DiffSynth: Latent In-Iteration Deflickering for Realistic Video Synthesis

Zhongjie Duan[1], Lizhou You[2], Chengyu Wang[3], Cen Chen[1], Ziheng Wu[3],
Weining Qian[1], Jun Huang[3]

[1]School of Data Science and Engineering, East China Normal University
[2]School of Information, Xiamen University
[3]Alibaba Group
{zjduan,cenchen, wnqian}@stu.ecnu.edu.cn
youlizhou@xmu.edu.cn
{chengyu.wcy, ziheng.wzh, huangjun.hj}@alibaba-inc.com

Abstract

In recent years, diffusion models have emerged as the most powerful approach in image synthesis. However, applying these models directly to video synthesis presents challenges, as it often leads to noticeable flickering contents. Although recently proposed zero-shot methods can alleviate flicker to some extent, we still struggle to generate coherent videos. In this paper, we propose DiffSynth, a novel approach that aims to convert image synthesis pipelines to video synthesis pipelines. DiffSynth consists of two key components: a latent in-iteration deflickering framework and a video deflickering algorithm. The latent in-iteration deflickering framework applies video deflickering to the latent space of diffusion models, effectively preventing flicker accumulation in intermediate steps. Additionally, we propose a video deflickering algorithm, named patch blending algorithm, that remaps objects in different frames and blends them together to enhance video consistency. One of the notable advantages of DiffSynth is its general applicability to various video synthesis tasks, including text-guided video stylization, fashion video synthesis, image-guided video stylization, video restoring, and 3D rendering. In the task of text-guided video stylization, we make it possible to synthesize high-quality videos without cherry-picking. The experimental results demonstrate the effectiveness of DiffSynth. All videos can be viewed on our project page[1]. Source codes will also be released[2].

Introduction

In recent years, diffusion models have achieved remarkable success in the field of image synthesis, surpassing Generative Adversarial Networks (GANs) (Dhariwal and Nichol 2021). In open-source communities, Stable Diffusion (Rombach et al. 2022) has emerged as the most popular diffusion model, and fine-tuned models based on Stable Diffusion have achieved astonishing success in various artistic styles. Furthermore, several research breakthroughs have improved the capabilities of Stable Diffusion (Gal et al. 2022; Zhang and Agrawala 2023; Hu et al. 2021; Ruiz et al. 2023; Meng et al. 2021). The achievements of Stable Diffusion have established it as the mainstream approach of image synthesis.

In our work, we further investigate the capabilities of diffusion models in video synthesis. It is well-known that di-

rectly applying image synthesis to each frame leads to significant flickering (Yu et al. 2021). To address this problem, researchers have proposed solutions from various perspectives. One end-to-end solution is to pre-train a new video synthesis model using video datasets (Blattmann et al. 2023). Yet, there are no large-scale and high-resolution video datasets available to train video diffusion models, and the computational resources required for training are enormous. Furthermore, most existing techniques based on Stable Diffusion cannot be directly applied to newly trained models, making it difficult to control the generated results of such models. Therefore, we focus on one-shot and zero-shot methods, aiming to transfer existing image synthesis models to video synthesis with minimal or even no training. In recent years, several methods for video synthesis have been proposed and gained popularity. For example, Tune-A-Video (Wu et al. 2022) achieves video editing by fine-tuning the textual part of the model. Text2Video-Zero (Khachatryan et al. 2023) restricts the content of adjacent frames using cross-frame attention. However, these methods are difficult to completely eliminate flicker in videos. Our aim is to design a more effective approach to fully eliminate flicker in videos synthesized by diffusion models.

To address the challenges mentioned above, we propose a novel approach named DiffSynth. Specifically, we design a latent in-iteration deflickering framework, with the aim of removing flicker during the intermediate iterations of video synthesis. In our approach, the video-level deflickering is applied to the decoded videos in the latent space. After that, the videos are encoded back from the latent deflickered representations. Hence, we can effectively prevent the accumulation of flicker in denoising steps. For video deflickering, we design a patch blending algorithm to ensure the performance. By remapping different frames to the same frame, we can obtain the appearance features of the same object in different frames. The remapping operator is implemented based on patch matching (Barnes et al. 2009), which can estimate the nearest-neighbor field between two frames. Next, we blend the results to obtain a video with consistent contents. This algorithm can either eliminate high-frequency flicker using a sliding window, or thoroughly remove all flicker by blending all frames. Note that the latter has high computational complexity. Hence, we specifically propose a low time-complexity approximation algorithm, making our

---

[1]https://anonymous456852.github.io/
[2]https://github.com/alibaba/EasyNLP/tree/master/diffusion

approach effective for long video synthesis.

DiffSynth is compatible with most models based on Stable Diffusion. Leveraging the research achievements in image synthesis, we have designed pipelines for multiple downstream tasks, including text-guided video stylization, fashion video synthesis, image-guided video stylization, video restoring, and 3D rendering. These video synthesis pipelines are all transferred from image synthesis pipelines, and we provide hyperparameter templates for these application scenarios to facilitate their usage. To validate the effectiveness of our method, we conducted extensive experiments in two application scenarios. Without any cherry-picking, we are able to generate coherent and realistic videos. Unsurprisingly, DiffSynth comprehensively outperforms existing baseline methods in quantitative metrics and user studies. We summarize the contributions of this paper as follows:

- We propose DiffSynth, a novel approach for coherent and realistic video synthesis.

- We devise a latent in-iteration deflickering framework, which applies video-level deflickering to the latent space of diffusion models, avoiding the accumulation of flicker during the iterative process.

- Based on patch matching, we propose a patch blending algorithm that can significantly eliminate flicker in videos synthesized by diffusion models.

