GENERATIVE INBETWEENING: ADAPTING IMAGE-TO-VIDEO DIFFUSION MODELS FOR KEYFRAME INTERPOLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

We present a method for generating video sequences with coherent motion between a pair of input keyframes. 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 keyframe interpolation, i.e., to produce a video between two input frames. We accomplish this adaptation 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 dual-directional diffusion sampling process that combines the overlapping model estimates starting from each of the two keyframes. Our experiments shows that our method outperforms both existing diffusion-based methods and traditional frame interpolation techniques.

1 INTRODUCTION

Recent advances of large-scale text-to-video and image-to-video models (Blattmann et al., 2023b;a; Wu et al., 2023; Xing et al., 2023; Bar-Tal et al., 2024; Zeng et al., 2024) 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 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 (Pont-Tuset et al., 2017) and Pexels[1], and our method produces notably higher quality videos with more coherent dynamics given distant keyframes.

2 Related Works

Frame interpolation Frame interpolation (Dong et al., 2023) 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 (Jiang et al., 2018; Niklaus & Liu, 2020; Huang et al., 2020; Park et al., 2020; Lee et al., 2020; Park et al., 2021) 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 (Kalluri et al., 2023; Shi et al., 2022) 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 ($\leq 1/30$s) samples in the video. Recent studies have begun to address large motions (Sim et al., 2021; Reda et al., 2022), or quadratic motion (Xu et al., 2019; Liu et al., 2020), 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 (Ho et al., 2020; Dhariwal & Nichol, 2021; Song & Ermon, 2019; Song et al., 2020a;b; Sohl-Dickstein et al., 2015) and videos (Ho et al., 2022b;a; Wu et al., 2023; Blattmann et al., 2023b). Early work MCVD (Voleti et al., 2022) devises a general-purpose diffusion model for a range of video generative modeling tasks including in-between video generation. More recent works (Guo et al., 2023; Jain et al., 2024; Xing et al., 2023) 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 *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 (Bar-Tal et al., 2023; Zhang et al., 2023; Tang et al., 2023; Lee et al., 2023) 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 (Bar-Tal et al., 2023) averages the diffusion model predictions to reconcile the different denoising processes. Recent work, TRF (Feng et al., 2024) 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 can-

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

not 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.

3 BACKGROUND

We introduce some background on Stable Video Diffusion (Blattmann et al., 2023a), 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.

3.1 STABLE VIDEO DIFFUSION

Diffusion models are trained to convert random noise into high-resolution images/videos via an iterative sampling process (Dhariwal & Nichol, 2021; Sohl-Dickstein et al., 2015; Song & Ermon, 2019; Song et al., 2020a;b; Liu et al., 2022). 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 (Karras et al., 2022).

3.2 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 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.

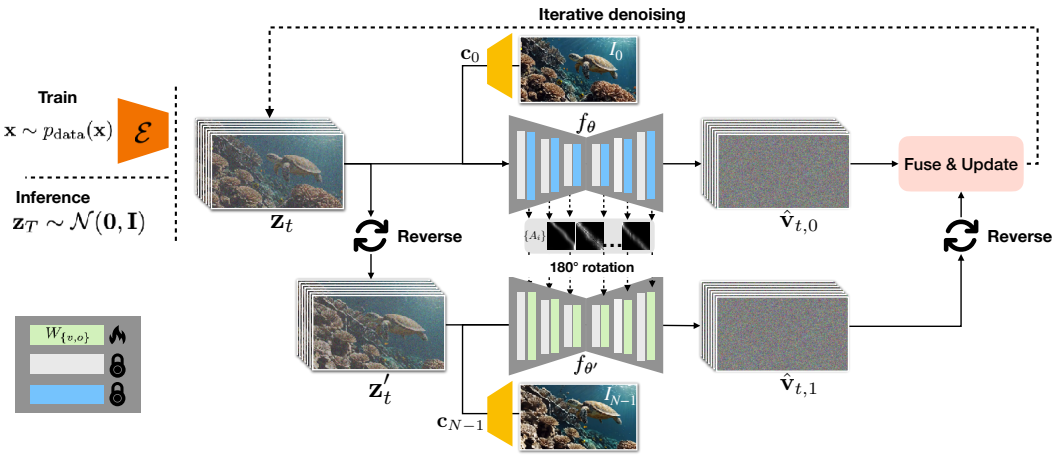

Figure 1: **Method overview.** In the lightweight backward motion fine-tuning stage, an input video $\mathbf{x} = \{I_0, I_1, ..., I_{N-1}\}$ is encoded into the latent space by $\mathcal{E}(\mathbf{x})$, and noise is added to create noisy latent $\mathbf{z}_t$; during inference, $\mathbf{z}_t$ is created 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. 2 for more details.

