DiffSynth: Latent In-Iteration Deflickering for Realistic Video Synthesis

Zhongjie Duan[1], Lizhou You[2], Chengyu Wang[3], Cen Chen[1(✉)], Ziheng Wu[4], Weining Qian[1], and Jun Huang[3]

[1] East China Normal University, Shanghai, China
zjduan@stu.ecnu.edu.cn, {cenchen,wnqian}@dase.ecnu.edu.cn
[2] Xiamen University, Xiamen, China
youlizhou@stu.xmu.edu.cn
[3] Alibaba Group, Hangzhou, China
{chengyu.wcy,huangjun.hj}@alibaba-inc.com
[4] Hangzhou, China

Abstract. In recent years, diffusion models have emerged as a powerful approach in the field of image synthesis. However, applying these models directly to video synthesis presents challenges, often leading to noticeable flickering in the content. Although recently proposed zero-shot methods can alleviate flickering to some extent, generating coherent videos remains a struggle. In this paper, we propose DiffSynth, a novel approach that converts image synthesis pipelines into 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 in the latent space of diffusion models, effectively preventing flicker accumulation in intermediate steps. Additionally, we introduce a video deflickering algorithm, named the patch blending algorithm, which remaps objects across different frames and blends them 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 restoration, 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, and we further showcase its practical value on Alibaba e-commerce platform.

Keywords: Video synthesis · Video deflickering · Generative models

1 Introduction

In recent years, diffusion models have achieved remarkable success in image synthesis, surpassing Generative Adversarial Networks (GANs) [5]. Within open-source communities, Stable Diffusion [24] has emerged as the most popular diffusion model, with fine-tuned versions achieving astonishing success across various

Z. Wu—Individual Researcher.

A. Bifet et al. (Eds.): ECML PKDD 2024, LNAI 14950, pp. 332–347, 2024.
https://doi.org/10.1007/978-3-031-70381-2_21

artistic styles. Furthermore, several research breakthroughs have enhanced the capabilities of Stable Diffusion [8,13,26,38]. These achievements have established Stable Diffusion as the mainstream backbone for image synthesis.

In this paper, we further investigate the capabilities of diffusion models in video synthesis. Directly applying image synthesis techniques to each frame typically leads to significant flickering [36], as each frame is synthesized independently. To address this problem, researchers have proposed solutions from various perspectives. One end-to-end solution involves pre-training a video synthesis model using video datasets [2]. However, the computational resources required for training video diffusion models are substantial. Moreover, most existing techniques based on Stable Diffusion cannot be straightforwardly applied to newly trained models, which complicates the control over the generated results of such models. Therefore, we focus on one-shot and zero-shot methods, aiming to adapt existing image synthesis models to video synthesis with minimal or even no training. In recent years, several video synthesis methods have gained popularity. For instance, Tune-A-Video [34] achieves video editing by fine-tuning the textual part of the model, while Text2Video-Zero [16] constrains the content using cross-frame attention. Nonetheless, these methods cannot completely eliminate flickering in videos.

To address the challenges mentioned above, we propose a novel approach named DiffSynth. Specifically, we design a latent in-iteration deflickering framework to remove flickering during the intermediate iterations of video synthesis. In our approach, video-level deflickering is applied to the videos decoded in the latent space. Subsequently, the videos are re-encoded to the latent representations. As a result, we can effectively prevent the accumulation of flickering during the denoising steps. For video deflickering, we devise a patch blending algorithm based on patch matching [1] to ensure optimal performance. By remapping frames to a reference frame, we can capture the appearance features of the same object across different frames. Then, by blending this content, we achieve a consistent video. Integrated with the patch blending algorithm, our in-iteration deflickering framework is capable of synthesizing realistic videos.

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 restoration, and 3D rendering. These video synthesis pipelines are derived from image synthesis pipelines, and we provide hyperparameter templates for these applications. To validate the effectiveness of our method, we conducted extensive experiments across several scenarios. DiffSynth can generate coherent and realistic videos without cherry-picking. In the text-guided video stylization task, DiffSynth significantly outperforms existing baseline methods in quantitative metrics and user studies. Additionally, DiffSynth has been applied to Alibaba e-commerce platform, enabling the generation of realistic fashion videos for product recommendations.

The source codes are released on GitHub[1], and the videos can be viewed on our project page[2]. We summarize the contributions of this paper as follows:

- We introduce DiffSynth, a novel approach for coherent and realistic video synthesis that is compatible with most Stable Diffusion-based models.
- We present a latent in-iteration deflickering framework, which applies video-level deflickering to the latent space of diffusion models, thereby avoiding flicker accumulation throughout the iterative process.
- We develop multiple video synthesis pipelines using DiffSynth for various tasks and demonstrate the superiority of our method through extensive experiments and practical applications.