- We design several video synthesis pipelines utilizing DiffSynth for various tasks and demonstrate the superiority of our method in extensive experiments.

Related Work

Diffusion Models

Diffusion models (Song and Ermon 2019; Nichol and Dhariwal 2021; Sohl-Dickstein et al. 2015) are a kind of generative model that decomposes the image synthesis process into a sequential application of denoising models. Unlike GAN-based models (Goodfellow et al. 2020; Huang et al. 2017), diffusion models do not rely on adversarial training and are generally easier to train. Latent Diffusion (Rombach et al. 2022) introduces the concept of converting images from the pixel space to the latent space, making it possible to train the model on limited computational resources. Based on Latent Diffusion, Stable Diffusion becomes the most popular and powerful model in research communities. Recent studies have further enhanced the impressive capabilities of Stable Diffusion in various ways. LoRA (Low-Rank Adaptation) (Hu et al. 2021) reduces computational resource requirements for fine-tuning by incorporating low-rank matrices into the model. To enhance the controllability of images generated by Stable Diffusion, ControlNet (Zhang and Agrawala 2023) uses zero convolution to inject additional conditional information into the model, controlling the model generating objects in specific shapes, characters in specific poses, etc. Textual Inversion (Gal et al. 2022) can synthesize a specific object by adding a new word to the vocabulary and training with a few images. Dreambooth (Ruiz et al. 2023) further improves the model's ability to generate

specific objects. Our proposed method is seamlessly compatible with these approaches, enhancing its capabilities in video synthesis.

Diffusion-based Video Synthesis

Inspired by the remarkable success of diffusion models, researchers endeavored to transfer diffusion models to video synthesis tasks, such as text-to-video synthesis, video style transfer and video editing. For example, Make-A-Video (Singer et al. 2022) and VideoLDM (Blattmann et al. 2023) extend a text-to-image diffusion model to a text-to-video diffusion model by incorporating temporal blocks. Gen-1 (Esser et al. 2023) utilizes a similar temporal architecture to transfer the video style. Besides pre-training a large-scale video model, recent studies focus on synthesizing videos using image models. Tune-A-Video (Wu et al. 2022) only fine-tunes several modules using input video, enabling video editing according to the given prompts. Some zero-shot methods, including FateZero (Qi et al. 2023), Pix2Video (Ceylan, Huang, and Mitra 2023), and Text2Video-Zero (Khachatryan et al. 2023), have explored the possibility of synthesizing videos without additional training. These studies collectively demonstrate the feasibility of applying diffusion models to video synthesis tasks. Motivated by the findings of these studies, we propose a novel zero-shot approach, aiming to convert existing image synthesis pipelines to video synthesis pipelines expediently.

Methodology

In this section, we briefly review the diffusion models and then introduce our proposed approach.

Preliminaries

Generally, diffusion models include many different architectures (Feng et al. 2023; Saharia et al. 2022; Ramesh et al. 2022). In this paper, we focus mainly on Stable Diffusion (Rombach et al. 2022), which is the most popular open-source architecture. A typical Stable Diffusion model contains the following three components:

- **Text Encoder**. A transformer-based language model in CLIP (Radford et al. 2021), converting texts to text embeddings. The text embeddings are subsequently used in classifier-free guidance (Ho and Salimans 2021).

- **U-Net** (Ronneberger, Fischer, and Brox 2015). A vision model ϵ with self-attention (Vaswani et al. 2017), cross-attention, and residual connections (He et al. 2016). This model is trained to denoise images in the latent space.

- **VAE** (Kingma and Welling 2013). A model consists of an encoder $\mathcal{E}$ and a decoder $\mathcal{D}$, where the encoder converts images to latent tensors, and the decoder reconstructs images according to latent tensors.

Both the diffusion process and its reverse process (i.e., the generation process) are conducted in latent space. There are $T + 1$ steps with $\{0, 1, \ldots, T\}$ representing different levels of noise. In the generation process, we start with a noise tensor x_T sampled from a Gaussian distribution, and then

$$\hat{x_0}$$

denoise it stepwise. The iterative formula of each step t is

$$x_{t-1} = \sqrt{\alpha_{t-1}} \left(\frac{x_t - \sqrt{1 - \alpha_t}\, \epsilon(x_t)}{\sqrt{\alpha_t}} \right) + \sqrt{1 - \alpha_{t-1}}\, \epsilon(x_t),$$

(1)

where α_t is the hyperparameter describing how much noise it contains in step t. After the iterative process, the latent tensor x_0 is decoded to an image $X = \mathcal{D}(x_0)$. In addition to text-to-image synthesis, we can modify this process to implement other pipelines. For example, if we add noise to an image and then denoise from an intermediate step, we obtain an image-to-image pipeline for image editing.

To convert an image synthesis pipeline to a video synthesis pipeline, a naive approach is to generate each frame independently. However, most models based on Stable Diffusion are trained on text-image datasets and lack the ability to generate coherent video frames. After each iterative step, the difference between two adjacent frames becomes larger. The difference is accumulated during the generation process and finally results in irreparable flicker and inconsistency.

Latent In-Iteration Deflickering

As we mentioned above, the generation process of diffusion models is conducted in latent space, not pixel space. Inspired by deflickering methods designed for videos (Lei, Xing, and Chen 2020; Lei et al. 2023), we design a framework to apply deflickering to the latent space for better video synthesis.