4 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 4.1, 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. 4.2. 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. 4.3. An overview of our method is shown in Fig. 1.

4.1 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.

4.2 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. 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 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.

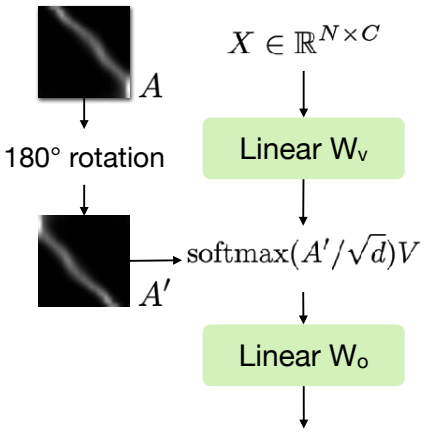

Figure 2: **Temporal self-attention module in the backward motion generation.** Given input tensor X, our attention mechanism additionally takes the respective attention map A from the pre-trained SVD featuring forward motion, rotating it by 180 degrees to create a reverse motion-time association A'. Note that $W_{\{v,o\}}$ are the only trainable parameters in this module.

ALGORITHM 1: Light-weight backward motion fine-tuning

Input: $f_\theta, p_{\text{data}}(\mathbf{x}), \mathcal{E}(\cdot)$
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;
  $t \sim \text{Uniform}(\{1, ..., T\}), \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$;
  $\mathbf{z}_t = \alpha_t \mathbf{z} + \sigma_t \boldsymbol{\epsilon}$;
  $\{A_i\} = \text{extract\_attention\_map}(f_\theta(\mathbf{z}_t; t, \mathbf{c}_0))$;
  $\mathbf{z}'_t = \text{flip}(\mathbf{z}_t)$;
  Compute conditioning $\mathbf{c}_{N-1}$ from I_{N-1};
  Take gradient descent step on $\nabla_{W_{\{v,o\}}} \| f_{\theta'}(\mathbf{z}'_t; t, \mathbf{c}_{N-1}, \{A'_i\}) - \mathbf{y} \|_2^2, \mathbf{y} = \alpha_t \text{flip}(\boldsymbol{\epsilon}) - \sigma_t \mathbf{z}'_t$;
end
Return: $W_{\{v,o\}}$

4.3 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 Figure 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 (Bansal et al., 2023; Feng et al., 2024), 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

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

4.4 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[2] 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 Experiments

In Figs. 3, 4, 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 supplementary to see the results more clearly. Sec. 5.1 describes the data we used to evaluate our method and the baselines. Sec. 5.2 demonstrates how our method outperforms traditional frame interpolation method FILM, and the recent work TRF (Feng et al., 2024) that also leverages SVD for video generation. Sec. 5.3 justifies our design decisions with an ablation study. Sec. 5.4 discusses the optimal scenarios where our method excels and sub-optimal ones where it outperforms the baselines but remains limited by SVD itself. Sec. 5.5 discusses our failure cases.