2 Related Work

2.1 Diffusion Models

Diffusion models [30] represent a class of generative models that transform the image synthesis process into a sequence of denoising steps. In contrast to GAN-based models, diffusion models forego adversarial training, making them generally more straightforward to train. Latent Diffusion [24] pioneers the concept of transitioning images from pixel space to latent space, enabling model training with limited computational resources. Building upon Latent Diffusion, Stable Diffusion has emerged as the preeminent model within research communities. Recent advancements have further amplified the remarkable capabilities of Stable Diffusion across various dimensions. LoRA (Low-Rank Adaptation) [13] minimizes the computational resources needed for fine-tuning by integrating low-rank matrices into the model's architecture. To improve the controllability of images produced by Stable Diffusion, ControlNet [38] applies zero convolution to infuse additional conditional information, guiding the model to produce objects with specific shapes, characters in certain poses, and so forth. Textual Inversion [8] enables the synthesis of particular objects by adding a new special word embedding to the vocabulary and training with a few images. Dreambooth [26] further refines the model's proficiency in generating specific objects. Our proposed method, DiffSynth, is fully compatible with these techniques, transferring their capabilities into video synthesis.

2.2 Diffusion-Based Video Synthesis

Inspired by the remarkable success of diffusion models, researchers have endeavored to adapt diffusion models to video synthesis tasks such as text-to-video synthesis, video style transfer, and video editing. For instance, Make-A-Video [29] and VideoLDM [2] extend a text-to-image diffusion model into a text-to-video diffusion model by incorporating temporal blocks. Gen-1 [6] employs a

[1] https://github.com/alibaba/EasyNLP/tree/master/diffusion/DiffSynth.
[2] https://ecnu-cilab.github.io/DiffSynth.github.io/.

similar temporal architecture to achieve video style transfer. Beyond pre-training large-scale video models, recent studies have concentrated on synthesizing videos using image models. Tune-A-Video [34] fine-tunes select modules with input video data, thereby enabling video editing in response to given prompts. Zero-shot methods, including FateZero [21], Pix2Video [4], and Text2Video-Zero [16], explore the potential of video synthesis without the need for additional training. These collective efforts underscore the viability of applying diffusion models to video synthesis tasks. Building on these advancements, we propose a novel zero-shot approach that efficiently transforms existing image synthesis pipelines into video synthesis pipelines.

3 Methodology

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

3.1 Preliminaries

Diffusion models encompass various architectures [7,23,27]. In this paper, we focus primarily on Stable Diffusion [24], which is the most popular open-source architecture. Stable Diffusion comprises the following three components:

- **Text Encoder**. A transformer-based language model from CLIP [22], which converts text into text embeddings. These embeddings are then utilized in classifier-free guidance [12].
- **U-Net** [25]. A vision model, denoted as ϵ, equipped with self-attention [32], cross-attention, and residual connections [10]. This model is trained to perform denoising in the latent space.
- **VAE** (Variational Autoencoder) [17]. This component consists of an encoder $\mathcal{E}$ and a decoder $\mathcal{D}$. The encoder maps images to latent tensors, while the decoder reconstructs images from these latent representations.

Both the diffusion process and its reverse process (i.e., the generation process) are conducted in the latent space. The process involves $T+1$ steps with $\{0, 1, \ldots, T\}$ representing different levels of noise. During the generation process, we begin with a noise tensor x_T sampled from a Gaussian distribution and denoise it step by step. The iterative formula for each step t is

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

where α_t is a hyperparameter that describes the amount of noise at step t. After completing the iterative process, the latent tensor x_0 is decoded into an image $X = \mathcal{D}(x_0)$. Beyond text-to-image synthesis, this process can be adapted for other pipelines. For instance, by adding noise to an image and denoising from an intermediate step, we can create an image-to-image pipeline for image editing.

To transform an image synthesis pipeline into a video synthesis pipeline, a simple approach would be to generate each frame independently. However, models based on Stable Diffusion, which are primarily trained on text-image datasets, do not inherently possess the capability to generate coherent video frames. As a result, the discrepancy between two adjacent frames tends to increase with each iterative step. This difference accumulates throughout the generation process, ultimately leading to irreparable flickering and inconsistency.