For a video consisting of n frames, at step t, we have n latent tensors $\{x_t^1, x_t^2, \ldots, x_t^n\}$ corresponding to each frame. If we directly compute $\{x_{t-1}^1, x_{t-1}^2, \ldots, x_{t-1}^n\}$ using formula (1), these latent tensors will become more inconsistent because each tensor is computed independently. To visualize the latent tensors, we first skip to the final step to calculate the estimation of $\{x_0^1, x_0^2, \ldots, x_0^n\}$:

$$\hat{x}_0^i = \frac{x_t^i - \sqrt{1 - \alpha_t}\, \epsilon(x_t^i)}{\sqrt{\alpha_t}}.$$

(2)

Then, we decode $\{\hat{x}_0^1, \hat{x}_0^2, \ldots, \hat{x}_0^n\}$ to images using the decoder component of VAE:

$$\hat{X}^i = \mathcal{D}(\hat{x}_0^i).$$

(3)

Theoretically, $\{\hat{X}^1, \hat{X}^2, \ldots, \hat{X}^n\}$ represent the frames when we denoise $\{x_t^1, x_t^2, \ldots, x_t^n\}$ along with a straight line directed by $\{\epsilon(x_t^1), \epsilon(x_t^2), \ldots, \epsilon(x_t^n)\}$. We employ a video-level deflickering method $\mathcal{F}$ to make the video coherent, i.e.,

$$\{\overline{X}^1, \overline{X}^2, \ldots, \overline{X}^n\} = \mathcal{F}\{\hat{X}^1, \hat{X}^2, \ldots, \hat{X}^n\}.$$

(4)

Next, we encode the processed frames into the latent space:

$$\overline{x}_0^i = \mathcal{E}\left(\overline{X}^i\right).$$

(5)

To synthesize the video frames $\{\overline{X}^1, \overline{X}^2, \ldots, \overline{X}^n\}$, the predicted noise at step t should be:

$$\overline{\epsilon}(x_t^i) = \frac{x_t^i - \sqrt{\alpha_t}\, \overline{x}_0^i}{\sqrt{1 - \alpha_t}}.$$

(6)

Thus, we obtain the reconstructed iterative formula as:

$$x_{t-1}^i = \sqrt{\alpha_{t-1}}\, \overline{x}_0^i + \sqrt{1 - \alpha_{t-1}}\, \overline{\epsilon}(x_t^i).$$

(7)

The pipeline (2-7) makes it possible to apply existing deflickering methods to latent tensors. At each denoising step, we can keep the difference between frames controllable and avoid the accumulation of flicker.

Patch Blending Algorithm

Another problem to be addressed is how to design the video-level deflickering method $\mathcal{F}$ mentioned above. In a video synthesis task without any reference, we can only employ blind video deflickering methods (Lei, Xing, and Chen 2020; Lei et al. 2023), thus it is difficult to keep the video fluent. In many video synthesis tasks (e.g., style transferring and video editing), we have the original video for reference. Hence, we mainly focus on these tasks in this work and propose the following deflickering algorithm.

Assuming that we have obtained the synthesized frames $\{\hat{X}^1, \hat{X}^2, \ldots, \hat{X}^n\}$ in one denoising step, we aim to make the video smoother with reference to the original video $\{X^1, X^2, \ldots, X^n\}$. For an object in frame $\hat{X}^i$, it may also occur in another frame $\hat{X}^j$, thus we intend to remap the corresponding areas in $\hat{X}^j$ to $\hat{X}^i$ and then blend them together. The blended frame will show the consistent information of both $\hat{X}^i$ and $\hat{X}^j$ if the remapping result is accurate. Although recent studies (Kirillov et al. 2023; Han et al. 2022) have achieved impressive success in object segmentation and tracking, it is still difficult to achieve pixel-level accuracy. We have also considered optical flow (Teed and Deng 2020), but we find that optical flow is typically not accurate when the interval of two frames is large. Finally, we decide to utilize a patch matching algorithm (Barnes et al. 2009), which is an effective algorithm to estimate the correspondence between two frames. In this algorithm, the two frames X^i and X^j are divided into some overlapping patches. We first compute a *nearest neighbor field* (NNF) that represents the matched patches and then reconstruct $\hat{X}^i$ using $\hat{X}^j$. Finally, we blend the reconstructed frames with the synthesized frames.

As an efficient implementation, we only remap and blend in a small sliding window. Yet, in the worst case, we need to remap and blend all frames together, which requires $\mathcal{O}(n^2)$ times of NNF estimation. The high time complexity is the main pitfall that prevents this algorithm from application. To improve computational efficiency, we propose an $\mathcal{O}(n \log n)$ approximate algorithm.

For convenience, we use a remapping operator $[j \to i]$ to denote the remapping process and use $\hat{X}^{j \to i}$ to denote the remapped results from $\hat{X}^j$ to $\hat{X}^i$. Note that the remapped and blended results could be remapped to another frame again, but multiple times of remapping and blending may result in non-negligible errors. We use a subscript number to denote the times of remapping and blending, i.e.,

$$[\hat{X}^{j \to i}]_0 = \hat{X}^i,$$

(8)

$$[\hat{X}^{j \to i}]_{u+1} = [\hat{X}^{j \to k}]_u [k \to i].$$

(9)

The blending process is denoted as a blending operator $\oplus$. If we directly use the average, this operator is equivalent to

Frame 1	Frame 2	Frame 3	Frame 4	Frame 5	Frame 6	Frame 7	Frame 8
$\hat{X}^1$	$\hat{X}^2$	$\hat{X}^3$	$\hat{X}^4$	$\hat{X}^5$	$\hat{X}^6$	$\hat{X}^7$	$\hat{X}^8$
	$[\hat{X}^{1\to2}]_1$		$[\hat{X}^{3\to4}]_1$		$[\hat{X}^{5\to6}]_1$		$[\hat{X}^{7\to8}]_1$
			$[\hat{X}^{1\to4}]_2 \oplus [\hat{X}^{2\to4}]_1$				$[\hat{X}^{5\to8}]_2 \oplus [\hat{X}^{6\to8}]_1$
							$[\hat{X}^{1\to8}]_3 \oplus [\hat{X}^{2\to8}]_2 \oplus [\hat{X}^{3\to8}]_2 \oplus [\hat{X}^{4\to8}]_1$

Table 1: An example of a remapping table in the patch blending algorithm.