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

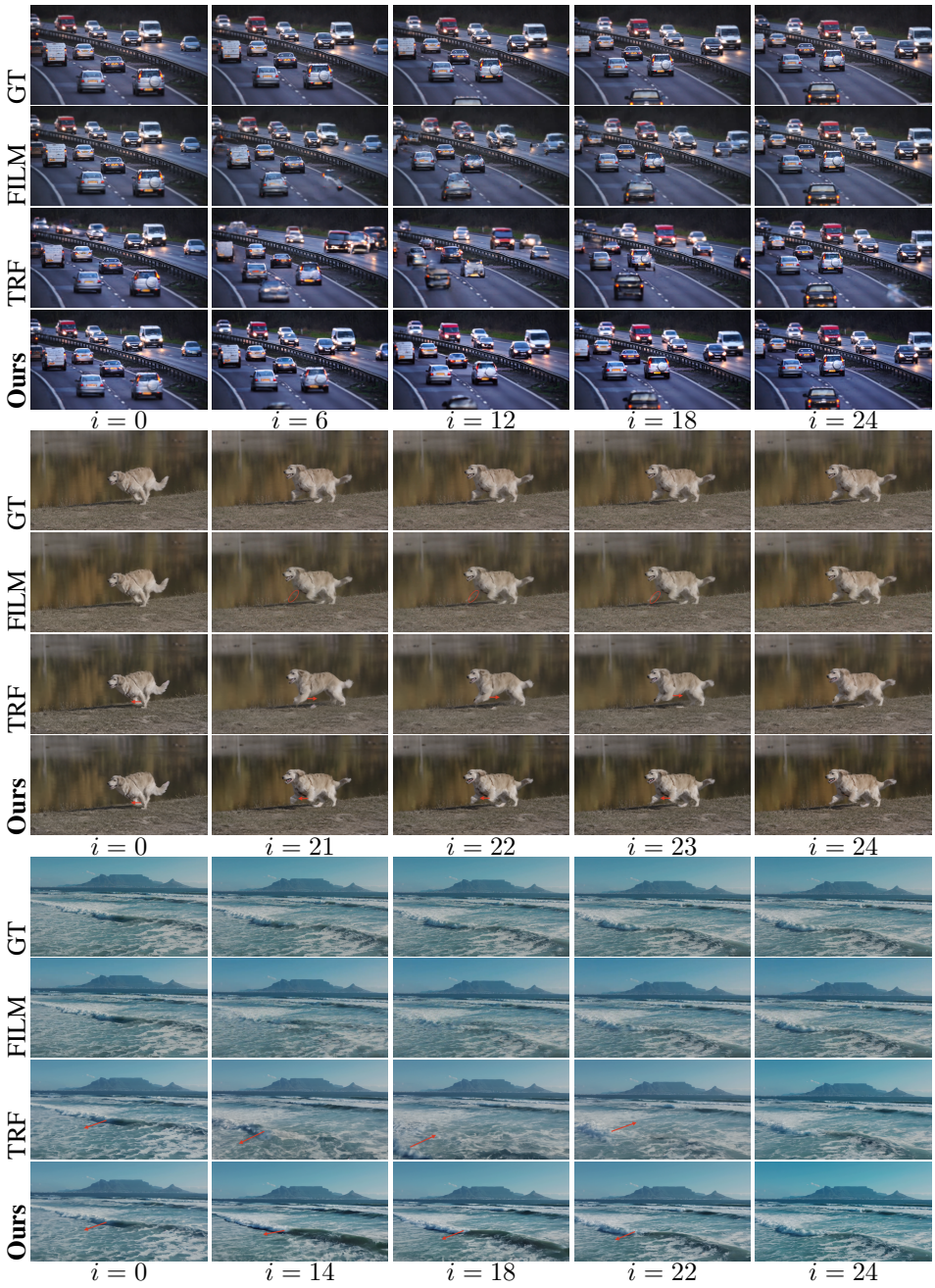

Figure 3: **Qualitative baseline comparisons.** Leftmost ($i = 0$) and rightmost columns ($i = 24$): start and end frames. TRF generates back-and-forth motions, such as vehicles moving forward and then reversing. FILM struggles to find correspondences when the input frames are distant and morphs from the first frame to the last. The red arrow indicates the direction of motion. We recommend viewing the supplementary videos.

5.1 Evaluation Dataset

We use two high-resolution (1080p) datasets for evaluations: (1) The Davis dataset (Pont-Tuset et al., 2017), 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[3], featuring directional dynamic scene motions such as vehicles moving, 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.

5.2 BASELINE COMPARISONS

We mainly compare our approach to FILM (Reda et al., 2022), the current state-of-the-art frame interpolation method for large motion, and TRF (Feng et al., 2024) which also adapts SVD for bounded generation. We show representative qualitative results in Figs. 3, 5. In addition, we also include results for the keyframe interpolation feature from the recent work DynamiCrafter (Xing et al., 2023)—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 (Heusel et al., 2017) and FVD (Ge et al., 2024), 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. 1, and our method outperforms **all** of the baselines by a significant margin.