3.2 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 [19], 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 the latent tensors at the next step $\{x_{t-1}^1, x_{t-1}^2, \ldots, x_{t-1}^n\}$ using formula (1), these latent tensors will become more inconsistent. To visualize the latent tensors, we first skip to the final step to calculate the estimation of $\{x_0^1, x_0^2, \ldots, x_0^n\}$:

$$\hat{x}_0^i = \frac{x_t^i - \sqrt{1-\alpha_t}\ \epsilon(x_t^i)}{\sqrt{\alpha_t}}. \tag{2}$$

Then, we decode $\{\hat{x}_0^1, \hat{x}_0^2, \ldots, \hat{x}_0^n\}$ to images using the decoder of VAE, i.e., $\hat{X}^i = \mathcal{D}(\hat{x}_0^i)$. Theoretically, $\{\hat{X}^1, \hat{X}^2, \ldots, \hat{X}^n\}$ represent the frames when we denoise along with a straight line directed by the vectors $\{\epsilon(x_t^1), \epsilon(x_t^2), \ldots, \epsilon(x_t^n)\}$. We employ a video-level deflickering method $\mathcal{F}$ to make the video coherent:

$$\{\overline{X}^1, \overline{X}^2, \ldots, \overline{X}^n\} = \mathcal{F}\{\hat{X}^1, \hat{X}^2, \ldots, \hat{X}^n\}, \tag{3}$$

where the deflickering method $\mathcal{F}$ will be detailed in the next subsection. Next, we encode the processed frames into the latent space, i.e. $\overline{x}_0^i = \mathcal{E}(\overline{X}^i)$. To synthesize the video frames $\{\overline{X}^1, \overline{X}^2, \ldots, \overline{X}^n\}$, the predicted noise at step t should be:

$$\overline{\epsilon}(x_t^i) = \frac{x_t^i - \sqrt{\alpha_t}\ \overline{x}_0^i}{\sqrt{1-\alpha_t}}. \tag{4}$$

Thus, we obtain the reconstructed iterative formula as:

$$x_{t-1}^i = \sqrt{\alpha_{t-1}}\ \overline{x}_0^i + \sqrt{1-\alpha_{t-1}}\ \overline{\epsilon}(x_t^i). \tag{5}$$

The pipeline (2–5) 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.

3.3 Patch Blending Algorithm

Addressing another challenge, we detail the design of the video-level deflickering method $\mathcal{F}$. In many video synthesis tasks (such as style transfer and video

editing), an original video is available for reference. The original video provides essential information for maintaining video consistency, therefore, we mainly address these scenarios and propose the following deflickering algorithm.

Assume we have synthesized frames $\{\hat{X}^1, \hat{X}^2, \ldots, \hat{X}^n\}$ through diffusion models in a single denoising step. Our goal is to smoothen the video by referencing the original video frames $\{X^1, X^2, \ldots, X^n\}$. If an object in frame $\hat{X}^i$ also appears in frame $\hat{X}^j$, we aim to remap the corresponding regions in $\hat{X}^j$ to $\hat{X}^i$ before blending them. The blended frame will exhibit consistent information from both $\hat{X}^i$ and $\hat{X}^j$ if the remapping is accurate. While recent advances in object segmentation and tracking [9,18] are commendable, pixel-level matching remains elusive. Optical flow [31] was considered but found to be typically imprecise with large frame intervals. Consequently, we opt for a patch matching algorithm [1], which effectively estimates correspondences between frames. In this algorithm, frames X^i and X^j are segmented into overlapping patches. We first calculate the *nearest neighbor field* (NNF) to identify matched patches and then reconstruct $\hat{X}^i$ using $\hat{X}^j$ [14], a process we refer to as image remapping. The final step involves blending the remapped frames. Formally, $\overline{X}^i = \frac{1}{n}\sum_{j=1}^{n} \hat{X}^i(\hat{X}^j)$, where $\hat{X}^i(\hat{X}^j)$ represents $\hat{X}^i$ reconstructed with $\hat{X}^j$. During blending, remapped neighboring frames mask the flickering content, resulting in a coherent video. The pseudocode for the denoising loop is presented in Algorithm 1.

Algorithm 1: The denoising loop of DiffSynth.

Input: A black box denoising function ϵ.
$\{x_T^1, x_T^2, \ldots, x_T^n\} \sim \mathcal{N}(\boldsymbol{O}, \boldsymbol{I})$;
for $t = T, T-1, \ldots, 1$ **do**
  for $i = 1, 2, \ldots, n$ **do**
    $\hat{x}_0^i \leftarrow \frac{x_t^i - \sqrt{1-\alpha_t}\ \epsilon(x_t^i)}{\sqrt{\alpha_t}}$;
    $\hat{X}^i \leftarrow \mathcal{D}(\hat{x}_0^i)$; ▷ Latent space → pixel space
  end
  for $i = 1, 2, \ldots, n$ **do**
    $\overline{X}^i \leftarrow \frac{1}{n}\sum_{j=1}^{n} \hat{X}^i(\hat{X}^j)$; ▷ Patch blending algorithm
    $\overline{x}_0^i \leftarrow \mathcal{E}\left(\overline{X}^i\right)$;
    $x_{t-1}^i \leftarrow \sqrt{\alpha_{t-1}}\ \overline{x}_0^i + \sqrt{1-\alpha_{t-1}}\ \left(\frac{x_t^i - \sqrt{\alpha_t}\ \overline{x}_0^i}{\sqrt{1-\alpha_t}}\right)$;
    ▷ Pixel space → latent space
  end
end
for $i = 1, 2, \ldots, n$ **do**
  $\hat{X}^i \leftarrow \mathcal{D}(x_0^i)$; ▷ Latent space → pixel space
  $\overline{X}^i \leftarrow \frac{1}{n}\sum_{j=1}^{n} \hat{X}^i(\hat{X}^j)$; ▷ Patch blending algorithm
end
Output: Synthesized video frames $\{\overline{X}^1, \overline{X}^2, \ldots, \overline{X}^n\}$.