"+". Note that the coefficient $\frac{1}{2}$ is not included in the blending operator. The final blended result should be divided by the number of frames. We can also train a lightweight network to implement this operator, in order to reduce the errors in the following approximate calculation. We leave this component for future work.

We use a remapping table to store some intermediate variables. An example of a remapping table is presented in Table 1. First, we store each synthesized frame in the first row. Then in the k-th iteration and the $i \cdot 2^k$-th column, we blend the frames in the $(i \cdot 2^k - 2^{k-1})$-th column and then remap it to the $i \cdot 2^k$-th frame. For example, see the 3rd row and the 4th column in Table 1. It is calculated as follows:

$$\left([\hat{X}^{1\to2}]_1 \oplus \hat{X}^2\right)[2 \to 4]$$
$$= \left([\hat{X}^{1\to2}]_1[2 \to 4]\right) \oplus \left(\hat{X}^2[2 \to 4]\right) \quad (10)$$
$$= [\hat{X}^{1\to4}]_2 \oplus [\hat{X}^{2\to4}]_1$$

With another array storing the "$\oplus$" sum of each column, we can calculate the whole table within $\mathcal{O}(n)$ time complexity, and the memory complexity is also $\mathcal{O}(n)$.

Once we obtain the remapping table, we can quickly calculate the smoothed frame $\overline{X}^i = \frac{1}{n} \oplus_{j=1}^{n} \hat{X}^{j\to i}$. For instance, when $i = 6$, we have:

$$\oplus_{j=1}^{6} \hat{X}^{j\to6}$$
$$\approx \left([\hat{X}^{1\to4}]_2 \oplus [\hat{X}^{2\to4}]_1 \oplus [\hat{X}^{3\to4}]_1 \oplus \hat{X}^4\right)[4 \to 6]$$
$$\oplus \left([\hat{X}^{5\to6}]_1 \oplus \hat{X}^6\right) \quad (11)$$
$$= [\hat{X}^{1\to6}]_3 \oplus [\hat{X}^{2\to6}]_2 \oplus [\hat{X}^{3\to6}]_2 \oplus [\hat{X}^{4\to6}]_1$$
$$\oplus [\hat{X}^{5\to6}]_1 \oplus \hat{X}^6$$

Similarly, we reverse the sequence and calculate the remaining part:

$$\oplus_{j=7}^{8} \hat{X}^{j\to6} \approx [\hat{X}^{7\to6}]_1 \oplus [\hat{X}^{8\to6}]_2 \quad (12)$$

As we mentioned above, the "$\oplus$" sum of each column has been stored in another array, thus we can calculate the estimation of $\oplus_{j=1}^{n} \hat{X}^{j\to i}$ within $\mathcal{O}(\log n)$ time.

Taking into account the accuracy of the estimated "$\oplus$" sum, we can easily prove that the maximum number of remapping and blending operations is $\mathcal{O}(\log n)$. In equations (11) and (12), it is apparent that the frames close to the i-th frame exhibit lower error compared to those located further away. Consequently, this algorithm achieves both speed and numerical stability.

Other Modifications

To convert an image synthesis pipeline to a video synthesis pipeline, we further modify the following details.

- **Fixed noise**. When we synthesize images, sampling from the same Gaussian noise leads to the same image if we leave other settings fixed. In video synthesis, the frames in a video are expected to be similar; thus we synthesize each frame from the same Gaussian noise. In some downstream tasks, some information from the input video is supposed to be retrained in the edited video, thus we add the same Gaussian noise to each frame.

- **Cross-frame attention**. If we want to generate an image similar to a reference image, we can concatenate our image with the reference image and synthesize our image in an in-painting pipeline. This is a widely used trick to control the generated content. However, the model will draw unexpected components near the seam line, because it tends to combine the two images into one complete image. Essentially, the information from the reference images is passed to our image mainly by self-attention (Vaswani et al. 2017). Thus, we change self-attention layers to cross-frame attention layers. In the denoising process of the i-th frame, we concatenate x_1, x_{i-1}, x_i and x_{i+1} together in the self-attention layers. If a ControlNet is used in the pipeline, we also convert self-attention layers of ControlNet to cross-frame attention layers.

- **Adaptive resolution**. Today, most videos on the social network are in the shape of rectangles, and their resolution is usually high. However, most diffusion models are trained to synthesize images in the shape of 512×512 squares. Considering the architecture of U-Net, we can easily change the shape and increase the resolution according to the downstream tasks. In our experiments, we surprisingly find that higher resolution sometimes leads to more fluent video frames. The reason is that the implicit patches represent more fine-grained information when the resolution is higher.

- **Deterministic sampling of VAE**. In equation (5) and image-to-image pipelines, the VAE encoder is called to encode images to latent space. In fact, the output of VAE is a Gaussian distribution, not a deterministic tensor. To maintain the consistency of different frames, we use deterministic sampling instead of Gaussian sampling (i.e., setting the standard deviation to zero).

- **Memory-efficient attention**. Cross-frame attention greatly increases the time consumed to synthesize a video, and the memory required is up to $\mathcal{O}(N^2)$, where N is the number of implicit patches. Existing studies

	Text-guided video stylization	Fashion video synthesis
Frame height	512	768
Frame width	960	512
Denoising steps	20	10
ControlNet scale	1.0 (Depth) 1.0 (SoftEdge)	0.3 (Depth) 0.6 (OpenPose)
CFG scale	7.5	7.5
Deflickering window size	∞	7
Deflickering frequency	5	1

Table 2: The hyperparameters in the experiments.

have shown that the attention mechanism could be implemented in low memory (Rabe and Staats 2021). Therefore, we employ flash attention (Dao et al. 2022) to implement the attention mechanism, which is capable of significantly improving computational efficiency and reducing the memory required.

