FastBlend: Enhancing Video Stylization Consistency via Model-Free Patch Blending

Zhongjie Duan[1,2] , **Chengyu Wang**[2] , **Cen Chen**[1] , **Weining Qian**[1] , **Jun Huang**[2] , **Mingyi Jin**[3]
[1]East China Normal University
[2]Alibaba Group
[3]Individual Researcher
zjduan@stu.ecnu.edu.cn, {chengyu.wcy,huangjun.hj}@alibaba-inc.com,
{cenchen, wnqian}@dase.ecnu.edu.cn, jinmingyi1998@sina.cn

Abstract

With the emergence of diffusion models and the rapid development of image processing, generating artistic images in style transfer tasks has become effortless. However, these impressive image processing approaches face consistency issues in video processing due to the independent processing of each frame. In this paper, we propose a powerful, model-free approach called FastBlend to address the consistency problem in video stylization. FastBlend functions as a post-processor and can be seamlessly integrated with diffusion models to create a robust video stylization pipeline. Based on a patch-matching algorithm, we remap and blend the aligned content across multiple frames, thus compensating for inconsistent content with neighboring frames. Moreover, we propose a tree-like data structure and a specialized loss function, aiming to optimize computational efficiency and visual quality for different application scenarios. Extensive experiments have demonstrated the effectiveness of FastBlend. Compared with both independent video deflickering algorithms and diffusion-based video processing methods, FastBlend is capable of synthesizing more coherent and realistic videos.

1 Introduction

In recent years, there has been rapid development in the field of image processing. Notably, diffusion models [Saharia *et al.*, 2022; Ramesh *et al.*, 2022] trained on large-scale datasets have ushered in a transformative era in image synthesis. It has been demonstrated that diffusion models outperform Generative Adversarial Networks (GANs) [Goodfellow *et al.*, 2014] comprehensively [Dhariwal and Nichol, 2021], even reaching a level of creative ability comparable to that of human artists [Yang *et al.*, 2022]. Stable Diffusion [Rombach *et al.*, 2022], which has become the most popular model architecture in open-source communities, has been applied to various domains, including image style transfer [Mou *et al.*, 2023], super-resolution [Li *et al.*, 2022], and image editing [Hertz *et al.*, 2022], achieving noteworthy milestones in diffusion-based approaches.

However, when extending these image processing techniques to video processing, we encounter the issue of maintaining video consistency [Yu *et al.*, 2021; Lei *et al.*, 2023; Yang *et al.*, 2023], particularly in video stylization. Since each frame in a video is processed independently, the direct application of image processing methods typically results in incoherent content, leading to noticeable flickering in generated videos. Recent approaches proposed to enhance the consistency of generated videos can be summarized as follows: 1) Pre-trained large models for coherent video synthesis [Blattmann *et al.*, 2023a; Guo *et al.*, 2023], 2) Zero-shot video processing utilizing image models [Qi *et al.*, 2023; Ceylan *et al.*, 2023; Khachatryan *et al.*, 2023], 3) Video deflickering external to diffusion models [Lei *et al.*, 2023; Ouyang *et al.*, 2023]. Despite these efforts, highlighted in a recent survey [Xing *et al.*, 2023], challenges remain. Pre-training large models on large-scale video datasets demands extremely high computational resources, and current models like Stable Video Diffusion [Blattmann *et al.*, 2023a] can only generate minimal movement. Other studies aim to transfer image diffusion models to video synthesis without additional training. Some zero-shot methods [Qi *et al.*, 2023; Ceylan *et al.*, 2023; Khachatryan *et al.*, 2023] have improved video consistency by modifying the generation process of diffusion models. Video post-processing methods [Lei *et al.*, 2023], which do not alter the diffusion models' generation process, can also be employed in the video processing pipeline. However, they often yield sub-optimal outcomes, struggling to leverage motion features from the input video and to address issues such as frame flickering and tearing effectively.