4 Experiments

To demonstrate the effectiveness of our approach, we conducted comparative experiments in text-guided video stylization. Examples of this task are presented in Fig. 1. We evaluated the synthesized videos using both quantitative metrics and user studies.

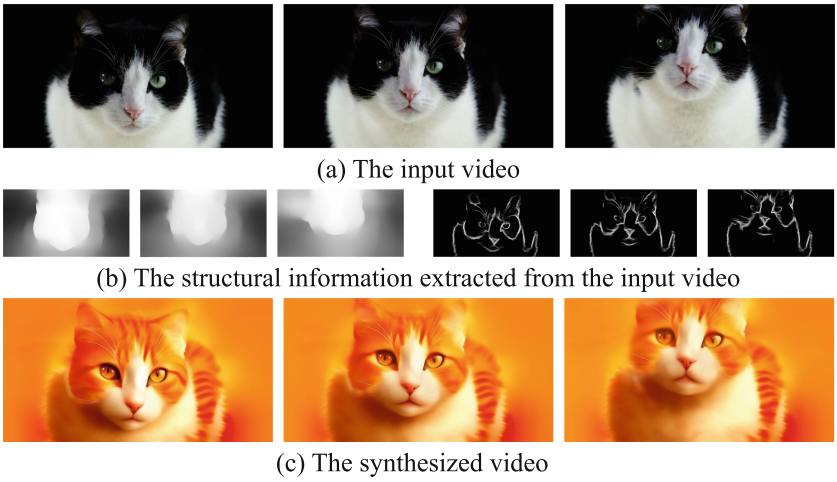


Fig. 1. An example of text-guided video stylization. The prompt in this example is "an orange and white cat". We recommend readers to see the videos on our project page. (Color figure online)

4.1 Experimental Settings

In the text-guided video stylization task, we design a pipeline to transfer the style of videos according to the given prompts. The dataset, consisting of 100 high-resolution videos, is collected from a community[3]. Each video is cut into 3 to 5 s, including at most 150 frames. We manually write prompts for each video. This dataset has been released on Github[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 structural guidance. Cross-frame attention [16] is enabled in these models. The information in the original video is only delivered to the pipeline by ControlNet, thus the color is completely ignored. The resolution is 512×960, the number of denoising steps is 20, and the classifier-free guidance scale [12] is 7.5. The patch blending algorithm is applied

[3] https://pixabay.com/.
[4] https://github.com/ECNU-CILAB/Pixabay100.
[5] https://civitai.com/models/4384/dreamshaper.
[6] https://huggingface.co/lllyasviel/control_v11f1p_sd15_depth.
[7] https://huggingface.co/lllyasviel/control_v11p_sd15_softedge.

in the 1st, 6th, 11th, and 16th steps. We further improve the contrast ratio and sharpen the frames slightly to improve the video quality. These parameters are tuned by human experts. To make the experiments reproducible, we synthesize each video with the same random seed, without any cherry-picking.

4.2 Quantitive Comparison

We compare our approach with several baseline approaches, including FateZero [21], Pix2Video [4], and Text2Video-Zero [16].

Table 1. Content alignment statistics in the video stylization task.

	Prompt similarity (CLIP Score ↑)	Content aesthetics (Aesthetic Score ↑)
FateZero	22.32	**5.44**
Pix2Video	**25.29**	5.20
Text2Video-Zero	23.86	5.13
DiffSynth	<u>24.63</u>	<u>5.37</u>

Table 2. Quantitive comparison between DiffSynth and other baseline methods.

	Content consistency			User preference
	Pixel-MSE-1 ↓	Pixel-MSE-4 ↓	Pixel-MSE-16 ↓	Percentage ↑
FateZero	–	–	–	24.43
Pix2Video	1203.04	1694.77	2722.65	7.65
Text2Video-Zero	342.85	726.49	1492.42	12.07
DiffSynth	**54.97**	**221.05**	**720.88**	**55.85**

Firstly, we evaluated the alignment of content generated by each method. Note that content alignment largely depends on the model used for generation. We measured the similarity between video content and input text using CLIP score [22], and the aesthetic quality of video content using aesthetic score [28]. The content alignment statistics are presented in Table 1, and a visual comparison example is shown in Fig. 2. Owing to the strong generative capabilities of the diffusion model, all of these methods are capable of generating visually pleasing video content based on the given prompt, and discernible gaps in content alignment are not observed among these methods.