- **Smoothed ControlNet annotator**. When a ControlNet model is used to control the content, an annotator is employed for processing the original frames, but we observed that some annotators may cause flickering control signal. For example, the OpenPose (Cao et al. 2017) annotator can cause tic of the limbs if the frames are not clear enough. To overcome this pitfall, we use Savitzky-Golay smoothing filters (Press and Teukolsky 1990) to smooth the coordinates of the key points detected.

Experiments

To demonstrate the effectiveness of our approach, we conducted the following experiments, including text-guided video stylization and fashion video synthesis. We compared our approach with several state-of-the-art approaches and evaluated the synthesized videos using both quantitive metrics and user study.

Experimental Settings

Text-Guided Video Stylization. In this task, we design a pipeline to transfer the style of videos according to given prompts. The dataset, consisting of 100 high-resolution videos, is collected from a community[3]. Each video is cut into 3 to 5 seconds, including at most 150 frames. We manually write prompts for each video[4]. The pipeline is composed of a popular customized model in open-source communities[5] and two ControlNet models. We use Depth[6] and SoftEdge[7] to provide structure guidance. The information in the original video is only delivered to the pipeline by ControlNet, thus the color is completely ignored. After the final step, we add an additional deflickering step to reduce the flicker generated in the last four steps and set the window size to 121. We further improve the contrast ratio and sharpen the frames slightly to improve video quality.

(a) The input video

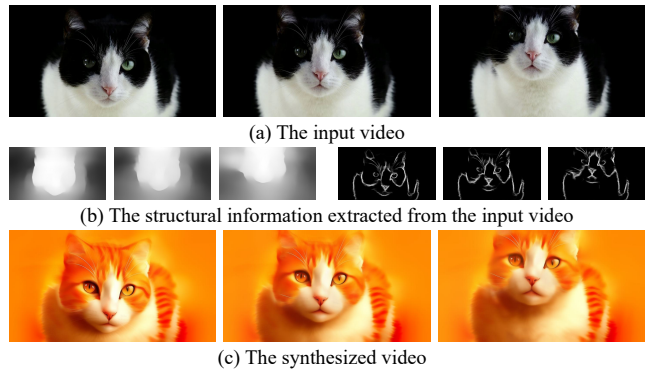(b) The structural information extracted from the input video

(c) The synthesized video

Figure 1: An example of text-guided video stylization. The prompt in this example is "an orange and white cat".

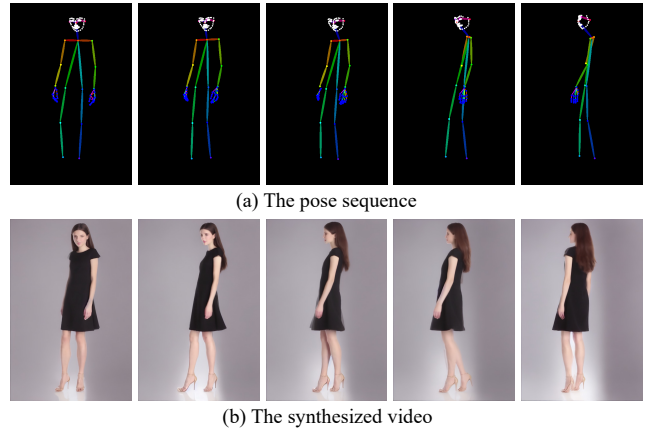(a) The pose sequence

(b) The synthesized video

Figure 2: An example of fashion video synthesis.

Fashion Video Synthesis. The second task is to customize virtual fashion models and synthesize fashion videos on e-commerce platforms. The dataset used in this experiment is a fashion video dataset (Zablotskaia et al. 2019), which contains hundreds of videos. There is a fashion model with fashion clothes in each video. We randomly select 10 source videos from the dataset and fine-tune Stable Diffusion 1.5[8] on each video, respectively. The fine-tuned models have learned the appearance of each fashion model; in other words, one fine-tuned diffusion model represents a virtual fashion model. We randomly select the other 10 target videos, then extract the pose sequence using OpenPose (Cao et al. 2017), and let the 10 virtual fashion models imitate the pose in the other 10 target videos. In this pipeline, we use OpenPose ControlNet[9] to control the pose of models. We notice that the pose extracted by OpenPose suffers from slight tic, although we use Savitzky-Golay smoothing filters (Press and Teukolsky 1990). To further smoothen the synthesized videos, we use Depth ControlNet to stabilize the motion. Finally, we obtain 10×10 realistic synthesized videos.

Hyperparameters. The hyperparameters of the above two tasks are presented in Table 2. These parameters are tuned by human experts. The size of the deflickering window is

---

	Content Consistency (Pixel-MSE ↓)	Prompt Similarity (CLIP Score ↑)	Content Aesthetics (Aesthetic Score ↑)
FateZero	-	22.32	5.44
Pix2Video	1203.04	25.29	5.20
Text2Video-Zero	342.85	23.86	5.13
DiffSynth	**54.97**	24.63	5.37

Table 3: Quantitative results in text-guided video stylization.

	Content Consistency (Pixel-MSE ↓)	Appearance Similarity (FID ↓)	Pose Error (Pose-MSE ↓)
DreamPose	100.44	75.43	0.63
DiffSynth	**24.13**	63.02	0.48

Table 4: Quantitative results in fashion video synthesis.