In this paper, we propose a model-free approach called *FastBlend*, aiming to enhance video stylization consistency. To ensure compatibility with existing methods, we operate exclusively within the image space rather than the latent space [Rombach *et al.*, 2022], thus avoiding modifications to diffusion models. Consequently, *FastBlend* functions as a post-processor and can be seamlessly integrated with diffusion models to create a robust video stylization pipeline. In video stylization tasks, we first employ a diffusion-based image processing pipeline to process each frame, transferring the overall style in accordance with given prompts. Subsequently, we leverage a patch-matching algorithm [Barnes *et al.*, 2009] to estimate Nearest Neighbor Fields (NNFs) [Mount, 2010], which contain the necessary information for remapping images. On the videos processed by diffusion models, we utilize these NNFs to align and blend content within a sliding window, thereby improving video consistency while maintaining visual quality. To enhance computational efficiency, and drawing inspiration from tree-like data structures [Fenwick, 1994; De Berg, 2000], we devise a new data structure optimized for video blending, ensuring minimal time complexity. Furthermore, to improve visual quality, we introduce a novel alignment loss designed for precise content alignment across different frames. In our experiments, we construct a video dataset for evaluation, which will be publicly released. Human evaluation results unanimously indicate that *FastBlend* significantly outperforms baseline methods. Additionally, we implement *FastBlend* with a focus on highly parallel processing on GPUs [Luebke, 2008], achieving exceptional computational efficiency. The source code has been released on GitHub[1]. In summary, the main contributions of this paper include:

- We introduce *FastBlend*, an effective approach for producing consistent videos, making it possible to directly apply existing imege processing methods to video stylization tasks.
- We devise an efficient algorithmic framework for video consistency enhancement, including tree-like data structures and a highly parallel computational architecture, leading to remarkable computational efficiency.
- We construct a video dataset for the evaluation of video stylization techniques, demonstrating through human evaluation that our proposed method consistently outperforms baseline methods, including both standalone video deflickering approaches and diffusion-based methods.

2 Related Work

2.1 Image Synthesis with Diffusion Models

Diffusion models represent a class of generative models that generate images through iterative denoising processes. Stable Diffusion [Rombach *et al.*, 2022], trained on a large-scale text-image dataset [Schuhmann *et al.*, 2022], has emerged as a powerful backbone for image synthesis. Methods based on Stable Diffusion have achieved impressive success. For instance, ControlNet [Zhang *et al.*, 2023] and T2I-Adapter [Mou *et al.*, 2023] enable the redrawing of appearances while preserving the underlying image structure and the transformation of hand-drawn sketches into realistic photographs. Techniques such as Textual Inversion [Gal *et al.*, 2022], LoRA [Hu *et al.*, 2021], and DreamBooth [Ruiz *et al.*, 2023] provide the flexibility to fine-tune Stable Diffusion for generating specific objects. In the realm of image editing, approaches such as Prompt-to-Prompt [Hertz *et al.*, 2022], and InstructPix2Pix [Brooks *et al.*, 2023] are capable of editing images according to user inputs in the form of text or sketches. Besides these diffusion-based methods, techniques such as Real-ESRGAN [Wang *et al.*, 2021], CodeFormer [Zhou *et al.*, 2022], and other image super-resolution and restoration methods can be combined with diffusion models to further enhance image quality. These image synthesis methods have inspired subsequent advancements in video processing.

2.2 Video Stylization with Diffusion Models

Unlike image processing, video processing presents additional challenges, often requiring more computational resources to ensure video consistency. Recent research focuses on extending image diffusion models to video stylization. For example, Gen-1 [Esser *et al.*, 2023] incorporates temporal structures into a diffusion model, training it to restyle videos. Training a video diffusion model is a resource-intensive endeavor, prompting researchers to explore zero-shot video stylization methods based on diffusion models from open-source communities. Examples include Text2LIVE [Bar-Tal *et al.*, 2022], FateZero [Qi *et al.*, 2023], Pix2Video [Ceylan *et al.*, 2023], and Text2Video-Zero [Khachatryan *et al.*, 2023]. These zero-shot video stylization methods process videos frame by frame, which requires substantial computational resources and poses challenges in maintaining video consistency. To address these issues, methods like Make-A-Video [Singer *et al.*, 2022] and Rerender-A-Video [Yang *et al.*, 2023] employ keyframe rendering and video interpolation [Jamriška *et al.*, 2019] to enhance video consistency. Similarly, CoDeF [Ouyang *et al.*, 2023] aims to render an entire video using only a single keyframe. Moreover, independent video deflickering algorithms such as All-In-One Deflicker [Lei *et al.*, 2023] can be integrated with diffusion models to marginally enhance video consistency, and some patch-based methods [Barnes *et al.*, 2009; Jamriška *et al.*, 2019] can render fluent videos using several keyframes, which motivates us to design a general approach for video consistency enhancement.