Secondly, we evaluated the content consistency of the generated videos. Following Pix2Video [4], we calculate Pixel-MSE-x to evaluate the consistency of the content. Pixel-MSE-x is the mean square error between the warped frame and its corresponding target frame, where the warped frame is computed based on the optical flow and x denotes the distance between the two frames. Pixel-MSE-1

is a metric indicative of the coherence between consecutive frames, while Pixel-MSE-16 is employed to assess the long-term consistency over a more extended temporal span. 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 approach can preserve the fantastic generative performance of the diffusion model while significantly improving the consistency.

Additionally, some studies [2,35] have pointed out that conventional metrics are sometimes not feasible. We further invite 20 participants and conduct a user study. We ask each participant to select the best results based on video consistency, text-video similarity, and aesthetics. The average results of the 100 videos in text-guided video stylization are presented in Table 2. Most participants think that the videos synthesized by DiffSynth look better than others.

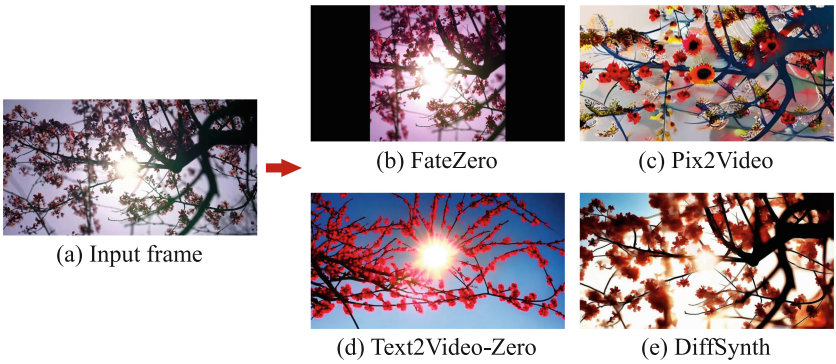


Fig. 2. Visual comparison between DiffSynth and other baseline methods. The prompt is "sun, tree, red flowers". Benefiting from the powerful generative capabilities of diffusion models, each method can transform the pink flowers into red. (Color figure online)

4.3 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. 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.

Additionally, we validate the effectiveness of the image remapping method in the patch blending algorithm. Here, we provided an example in Fig. 5. In the input video, we observe the rotation of the second tiger's head. In the frame obtained by optical flow estimation (Fig. 5(d)), the head is excessively compressed. The frame remapped by DiffSynth (Fig. 5(c)) is more consistent with the input video (Fig. 5(a)). This observation underscores the capability of the

remapping method in DiffSynth to reconstruct images based on the input video faithfully.

5 More Pipelines for Video Synthesis Applications

5.1 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 are always necessary. Prompt engineering has become an interesting but challenging task [33]. To intuitively guide the model to generate videos, we employ a ControlNet model[8] to further enable the image guiding mechanism. As shown in Fig. 4.A, 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.

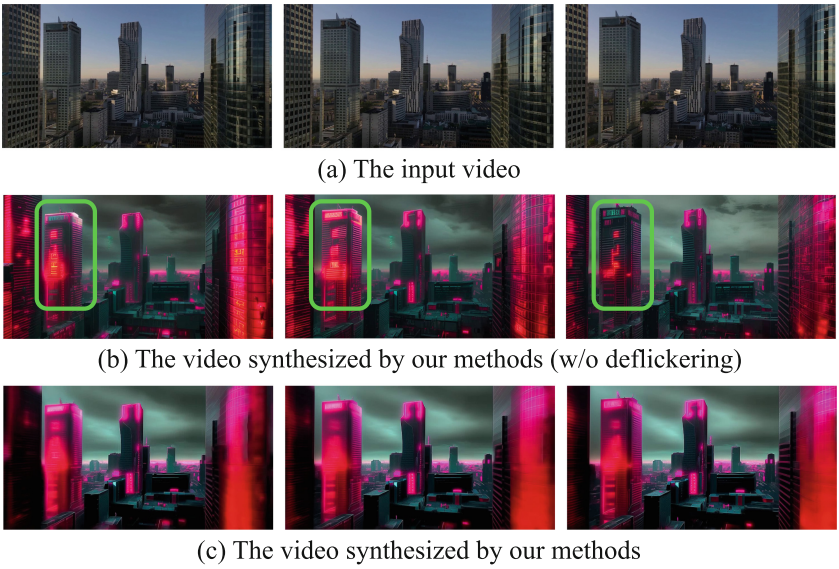


Fig. 3. An example of ablation study. The prompt of this example is "cyberpunk, city, red neon light". The light on the buildings is more consistent when the deflickering algorithm is enabled. We recommend readers to see the videos on our project page. (Color figure online)