∞, denoting that we use the fast remapping and blending algorithm to blend all frames. The deflickering frequency denotes how frequently we apply the deflickering algorithm. For example, when the number of denoising steps is 20 and the deflickering frequency is 5, we apply the deflickering algorithm at the 1st, 6th, 11th, and 16th steps. To make the experiments reproducible, we synthesize each video with the same random seed, without any cherry-picking.

Quantitive Results

We evaluate the quality of synthesized videos, with results presented in Table 3 and Table 4. In text-guided video stylization, we compare our approach with several baseline approaches, including FateZero (Qi et al. 2023), Pix2Video (Ceylan, Huang, and Mitra 2023), and Text2Video-Zero (Khachatryan et al. 2023). In fashion video synthesis, we compare it with DreamPose. Following Pix2Video, we calculate Pixel-MSE to evaluate the consistency of the content. Note that the official code of FateZero does not support the resolution except 512×512, so we cannot calculate its Pixel-MSE. From this metric perspective, DiffSynth clearly outperforms other approaches. It shows that our method can generate smoother videos than others. In addition, we computed other metrics to evaluate the quality of each frame. In text-guided video stylization, we employ CLIP score (Radford et al. 2021) to measure the relevance between the synthesized video and the prompt, and use the aesthetic score (Schuhmann et al. 2022) to assess the aesthetics of frames. FateZero can only make minor changes to the frames, failing to generate videos that match the textual description. Pix2Video is excessively focused on textual information, resulting in incoherent frames. Text2Video-Zero performed slightly better than the aforementioned methods but still lags behind our approach. In fashion video synthesis, we employ the Fréchet Inception Distance (FID) (Heusel et al. 2017) to measure the appearance similarity between synthesized videos and source videos, and utilize Pose-MSE (Mean Squared Error of the keypoint distances recognized by OpenPose (Cao et al. 2017)) to assess the pose error between synthesized videos and target videos. Our method exceeds DreamPose in all metrics. These experimental results demonstrate the effectiveness of DiffSynth.

Approach	FateZero	Pix2Video	Text2Video-Zero	DiffSynth
Percentage	24.43	7.65	12.07	**55.85**

Table 5: Percentage of videos selected as best in user study.

(a) The input video

(b) The video synthesized by our methods (w/o deflickering)
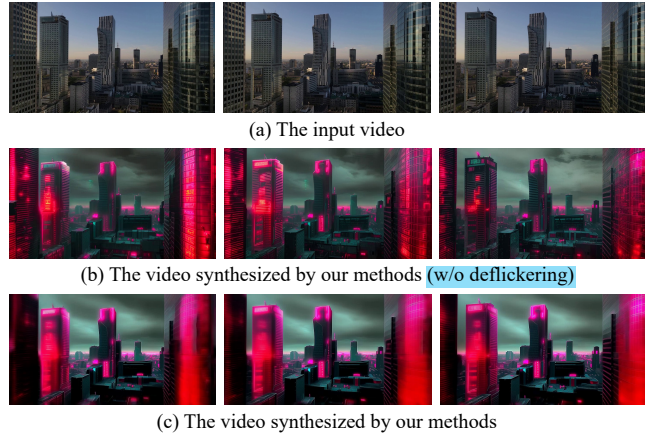
(c) The video synthesized by our methods

Figure 3: An example of ablation study. The prompt of this example is "cyberpunk, city, red neon light".

User Study

Some studies (Blattmann et al. 2023; Yang et al. 2023) have pointed out that conventional metrics are sometimes not feasible. We invite 20 participants and conduct a user study. We ask each participant to select the best results based on video consistancy, text-video similarity, and aesthetics. The average results of the 100 videos in text-guided video stylization are presented in Table 5. Unsurprisingly, most of the participants think that the videos synthesized by DiffSynth look better than others.

Ablation Study

To evaluate the effectiveness of our proposed deflickering algorithm (i.e., the patch blending algorithm), we conduct an ablation experiment. Figure 3 illustrates an example of our ablation study. For more examples, please refer to our project page. When the deflickering algorithm is disabled, it is evident that the video exhibits inconsistencies in various aspects, including the brightness of the sky, the lights on the buildings, and the texts at the central building. When the deflickering algorithm is enabled, these objects are more aligned, resulting in more coherent videos.

Other Applications

Besides the above two tasks, we also design several fancy pipelines in the following scenarios. These video synthesis pipelines are all transferred from image synthesis pipelines.

Image-Guided Video Stylization

In the above stylization pipeline, the frames are synthesized according to input prompts. Practically, the synthesized videos are greatly influenced by the prompts, and carefully tuned prompts can improve the quality of videos.
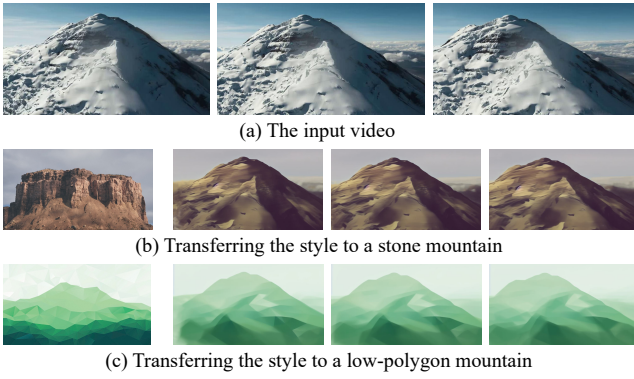
(a) The input video

(b) Transferring the style to a stone mountain

(c) Transferring the style to a low-polygon mountain

Figure 4: Examples of image-guided video stylization.

(a) The input video

(b) The restored video

Figure 5: An example of video restoring.