	Pexels		**Davis**	
	FID ↓	FVD ↓	FID ↓	FVD↓
FILM (Reda et al., 2019)	25.16	371.83	41.85	1048.65
TRF (Feng et al., 2024)	31.43	563.16	36.79	563.07
DynamiCrafter (Xing et al., 2023)	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 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.

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. 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, 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 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. 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.

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

5.3 ABLATIONS

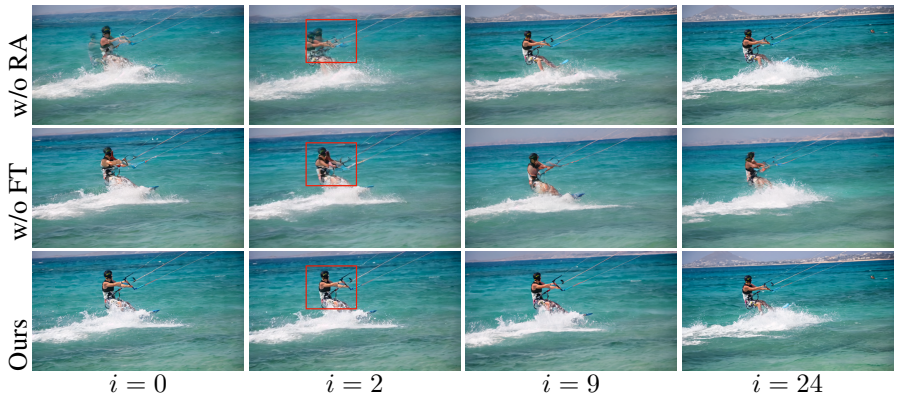

Figure 4: **Ablation study.** We evaluate other options for generating in-between motion consistency. (1) *Ours w/o RA*: full pipeline with fine-tuning all parameters $W_{\{q,k,v,o\}}$ in the temporal attention layers but without using 180-degree rotated temporal self-attention maps as extra input (top 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.

In Fig. 4 and Tab. 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. 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. 4.1, 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. 4 and Tab. 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 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, 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.

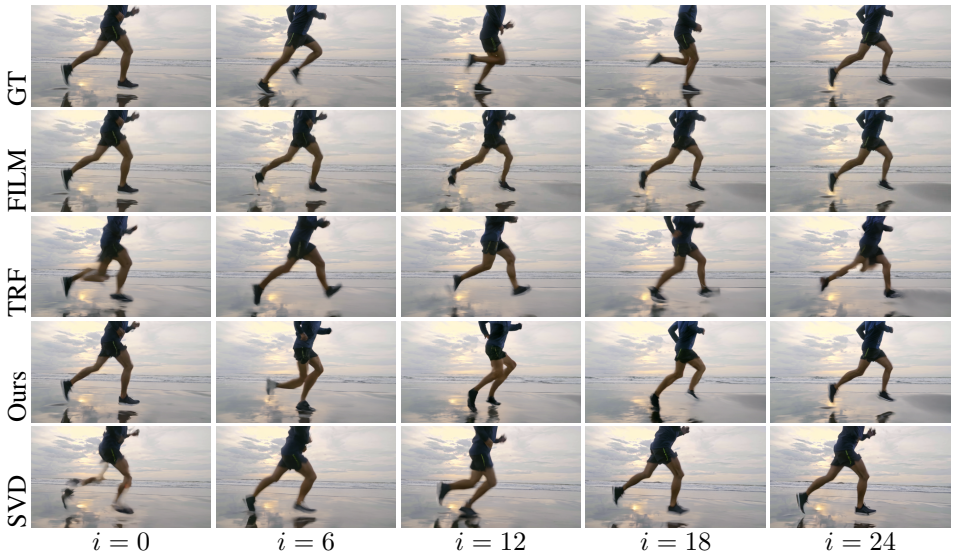

Figure 5: Our method outperforms FILM and TRF in generating articulated movements inbetween, but still struggles to create natural kinematic motions because of the limitation of SVD itself failing to generated complex kinematics (bottom row). Note that the input image serve as conditioning to SVD, so generated first frame might differ from the input image if SVD struggles to create plausible videos from that input.

5.5 Failures

When the input pairs are captured at such distant intervals that they have sparse correspondences, as shown in Fig. 6, 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.