3 Methodology

3.1 Overview

FastBlend is a model-free algorithm. To ensure compatibility, we use *FastBlend* solely as a post-processing method, without altering the generative process of the diffusion model. The overall workflow of *FastBlend* is illustrated in Figure 1. When applying diffusion models to video stylization, content inconsistencies often arise. For clarity, we denote the original video as the guide video $\{G_i\}_{i=0}^{N-1}$ and the video processed by diffusion models as the style video $\{S_i\}_{i=0}^{N-1}$. As presented in Figure 1, the core concept of *FastBlend* is to blend the aligned content within a sliding window, thereby covering the inconsistent content with content from neighboring frames. In this section, we begin by introducing each step of *FastBlend*, followed by a discussion on the enhancements in efficiency and quality.

3.2 Patch Matching

The first step is to extract the motion feature from the guide video $\{G_i\}_{i=0}^{N-1}$. Given a source frame G_i and a target frame G_j in the guide video, we compute an approximate Nearest Neighbor Field (NNF) $F = \text{NNF}(G_i, G_j)$, which represents

[1] https://github.com/Artiprocher/sd-webui-fastblend

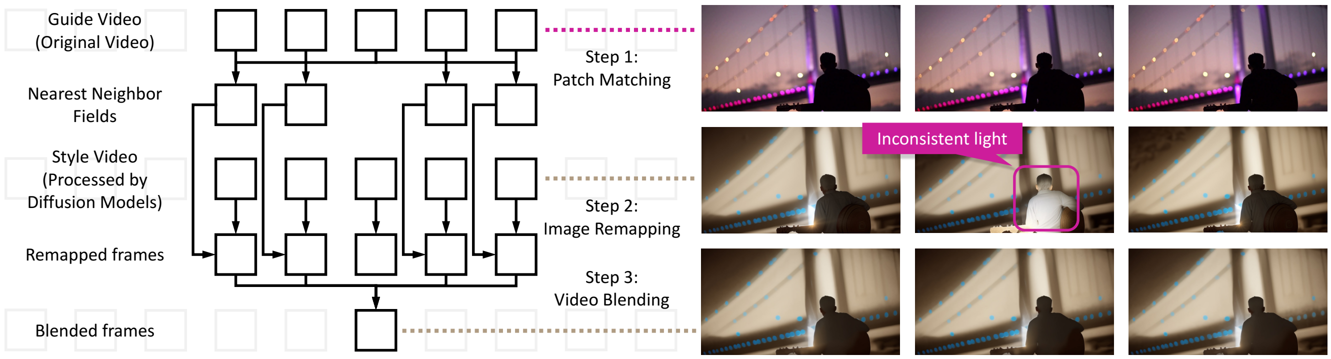


Figure 1: Overall framework of *FastBlend*, where the video is first processed by diffusion models, followed by the application of *FastBlend* to enhance consistency. The prompt for the diffusion model is "A man playing guitar, back, white light, old photo". In this example, the inconsistent lighting in the middle frame is corrected by the neighboring frames. Specifically, *FastBlend* consists of three steps: 1) Estimating the NNFs from the guide video, which serves as a representation of the motion; 2) Remapping the frames in a sliding window to align them with a common frame; 3) Blending the remapped frames to ensure visual consistency.

the matches between the two frames. Unlike optical flow [Teed and Deng, 2020], NNF identifies the best patch-level matches and was originally proposed for tasks like image reconstruction [Guo *et al.*, 2021; Liu *et al.*, 2021]. For convenience, we use $G_i[x,y] \in \mathbb{R}^{(2p+1)\times(2p+1)\times 3}$ to denote the patch centered around the position (x, y), with p being the patch radius. The patch size $2p + 1$ is set to 5 by default. More precisely, $F(x, y) = (x', y')$ indicates that the patch $G_j[x, y]$ matches $G_i[x', y']$. The pseudocode for the base patch matching algorithm is presented in Algorithm 1. We construct an image pyramid and initially estimate the NNF at a low resolution, then upscale it for further refinement. We use a customizable loss function $\mathcal{L}$ to calculate the matching errors. The base loss function is formulated as

$$\mathcal{L}(G_i, G_j, F)_{x,y} = \|G_i[F(x, y)] - G_j[x, y]\|_2^2. \quad (1)$$

We also design several customized loss functions, which are detailed in subsequent subsections. The estimated NNF F is updated iteratively; in each iteration, we scan the updating sequence of F and replace the values that reduce the error. The updating sequence originates from two steps proposed by Barnes et al. [Barnes *et al.*, 2009]:

