Xiangyu Xu, Li Siyao, Wenxiu Sun, Qian Yin, and Ming-Hsuan Yang. Quadratic video interpolation. *Advances in Neural Information Processing Systems*, 32, 2019.

[149] Xing Xu, Tan Wang, Yang Yang, Lin Zuo, Fumin Shen, and Heng Tao Shen. Cross-modal attention with semantic consistence for image–text matching. *IEEE transactions on neural networks and learning systems*, 31(12):5412–5425, 2020.

[150] Kewei Yang, Kang Chen, Daoliang Guo, Song-Hai Zhang, Yuan-Chen Guo, and Weidong Zhang. Face2face ρ: Real-time high-resolution one-shot face reenactment. In *European conference on computer vision*, pages 55–71. Springer, 2022.

[151] 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.

[152] Linwei Ye, Mrigank Rochan, Zhi Liu, and Yang Wang. Cross-modal self-attention network for referring image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10502–10511, 2019.

[153] Egor Zakharov, Aleksei Ivakhnenko, Aliaksandra Shysheya, and Victor Lempitsky. Fast bi-layer neural synthesis of one-shot realistic head avatars. In *European Conference on Computer Vision*, pages 524–540. Springer, 2020.

[154] Yan Zeng, Guoqiang Wei, Jiani Zheng, Jiaxin Zou, Yang Wei, Yuchen Zhang, and Hang Li. Make pixels dance: High-dynamic video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8850–8860, 2024.

[155] Haoxian Zhang, Yang Zhao, and Ronggang Wang. A flexible recurrent residual pyramid network for video frame interpolation. In *European Conference on Computer Vision*, pages 474–491. Springer, 2020.

[156] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023.

[157] Mingao Zhang, Changhong Liu, Yong Chen, Zhenchun Lei, and Mingwen Wang. Music-to-dance generation with multiple conformer. In *Proceedings of the 2022 International Conference on Multimedia Retrieval*, pages 34–38, 2022.

[158] Qinsheng Zhang, Jiaming Song, Xun Huang, Yongxin Chen, and Ming-Yu Liu. Diffcollage: Parallel generation of large content with diffusion models. *arXiv preprint arXiv:2303.17076*, 2023.

[159] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[160] Shengyu Zhao, Jonathan Cui, Yilun Sheng, Yue Dong, Xiao Liang, Eric I Chang, and Yan Xu. Large scale image completion via co-modulated generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2021.

[161] Yang Zhou, Jimei Yang, Dingzeyu Li, Jun Saito, Deepali Aneja, and Evangelos Kalogerakis. Audio-driven neural gesture reenactment with video motion graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3418–3428, 2022.

[162] Ge Zhu, Juan-Pablo Caceres, and Justin Salamon. Filler word detection and classification: A dataset and benchmark. *arXiv preprint arXiv:2203.15135*, 2022.

[163] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.