5.2 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

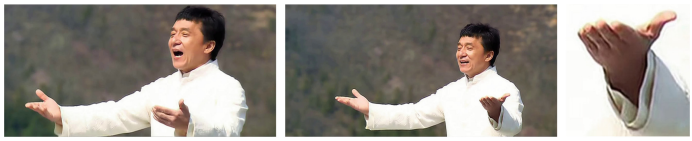
[8] https://huggingface.co/lllyasviel/control_v11e_sd15_shuffle.

(a) The input video

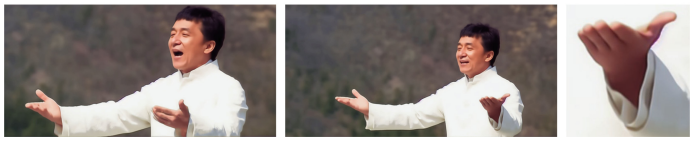
(b) Transferring the style to a stone mountain

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

A. Examples of image-guided video stylization.



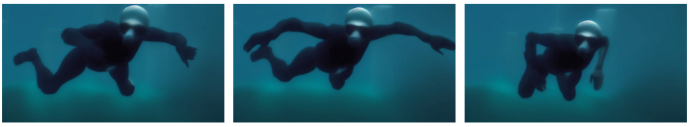
(a) The input video



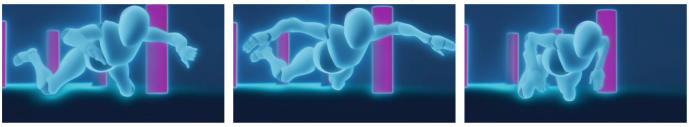
(b) The restored video

B. Examples of video restoring.

(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.”

C. Examples of 3D rendering.

Fig. 4. Examples in other tasks.

compression during distribution. Restoration of these videos is a difficult task. 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[9], 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 pure noise because we tend to preserve some information in the original video, with only limited modifications to the details. Figure 4.B shows some examples. We can see that our pipeline is capable of restoring the details of historic videos.

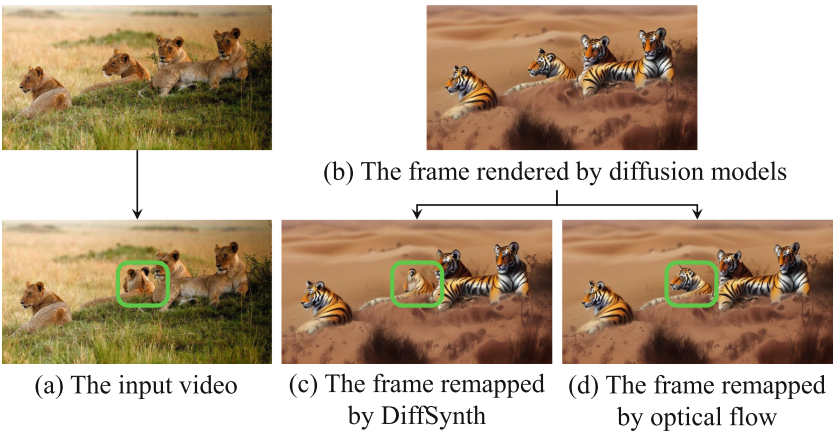


Fig. 5. An example of frames remapped by different methods. The prompt in this example is "tigers, desert". DiffSynth can accurately reproduce the motion in the input video. Please see the second tiger's head.

Table 3. Quantitative results in fashion video synthesis.

	Content Consistency (Pixel-MSE-1 ↓)	Appearance Similarity (FID ↓)	Pose Error (Pose-MSE ↓)
DreamPose	100.44	75.43	0.63
DiffSynth	**24.13**	**63.02**	**0.48**

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

5.3 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 the frames. There are no unique answers to the definition of what structural information is, and we take reference from Gen-1 [6] 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 Fig. 4.C. Our pipeline can transform the 3D gray object into a realistic object, and it only requires creators to provide unrendered videos and prompts. Therefore, our work has the capacity to benefit the media industry through video designs.