Figure 6: **Failure case.** Our method fails to work well in the cases where input pairs have sparse correspondences.

6 Discussions & Limitations

Our method is limited by the motion quality of the underlying base model, Stable Video Diffusion (SVD), as discussed in Sec. 5.4. 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[4], 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.

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

REFERENCES

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*, pp. 843–852, 2023.

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

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.

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*, 2023a.

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*, pp. 22563–22575, 2023b.

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

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.

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.

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*, pp. 7277–7288, 2024.

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.

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.

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

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*, 2022a.

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, 2022b.

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.

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

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*, pp. 9000–9008, 2018.

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*, pp. 2071–2082, 2023.

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.

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*, pp. 5316–5325, 2020.

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.

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

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*, pp. 41–56. Springer, 2020.

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

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*, pp. 109–125. Springer, 2020.

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*, pp. 14539–14548, 2021.

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.

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*, pp. 250–266. Springer, 2022.

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*, pp. 892–900, 2019.

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*, pp. 17482–17491, 2022.

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

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

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

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

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*, 2020b.

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.

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.

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*, pp. 7623–7633, 2023.

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.

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

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*, pp. 8850–8860, 2024.

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.

A APPENDIX

A.1 SENSITIVITY TO THE SCALE OF TRAINING SET

As stated in Sec. 4.4, 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. 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. 7).

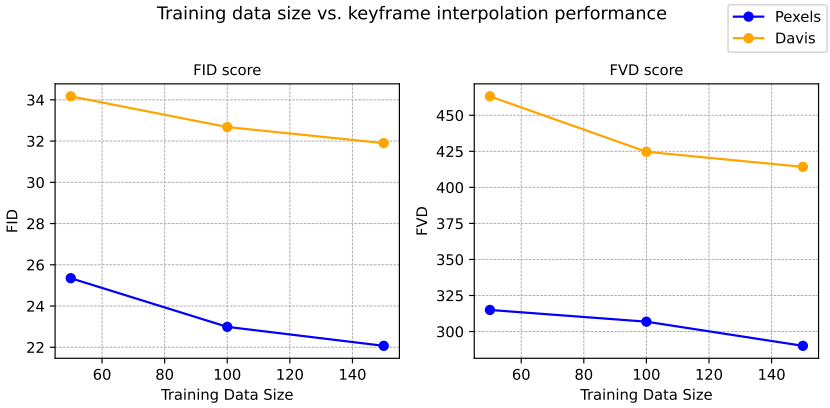

Figure 7: Ablation on how the scales of the training dataset affect our model's performance.

	Pexels		Davis	
	FID ↓	FVD ↓	FID ↓	FVD↓
FILM	25.16	371.83	41.85	1048.65
TRF	31.43	563.16	36.79	563.07
Ours ($motion\ bucket\ id = 255$)	22.18	306.61	34.06	426.53
Ours ($motion\ bucket\ id = 65$)	23.33	270.27	32.75	491.85
Ours ($motion\ bucket\ id = 127$)	22.99	306.84	32.68	424.69

Table 2: Ablation on how the conditioning parameters $motion\ bucket\ id$ in Stable Video Diffusion affect our model's performance. Our method uses 127 as default in the paper.

A.2 THE INFLUENCE OF MOTION BUCKET ID IN STABLE VIDEO DIFFUSION

Stable Video Diffusion[5] (Blattmann et al., 2023a) takes $motion\ bucket\ id$ as micro conditioning parameter in the video generation process, which is expected to affect the motion magnitude the generated video: higher values result in more dynamic video and vice versa. However, keyframe interpolation is a more constrained task where the second end frame provides additional guidance for the generation (Feng et al., 2024). Here we experiment with different $motion\ bucket\ id$ values in Tab. 2, and our method still outperforms TRF and FILM. On the Pexels dataset, $motion\ bucket\ id$ of 65 results in better FVD score (generated motion closer to the ground truth videos), However, the same value results in worse FVD score on the Davis dataset. This discrepancy is likely due to motion difference between the two datasets.