- **Propagation**: Update matches using adjacent matches. $F'(x, y) = F(x + d_x, y + d_y) - (d_x, d_y)$, where $(d_x, d_y) \in \{(-1, 0), (1, 0), (0, -1), (0, 1)\}$ corresponds to the four cardinal directions.
- **Random search**: Search for better matches in the entire image. $F'(x, y) = F(x, y) + (d_x, d_y)$, where $(d_x, d_y) \sim \mathcal{U}[-r, r]$ and r decreases to zero during the iterative process.

To improve efficiency, we concurrently update each value in F. Additionally, we store images in batches to fully utilize the computing units on GPUs, making our implementation highly parallel.

3.3 Image Remapping

Once we have NNF(G_i, G_j), we can reconstruct the target frame $\hat{G}_j$ using the source frame G_i. Initially, the source frame is converted into $h \times w$ patches, each with a shape of $(2p + 1) \times (2p + 1) \times 3$. Then, the patches are rearranged according to the NNF F. To obtain the reconstructed target frame, we compute the average at the overlapping parts. Note that the VRAM required to store the patches is $(2p+1)^2$ times that of a single image, which poses a challenge for implementation due to storage constraints. To reduce the VRAM requirement and improve I/O efficiency, we directly compute each pixel in the reconstructed image, thus avoiding the storage of intermediate results. The pseudocode for this algorithm is presented in Algorithm 2, effectively reducing the space complexity from $\mathcal{O}(hwp^2)$ to $\mathcal{O}(hw)$. This function is compiled using the NVCC compiler [Grover and Lin, 2012] and runs on NVIDIA GPUs. Similar to Algorithm 1, this algorithm also supports batched data processing.

Algorithm 1 Base Patch Matching

Input: $G_i \in \mathbb{R}^{h\times w\times 3}$: source image; $G_j \in \mathbb{R}^{h\times w\times 3}$: target image; $\mathcal{L}$: customizable loss function; n: number of iterations ($n = 5$ by default)
Randomly initialize $F \in \mathbb{N}^{h\times w\times 2}$
for each pyramid level (h', w') **do**
  Resize images G_i, G_j to $G'_i, G'_j \in \mathbb{R}^{h'\times w'\times 3}$
  Upsample F to $\mathbb{R}^{h'\times w'\times 2}$
  Initialize error matrix $E \leftarrow \mathcal{L}(G'_i, G'_j, F) \in \mathbb{R}^{h\times w}$
  for $i = 1$ **to** n **do**
    for F' in updating sequence of F **do**
      $E' \leftarrow \mathcal{L}(G'_i, G'_j, F')$
      $F(E' < E) \leftarrow F'(E' < E)$
      $E(E' < E) \leftarrow E'(E' < E)$
    end for
  end for
end for
return F

To make the style video $\{S_i\}_{i=0}^{N-1}$ consistent, we remap frames within a sliding window to align with a common frame. Note that the source frames are from the style video rather than the guide video, which can sometimes result in the remapped image appearing fragmented. Inspired by Ebsynth

Algorithm 2 Memory-efficient Image Remapping

Input: $G_i \in \mathbb{R}^{h\times w\times 3}$: source image; $F \in \mathbb{N}^{h\times w\times 2}$: estimated NNF; (x, y): coordinate of the pixel to be computed
$\hat{G}_j(x, y) \leftarrow \mathbf{0} \in \mathbb{R}^3$
for $d_x = -p$ **to** p **do**
  for $d_y = -p$ **to** p **do**
    $(x', y') \leftarrow F(x + d_x, y + d_y) - (d_x, d_y)$
    $\hat{G}_j(x, y) \leftarrow \hat{G}_j(x, y) + G_i(x', y')$
  end for
end for
$\hat{G}_j(x, y) \leftarrow \frac{\hat{G}_j(x,y)}{(2p+1)^2}$
return $\hat{G}_j(x, y)$

[Jamriška *et al.*, 2019], we utilize an improved loss function:

$$\begin{aligned}\mathcal{L}(G_i, G_j, F)_{x,y} =&\alpha\|G_i[F(x, y)] - G_j[x, y]\|_2^2 \\ &+ \|S_i[F(x, y)] - \hat{S}_j[x, y]\|_2^2,\end{aligned} \tag{2}$$

where α is a hyperparameter that determines the extent to which motion information in the input video will be used for remapping. The default value for α is set to 10. $\hat{S}_j$ is the remapped frame and will be updated during the iterations. This loss function can significantly improve the visual quality of remapped frames, as evaluated by prior studies [Jamriška *et al.*, 2019].