6 Industrial Application

In this section, we briefly present a real-world use case of DiffSynth on Alibaba e-commerce platform. We design a pipeline for fashion video synthesis. As illustrated in Fig. 6, this pipeline consists of four stages. In the first stage, we invite fashion models to shoot photos and videos as training data, where the data usage is completely authorized. In the second stage, using the dataset collected from each fashion model, we fine-tune the diffusion model using LoRA. The fine-tuned models have learned the visual features of fashion models, in other words, a fine-tuned diffusion model represents a virtual fashion model. In the third stage, we

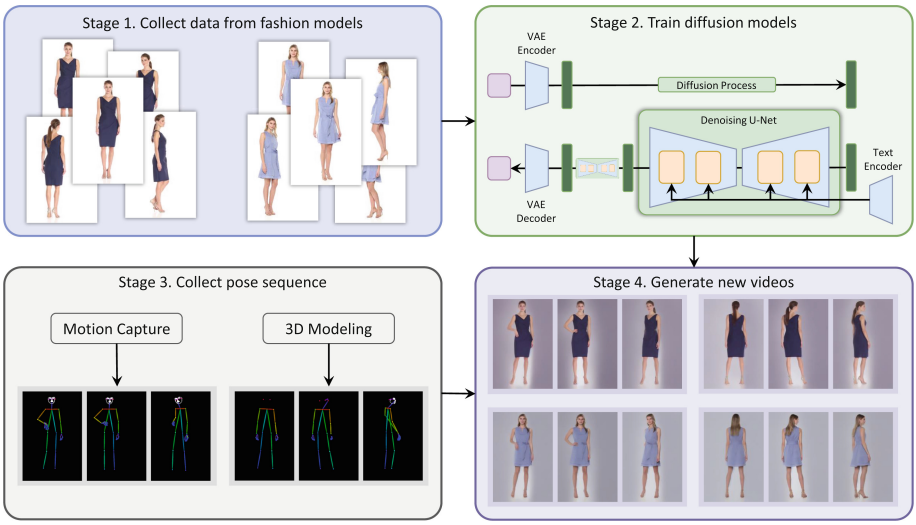


Fig. 6. The pipeline of fashion video synthesis. By fine-tuning diffusion models using the data collected from fashion models, we create virtual fashion models. Given a pose sequence, DiffSynth can let the virtual fashion models perform specific actions.

construct pose sequences. Pose sequences can be obtained using motion capture or 3D modeling techniques. In the final stage, we use DiffSynth to generate realistic videos based on the pose sequences, allowing the virtual fashion model to perform specific actions.

We evaluate this pipeline using a fashion video dataset [37], which contains hundreds of videos. We randomly select 10 source videos from the dataset and fine-tune the diffusion model on each video, respectively. After that, we randomly select the other 10 target videos, then extract the pose sequence using OpenPose [3], and let the 10 virtual fashion models imitate the pose in the other 10 target videos. In this pipeline, we use OpenPose[10] and Depth ControlNet models to control the pose of models. We notice that the pose extracted by OpenPose suffers from slight tic, thus we apply Savitzky-Golay smoothing filters [20] to the pose sequence. Finally, we obtain 10×10 realistic synthesized videos. We compare DiffSynth with DreamPose [15]. The evaluation metrics include Pixel-MSE-1, FID (FrÃľchet Inception Distance [11]) and Pose-MSE (Mean Squared Error of the keypoint distances recognized by OpenPose [3]). The experimental results of fashion video synthesis are presented in Table 3. We can see that our method exceeds DreamPose in all metrics.

For real-world usage, our pipeline is employed to create virtual fashion figures on an e-commerce platform. For online retailers, our service allows them to create their own virtual fashion models using only a few images. The fashion videos are then generated by our pipeline, and can be presented after manual checking, in case of the generation of inappropriate content (although unlikely).

7 Conclusion and Future Work

In this paper, we investigate the application of diffusion models in video synthesis. We propose DiffSynth, a 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. With this algorithm, we can easily synthesize realistic videos. We further show that our approach is applicable to various application scenarios. Comprehensive experimental results demonstrate that our method significantly outperforms previous methods. Yet, our work still has a few limitations. Future efforts will focus on optimizing the algorithm for speed to support real-time video synthesis applications.

Acknowledgments. This work was supported by the National Natural Science Foundation of China under grant Number 62202170, Fundamental Research Funds for the Central Universities under grant Number YBNLTS2023-014, and Alibaba Group through the Alibaba Innovation Research Program.

[10] https://huggingface.co/lllyasviel/control_v11p_sd15_openpose.