Prompt engineering has become an interesting but challenging task (Witteveen and Andrews 2022). To intuitively guide the model generating videos, we employ a ControlNet model[10] to further enable the image guiding mechanism. As shown in Figure 4, this pipeline can transfer the style of a content video according to the style of an image. Compared to the above pipeline, this pipeline does not require well-written prompts. Therefore, it is easy to use for creators who are not familiar with diffusion models.

Video Restoring

In earlier years, due to the immaturity of photography, some old videos only have very low resolution and may have hard-to-repair noise because of repeated compression during distribution. Restoration of these videos requires a lot of costs. One method of restoration using deep learning is super-resolution, but the defects such as noise in the original video are also retained in the restored video. We decided to use our video processing method to restore these videos. Combined with Tile ControlNet[11], the Stable Diffusion model can ignore the defects in old videos and redraw noisy video frames. Specifically, we first increase the resolution of the original video using the super-resolution method and then map each frame to the latent space. After adding noise to the intermediate steps, we subsequently use Tile ControlNet for denoising. Unlike in the other application scenarios, we do not generate each frame from scratch

---

[10]https://huggingface.co/lllyasviel/control_v11e_sd15_shuffle

[11]https://huggingface.co/lllyasviel/control_v11f1e_sd15_tile

(a) The input video

(b) The video rendered with prompt "A diver, black suit, black mask, swimming, in the deep sea."

(c) The video rendered with prompt "A cyberpunk robot, diver, white, future technology, swimming, in the swimming pool, red neon light."
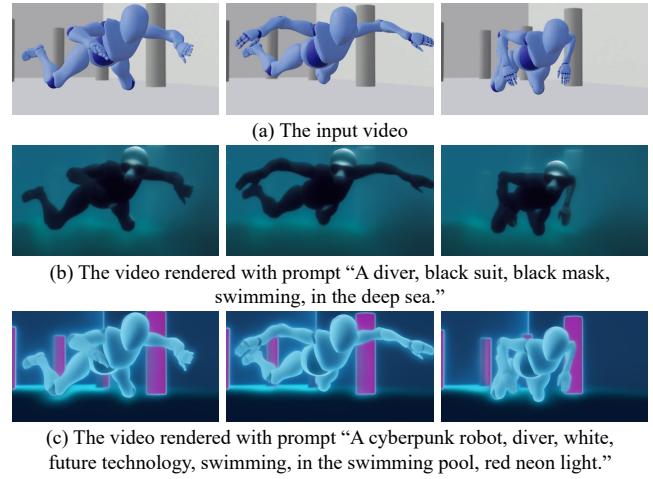
Figure 6: Examples of 3D rendering.

because we tend to preserve as much information as possible in the original video, with only limited modifications to the details. Figure 5 shows an example. We can see that our pipeline is capable of restoring the details of historic videos.

3D Rendering

In the media industry, creators have to draw maps for a 3D object to render a video. Utilizing our approach, we can directly render a video automatically. We first extract some necessary information from the unrendered video frames to represent the structure of frames. There are no unique answers to the definition of what structural information is, and we take reference from Gen-1 (Esser et al. 2023) to extract depth information. In addition, we also use the SoftEdge information if necessary. We then design a ControlNet-based pipeline to render the video. An example is presented in Figure 6. Our pipeline can transform the 3D gray object to a realistic object, and it only requires creators to provide unrendered videos and prompts. Therefore, our work has the capacity of benefiting the media industry by video designs.

Conclusion and Future Work

In this paper, we investigate the application of diffusion models to video synthesis. We propose the latent in-iteration deflickering approach, making it possible to apply existing video deflickering methods to the latent space, thereby avoiding flicker accumulation during the iterative process. We specifically design a deflickering algorithm based on patch matching for diffusion models. With this algorithm, we can synthesize coherent and realistic videos without any cherry-picking. We further show that our approach is applicable to various application scenarios. Comprehensive experimental results demonstrate the effectiveness of our method, outperforming previous methods.

Yet, our work still has a few limitations. Although the $\mathcal{O}(n \log n)$ time complexity can be achieved, the efficiency of our approach can be further improved for wider applications. For example, running on an NVIDIA A10 GPU, our program requires approximately 1 minute to synthesize 1

frame in fashion video synthesis. Furthermore, the blending operator can be improved to generate details better. We leave these problems for future work.

References

Barnes, C.; Shechtman, E.; Finkelstein, A.; and Goldman, D. B. 2009. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3): 24.

Blattmann, A.; Rombach, R.; Ling, H.; Dockhorn, T.; Kim, S. W.; Fidler, S.; and Kreis, K. 2023. Align your latents: High-resolution video synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 22563–22575.

Cao, Z.; Simon, T.; Wei, S.-E.; and Sheikh, Y. 2017. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 7291–7299.

Ceylan, D.; Huang, C.-H. P.; and Mitra, N. J. 2023. Pix2video: Video editing using image diffusion. *arXiv preprint arXiv:2303.12688*.

Dao, T.; Fu, D.; Ermon, S.; Rudra, A.; and Ré, C. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35: 16344–16359.

Dhariwal, P.; and Nichol, A. 2021. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34: 8780–8794.

Esser, P.; Chiu, J.; Atighehchian, P.; Granskog, J.; and Germanidis, A. 2023. Structure and content-guided video synthesis with diffusion models. *arXiv preprint arXiv:2302.03011*.

Feng, Z.; Zhang, Z.; Yu, X.; Fang, Y.; Li, L.; Chen, X.; Lu, Y.; Liu, J.; Yin, W.; Feng, S.; et al. 2023. ERNIE-ViLG 2.0: Improving text-to-image diffusion model with knowledge-enhanced mixture-of-denoising-experts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10135–10145.