3.4 Video Blending

Leveraging the patch matching and image remapping algorithms, we can align the content in different frames. Next, we blend the remapped frames within a sliding window. Specifically, we define the blended frame $\overline{S}_i$ as

$$\overline{S}_i = \frac{1}{2M+1}\sum_{j=i-M}^{i+M}(S_j \to S_i), \tag{3}$$

where $(S_j \to S_i)$ denotes the frame remapped from S_j to S_i using $\text{NNF}(G_j, G_i)$. After remapping and blending, the frames $\{\overline{S}_i\}_{i=0}^{N-1}$ form a consistent video.

In certain application contexts, such as the film industry, the quality of video content is paramount, while in others, computational efficiency is prioritized. To meet diverse requirements, we have developed three distinct inference modes for video blending. The initial mode, without additional adjustments, is denoted as **Balanced Mode**. The other two modes, **Fast Mode** and **Accurate Mode**, are optimized to improve efficiency and video quality, respectively.

3.5 Efficiency Improvement

When implementing the blending algorithm naively, we require $\mathcal{O}(NM)$ NNF estimations, where M represents the size of the sliding window. If M is too large, this naive approach becomes prohibitively slow. To enhance efficiency, we propose a novel data structure called **Remapping Table** for fast remapping and blending. This structure is a tree-like data structure reminiscent of certain tree-like arrays [Fenwick, 1994; De Berg, 2000]. Since the remapped and blended frames can be remapped iteratively, i.e.,

$$(S_i \to S_j) \to S_k \approx S_i \to S_k, \tag{4}$$

$$(S_i + S_j) \to S_k \approx (S_i \to S_k) + (S_j \to S_k). \tag{5}$$

In other words, we can store some intermediate variables to reduce the time complexity. This data structure can compute the estimation of $\overline{S}_i$ with low time complexity. For readers not familiar with tree-like data structures, it is advisable to approach this data structure as a black box, thereby alleviating the need for intimate understanding.

Overall, **Fast Mode** requires $\mathcal{O}(N \log N)$ NNF estimations, which are independent of the sliding window size, thus enabling the use of large windows without a significant increase in computational cost.

3.6 Quality Improvement

When flicker noise in a video is excessively pronounced, simply blending the frames together may result in a smoggy appearance. This issue arises from the inconsistent remapping of content to identical positions across different frames. To mitigate this challenge, we adjust the loss function to more consistently align the contents between frames. Ideally, when different source frames $\{S_i\}_{i=0}^{N-1}$ are remapped to the same target frame S_j, the set $\{S_i \to S_j\}_{i=0}^{N-1}$ should be nearly identical; otherwise, details may be lost during averaging. We first compute the average remapped image: $\overline{S}_j = \frac{1}{N}\sum_{i=0}^{N-1}(S_i \to S_j)$, where $\overline{S}_j$ is updated iteratively based on intermediate variables. We then calculate the distance between $\overline{S}_j$ and each remapped image. The modified alignment loss function is formulated as follows:

$$\begin{aligned}\mathcal{L}(G_i, G_j, F)_{x,y} =&\alpha\|G_i[F(x, y)] - G_j[x, y]\|_2^2 \\ &+ \|S_i[F(x, y)] - \overline{S}_j[x, y]\|_2^2,\end{aligned} \tag{6}$$

where the alignment term $\|S_i[F(x, y)] - \overline{S}_j[x, y]\|_2^2$ signifies the variance of the pixels remapped to position (x, y). By minimizing this term, the variance of $\{S_i \to S_j\}_{i=0}^{N-1}$ is reduced, thus the details are aligned. The **Accurate Mode** necessitates $\mathcal{O}(NM)$ NNF estimations and image remappings. Additionally, the construction of a prefix remapping table is not required. As the frames are processed sequentially, the space complexity is reduced from $\mathcal{O}(N)$ to $\mathcal{O}(M)$. This reduction facilitates the processing of longer videos in **Accurate Mode**.