A.3 PSEUDO CODE FOR LIGHTWEIGHT BACKWARD MOTION FINE-TUNING AND DUAL DIRECTIONAL SAMPLING

[5]We use the public available model weights `https://huggingface.co/stabilityai/stable-video-diffusion-img2vid-xt`

```
def get_trainable_params(rev_UNet):
    # Only finetune Wv and Wv in temporal self attention layers
    rev_unet_train_params = []
    for name, param in rev_UNet.named_parameters():
        if 'temporal_transformer_blocks.0.attn1.to_v.weight' in name
        or 'temporal_transformer_blocks.0.attn1.to_out.0.weight' in name:
            rev_unet_train_params.append(param)
            param.requires_grad = True
    return rev_unet_train_params

# Backward motion fine-tuning
rev_UNet = copy.deepcopy(ori_UNet)
ori_UNet.requires_grad(False) # pretrained 3DUNet in SVD
rev_UNet.requires_grad(False) # backward motion UNet to be fine-tuned
optimizer = optim.AdamW(get_trainable_params(rev_UNet)
for epoch in range(0, num_train_epochs):
    rev_UNet.train()
    loss = 0.
    for batch_video in train_dataloader:
        I_0 = batch_video[:, 0] # get the first frame from the video
        I_N = batch_video[:, -1] # get the last frame from the video
        c_0 = compute_image_conditioning(I_0)
        c_N = compute_image_conditioning(I_N)

        batch_video = rearrange(batch_video, "b f c h w -> (b f) c h w")
        z = vae.encode(batch_video)
        z = rearrange(z_0, "(b f) c h w -> b f c h w", f=num_frames)
        z_rev = torch.flip(z, dims=(1,)) # GT reverse latent video

        noise = torch.rand_like(z)
        t = torch.randint(0, num_train_timesteps)
        z_t = noise_scheduler.add_noise(z, noise, t)
        z_t_rev = torch.flip(z_t, dims=(1,))

        pred_noise_1, attention_maps = ori_UNet(z_t, t, c_0)

        # rotate attention maps by 180 degree
        rotated_attention_maps = [torch.flip(attention_map, dims=(-2, -1))
                    for attention_map in attention_maps]

        pred_noise_2 = rev_UNet(z_t_rev, t, c_N, rotated_attention_maps)
        pred_z_rev = noise_scheduler.predict_denoised_sample(pred_noise_2)
        loss += mse_loss(pred_z_rev, z_rev)
    loss.backward()
    optimizer.step()
    optimizer.zero_grad()
return rev_UNet
```

Figure 8: Pytorch pseudocode for lightweight backward motion fine-tuning.

```
# Dual directional diffusion sampling for generating in-between video
# ori_UNet: pretrained 3DUNet in SVD
# rev_UNet: fine-tuned backward motion UNet

c_0 = compute_image_conditioning(I_0)
c_N = compute_image_conditioning(I_N)
z_t = torch.randn(latent_shape) # initialize video latent variable
timesteps = scheduler.set_timesteps(num_steps)
for i, t in enumerate(timesteps):
    # predicted noise in forward motion start from I_0
    pred_noise_1, attention_maps = ori_UNet(z_t, t, c_0)
    if do_classifier_free_guidance:
        pred_noise_uncond_1 = ori_UNet(z_t, t, null_conditioning)
        pred_noise_1 =  pred_noise_uncond_1 +
        guidance_scale * (pred_noise_1 - pred_noise_uncond_1)

    rotated_attention_maps = [torch.flip(attention_map, dims=(-2, -1))
                        for attention_map in attention_maps]

    # predicted noise in backward motion start from I_N
    z_t_rev = torch.flip(z_t, dims=(1,))
    pred_noise_2 = rev_UNet(z_t_rev, t, c_N, rotated_attention_maps)
    if do_classifier_free_guidance:
        pred_noise_uncond_2 = rev_UNet(z_t_rev, t, null_conditioning,
                                       rotated_attention_maps)
        pred_noise_2 = pred_noise_uncond_2
        + guidance_scale * (pred_noise_2 - pred_noise_uncond_2)

    pred_noise_2 = torch.flip(pred_noise_2, dims=(1,))

    pred_noise = (pred_noise_1 + pred_noise)/2. # fuse
    z_t = scheduler.update(z_t, pred_noise, t) # denoise
output_video = vae.decode(z_t)
return output_video
```

Figure 9: Pytorch pseudocode for dual directional diffusion sampling.