Gal, R.; Alaluf, Y.; Atzmon, Y.; Patashnik, O.; Bermano, A. H.; Chechik, G.; and Cohen-or, D. 2022. An Image is Worth One Word: Personalizing Text-to-Image Generation using Textual Inversion. In *The Eleventh International Conference on Learning Representations*.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2020. Generative adversarial networks. *Communications of the ACM*, 63(11): 139–144.

Han, S.; Huang, P.; Wang, H.; Yu, E.; Liu, D.; and Pan, X. 2022. Mat: Motion-aware multi-object tracking. *Neurocomputing*, 476: 75–86.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Heusel, M.; Ramsauer, H.; Unterthiner, T.; Nessler, B.; and Hochreiter, S. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30.

Ho, J.; and Salimans, T. 2021. Classifier-Free Diffusion Guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*.

Hu, E. J.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W.; et al. 2021. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

Huang, X.; Li, Y.; Poursaeed, O.; Hopcroft, J.; and Belongie, S. 2017. Stacked generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5077–5086.

Khachatryan, L.; Movsisyan, A.; Tadevosyan, V.; Henschel, R.; Wang, Z.; Navasardyan, S.; and Shi, H. 2023. Text2video-zero: Text-to-image diffusion models are zero-shot video generators. *arXiv preprint arXiv:2303.13439*.

Kingma, D. P.; and Welling, M. 2013. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*.

Kirillov, A.; Mintun, E.; Ravi, N.; Mao, H.; Rolland, C.; Gustafson, L.; Xiao, T.; Whitehead, S.; Berg, A. C.; Lo, W.-Y.; et al. 2023. Segment anything. *arXiv preprint arXiv:2304.02643*.

Lei, C.; Ren, X.; Zhang, Z.; and Chen, Q. 2023. Blind Video Deflickering by Neural Filtering with a Flawed Atlas. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10439–10448.

Lei, C.; Xing, Y.; and Chen, Q. 2020. Blind video temporal consistency via deep video prior. *Advances in Neural Information Processing Systems*, 33: 1083–1093.

Meng, C.; He, Y.; Song, Y.; Song, J.; Wu, J.; Zhu, J.-Y.; and Ermon, S. 2021. Sdedit: Guided image synthesis and editing with stochastic differential equations. In *International Conference on Learning Representations*.

Nichol, A. Q.; and Dhariwal, P. 2021. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, 8162–8171. PMLR.

Press, W. H.; and Teukolsky, S. A. 1990. Savitzky-Golay smoothing filters. *Computers in Physics*, 4(6): 669–672.

Qi, C.; Cun, X.; Zhang, Y.; Lei, C.; Wang, X.; Shan, Y.; and Chen, Q. 2023. Fatezero: Fusing attentions for zero-shot text-based video editing. *arXiv preprint arXiv:2303.09535*.

Rabe, M. N.; and Staats, C. 2021. Self-attention Does Not Need $O(n^2)$ Memory. *arXiv preprint arXiv:2112.05682*.

Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, 8748–8763. PMLR.

Ramesh, A.; Dhariwal, P.; Nichol, A.; Chu, C.; and Chen, M. 2022. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*.

Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; and Ommer, B. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10684–10695.

Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, 234–241. Springer.

Ruiz, N.; Li, Y.; Jampani, V.; Pritch, Y.; Rubinstein, M.; and Aberman, K. 2023. 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*, 22500–22510.

Saharia, C.; Chan, W.; Saxena, S.; Li, L.; Whang, J.; Denton, E. L.; Ghasemipour, K.; Gontijo Lopes, R.; Karagol Ayan, B.; Salimans, T.; et al. 2022. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35: 36479–36494.

Schuhmann, C.; Beaumont, R.; Vencu, R.; Gordon, C.; Wightman, R.; Cherti, M.; Coombes, T.; Katta, A.; Mullis, C.; Wortsman, M.; et al. 2022. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35: 25278–25294.

Singer, U.; Polyak, A.; Hayes, T.; Yin, X.; An, J.; Zhang, S.; Hu, Q.; Yang, H.; Ashual, O.; Gafni, O.; et al. 2022. Make-A-Video: Text-to-Video Generation without Text-Video Data. In *The Eleventh International Conference on Learning Representations*.

Sohl-Dickstein, J.; Weiss, E.; Maheswaranathan, N.; and Ganguli, S. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, 2256–2265. PMLR.

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

Teed, Z.; and Deng, J. 2020. Raft: Recurrent all-pairs field transforms for optical flow. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, 402–419. Springer.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Witteveen, S.; and Andrews, M. 2022. Investigating prompt engineering in diffusion models. *arXiv preprint arXiv:2211.15462*.

Wu, J. Z.; Ge, Y.; Wang, X.; Lei, W.; Gu, Y.; Hsu, W.; Shan, Y.; Qie, X.; and Shou, M. Z. 2022. Tune-A-Video: One-Shot Tuning of Image Diffusion Models for Text-to-Video Generation. *arXiv preprint arXiv:2212.11565*.

Yang, S.; Zhou, Y.; Liu, Z.; and Loy, C. C. 2023. Rerender A Video: Zero-Shot Text-Guided Video-to-Video Translation. *arXiv preprint arXiv:2306.07954*.

Yu, S.; Tack, J.; Mo, S.; Kim, H.; Kim, J.; Ha, J.-W.; and Shin, J. 2021. Generating Videos with Dynamics-aware Implicit Generative Adversarial Networks. In *International Conference on Learning Representations*.

Zablotskaia, P.; Siarohin, A.; Zhao, B.; and Sigal, L. 2019. Dwnet: Dense warp-based network for pose-guided human video generation. *arXiv preprint arXiv:1910.09139*.

Zhang, L.; and Agrawala, M. 2023. Adding conditional control to text-to-image diffusion models. *arXiv preprint arXiv:2302.05543*.