4 Experiments

To demonstrate the efficacy of *FastBlend*, we conduct evaluations comparing *FastBlend* with other baseline methods in the domain of video stylization. The goal is to transfer the style of a given video according to the provided textual prompt while retaining the structural information of the original content. We first present several video samples to illustrate the differences between the methods and then quantify the performance through quantitive metrics and human evaluation.

4.1 Experimental Settings

Baseline Methods. Considering that *FastBlend* functions as a post-processor, we compare it with both independent video

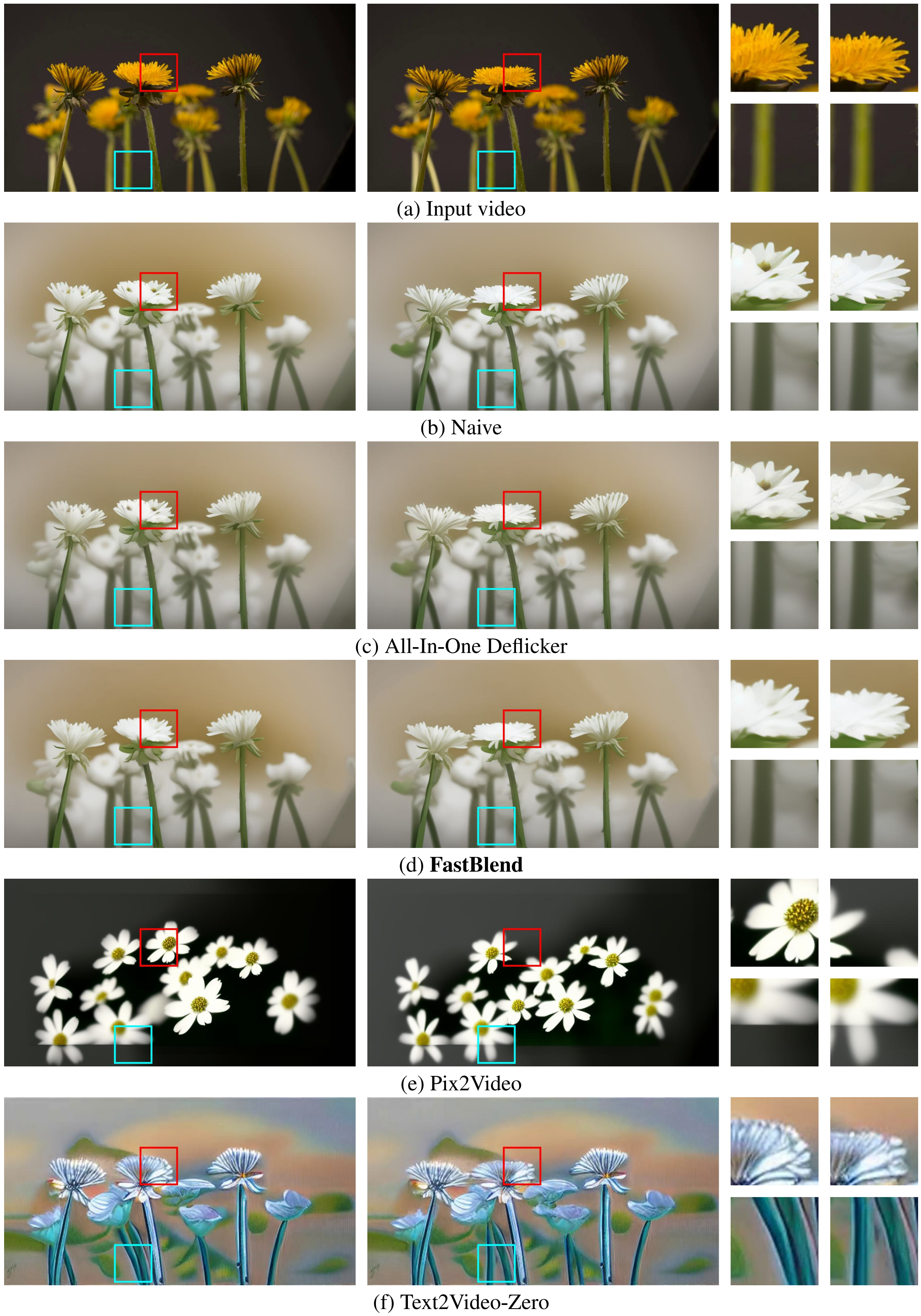


Figure 2: Examples in video stylization. The prompt is “white flowers”. “Naive” denotes directly processing each frame using the diffusion model. All-In-One Deflicker and *FastBlend* are post-processing methods on “Naive”. Pix2Video and Text2Video-Zero are two video stylization methods that modify the generation process of diffusion models.