References

1. Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D.B.: PatchMatch: a randomized correspondence algorithm for structural image editing. ACM Trans. Graph. **28**(3), 24 (2009)
2. Blattmann, A., et al.: 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 (2023)
3. Cao, Z., Simon, T., Wei, S.E., Sheikh, Y.: Realtime multi-person 2D pose estimation using part affinity fields. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7291–7299 (2017)
4. Ceylan, D., Huang, C.H.P., Mitra, N.J.: Pix2video: video editing using image diffusion. arXiv preprint arXiv:2303.12688 (2023)
5. Dhariwal, P., Nichol, A.: Diffusion models beat GANs on image synthesis. Adv. Neural. Inf. Process. Syst. **34**, 8780–8794 (2021)
6. Esser, P., Chiu, J., Atighehchian, P., Granskog, J., Germanidis, A.: Structure and content-guided video synthesis with diffusion models. arXiv preprint arXiv:2302.03011 (2023)
7. Feng, Z., et al.: 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, pp. 10135–10145 (2023)
8. Gal, R., et al.: An image is worth one word: personalizing text-to-image generation using textual inversion. In: The Eleventh International Conference on Learning Representations (2022)
9. Han, S., Huang, P., Wang, H., Yu, E., Liu, D., Pan, X.: MAT: motion-aware multi-object tracking. Neurocomputing **476**, 75–86 (2022)
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
11. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
12. Ho, J., Salimans, T.: Classifier-free diffusion guidance. In: NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications (2021)
13. Hu, E.J., et al.: LoRA: low-rank adaptation of large language models. In: International Conference on Learning Representations (2021)
14. Jamriška, O., et al.: Stylizing video by example. ACM Transactions on Graphics (TOG) **38**(4), 1–11 (2019)
15. Karras, J., Holynski, A., Wang, T.C., Kemelmacher-Shlizerman, I.: DreamPose: fashion image-to-video synthesis via stable diffusion. arXiv preprint arXiv:2304.06025 (2023)
16. Khachatryan, L., et al.: Text2Video-Zero: text-to-image diffusion models are zero-shot video generators. arXiv preprint arXiv:2303.13439 (2023)
17. Kingma, D.P., Welling, M.: Auto-encoding variational Bayes. In: International Conference on Learning Representations (2013)
18. Kirillov, A., et al.: Segment anything. arXiv preprint arXiv:2304.02643 (2023)
19. Lei, C., Ren, X., Zhang, Z., Chen, Q.: Blind video deflickering by neural filtering with a flawed atlas. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10439–10448 (2023)

20. Press, W.H., Teukolsky, S.A.: Savitzky-Golay smoothing filters. Comput. Phys. **4**(6), 669–672 (1990)
21. Qi, C., et al.: FateZero: fusing attentions for zero-shot text-based video editing. arXiv preprint arXiv:2303.09535 (2023)
22. Radford, A., et al.: Learning transferable visual models from natural language supervision. In: International Conference on Machine Learning, pp. 8748–8763. PMLR (2021)
23. Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., Chen, M.: Hierarchical text-conditional image generation with CLIP latents. arXiv preprint arXiv:2204.06125 (2022)
24. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10684–10695 (2022)
25. Ronneberger, O., Fischer, P., Brox, T.: U-Net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III, pp. 234–241. Springer International Publishing, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28
26. Ruiz, N., Li, Y., Jampani, V., Pritch, Y., Rubinstein, M., Aberman, K.: 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, pp. 22500–22510 (2023)
27. Saharia, C., et al.: Photorealistic text-to-image diffusion models with deep language understanding. Adv. Neural. Inf. Process. Syst. **35**, 36479–36494 (2022)
28. Schuhmann, C., et al.: Laion-5b: an open large-scale dataset for training next generation image-text models. Adv. Neural. Inf. Process. Syst. **35**, 25278–25294 (2022)
29. Singer, U., et al.: Make-a-Video: text-to-video generation without text-video data. In: The Eleventh International Conference on Learning Representations (2022)
30. Song, Y., Ermon, S.: Generative modeling by estimating gradients of the data distribution. In: Advances in Neural Information Processing Systems, vol. 32 (2019)
31. Teed, Z., Deng, J.: RAFT: recurrent all-pairs field transforms for optical flow. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II, pp. 402–419. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-58536-5_24
32. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
33. Witteveen, S., Andrews, M.: Investigating prompt engineering in diffusion models. arXiv preprint arXiv:2211.15462 (2022)
34. Wu, J.Z., et al.: Tune-a-Video: one-shot tuning of image diffusion models for text-to-video generation. arXiv preprint arXiv:2212.11565 (2022)
35. Yang, S., Zhou, Y., Liu, Z., Loy, C.C.: Rerender a Video: zero-shot text-guided video-to-video translation. arXiv preprint arXiv:2306.07954 (2023)
36. Yu, S., et al.: Generating videos with dynamics-aware implicit generative adversarial networks. In: International Conference on Learning Representations (2021)
37. Zablotskaia, P., Siarohin, A., Zhao, B., Sigal, L.: DwNet: dense warp-based network for pose-guided human video generation. arXiv preprint arXiv:1910.09139 (2019)
38. Zhang, L., Agrawala, M.: Adding conditional control to text-to-image diffusion models. arXiv preprint arXiv:2302.05543 (2023)