deflickering methods and video stylization methods that modify the diffusion model. The baseline methods in our experiments include: **All-In-One Deflicker** [Lei *et al.*, 2023]: a state-of-the-art deflickering method that can eliminate flickering artifacts by leveraging a neural atlas in conjunction with a neural filtering strategy. **Pix2Video** [Ceylan *et al.*, 2023]: a video-to-video translation method that uses self-attention feature injection to maintain frame consistency. **Text2Video-Zero** [Khachatryan *et al.*, 2023]: a training-free approach that leverages motion dynamics and cross-frame attention for consistent video processing.

Dataset. For comparative experiments, we created a dataset named Pixabay100. This dataset contains 100 videos collected from Pixabay[2]. We manually crafted prompts for stylization and editing tailored to these videos. Since Pix2Video requires a prompt describing the original video, we also provide descriptions for each video. This dataset will be made publicly available. The resolutions of the videos vary, and we resize each frame to 512×960 for consistency.

Models and Parameters. When reproducing Pix2Video and Text2Video-Zero, we use the default models and settings provided by these baseline methods. In the comparison between *FastBlend* and All-In-One Deflicker, we first process the frames using diffusion models and then apply the two post-processing methods respectively. We utilize a widely acclaimed diffusion model, DreamShaper[3], from open-source communities, to process each video frame by frame. To retain the original video's structural information, we employ two ControlNet [Zhang *et al.*, 2023] models, SoftEdge and Depth. The number of sampling steps is set to 20, the ControlNet scale to 1.0, the classifier-free guidance scale [Ho and Salimans, 2021] to 7.5, and the sampling scheduler to DDIM [Song *et al.*, 2020]. These hyperparameters were tuned empirically. Moreover, we enable cross-frame attention, a strategy widely proven to be effective for consistency [Yang *et al.*, 2023; Duan *et al.*, 2023; Qi *et al.*, 2023; Ceylan *et al.*, 2023; Khachatryan *et al.*, 2023]. *FastBlend*'s efficiency improvement feature is activated for fast processing, with a sliding window size of 30 and a batch size of 64.

4.2 Case Study

An illustrative video is shown in Figure 2, where the prompt is "white flowers" We enlarge some areas to compare the results intuitively. Using the aforementioned diffusion model, we process the video naively, and the processed video is shown in Figure 2(b). The color of the flowers in the processed video successfully changes to white. However, there is a noticeable inconsistency in the flowers. Comparing All-In-One Deflicker with *FastBlend*, we observe that All-In-One Deflicker primarily mitigates slight flickering, as shown in Figure 2(c), whereas *FastBlend* effectively aligns the elements, as indicated by the coherent petals in Figure 2(d). The results from Pix2Video and Text2Video-Zero (Figure 2(e) and Figure 2(f)) also exhibit inconsistent content, where Pix2Video struggles to maintain the structural information from the input video and Text2Video-Zero cannot synthesize stable stems. This case study demonstrates that *FastBlend*, in combination with the diffusion model, can generate coherent and realistic videos in video stylization, significantly enhancing consistency.

[2]https://pixabay.com/videos/

[3]https://civitai.com/models/4384/dreamshaper

Method	Pixel-MSE ↓
Pix2Video	1203.04
Text2Video-Zero	342.85
All-In-One Deflicker	53.38
FastBlend	**35.79**

Table 1: The results of quantitive metrics.

FastBlend is better	Tie	Al-In-One Deflicker is better
75.64%	14.10%	10.26%
FastBlend is better	Tie	Pix2Video is better
91.49%	4.26%	4.26%
FastBlend is better	Tie	Text2Video-Zero is better
89.44%	6.83%	3.73%

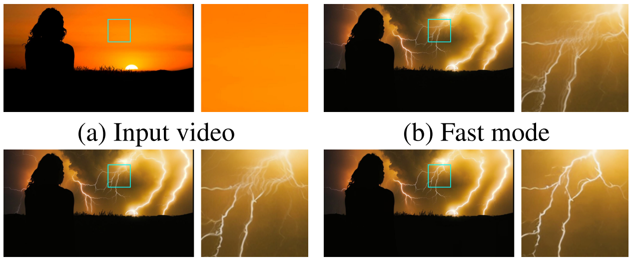
Table 2: The results of human evaluation.

4.3 Quantitative Evaluation

To quantitatively evaluate *FastBlend* and other baseline methods, we calculate the Pixel-MSE [Ceylan *et al.*, 2023] for the videos generated by each method. The Pixel-MSE is the mean square error between the warped frame and its corresponding target frame. The results are presented in Table 1. *FastBlend* achieves the lowest Pixel-MSE, outperforming the other baseline methods by a large margin. Additionally, given that some prior studies [Blattmann *et al.*, 2023b; Ouyang *et al.*, 2023] have questioned the reliability of such evaluation metrics, we further invited 15 participants to partake in a double-blind evaluation. In each round of the evaluation, we randomly chose a video, presenting the participants with videos generated by two different methods. One video was generated by *FastBlend*, and the other by a randomly selected baseline method. The positions of these two videos were also randomized. Participants were asked to choose the video that appeared better in terms of consistency and clarity, or to select "tie" if they could not determine which video was superior. The results of the human evaluation are shown in Table 2, where participants unanimously agreed that *FastBlend*'s overall performance was significantly better than that of the baseline methods.

4.4 Ablation Study

We compared the performance of the different inference modes. Figure 3 illustrates the results for three inference modes. The positions of lightning in the video are random and inconsistent. In the **Balanced Mode** (Figure 3(c)), there is a slight ghosting effect, where the lightning from different frames is blended together. This ghosting issue is more pronounced in the **Fast Mode** (Figure 3(b)). Conversely, in the **Accurate Mode** (Figure 3(d)), the alignment loss function (6) guides the optimization process toward the elimination of unnecessary details, resulting in a clearer representation by merging lightning elements from multiple frames.



(a) Input video (b) Fast mode (c) Balanced mode (d) Accurate mode

Figure 3: Comparison of the three inference modes for blending. The prompt is "a woman, lightning, ball lightning, super power".

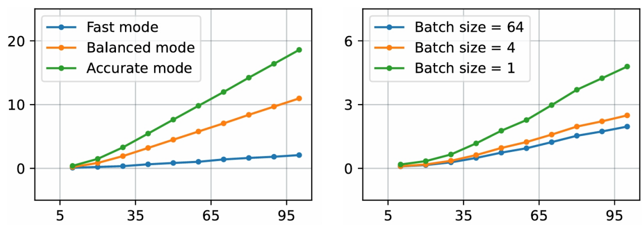


Figure 4: Inference time of different inference modes and batch sizes. The horizontal axis represents the number of video frames, and the vertical axis represents the inference time (in minutes).

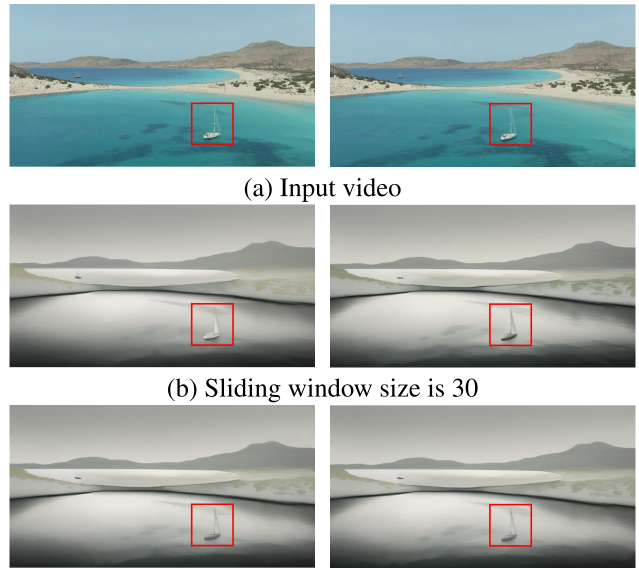
This example demonstrates the efficacy of the quality improvement strategies discussed in **Accurate Mode**.

4.5 Efficiency Analysis

We assessed the computational efficiency of *FastBlend* using an NVIDIA RTX 4090 GPU, recording the time taken to render 100 frames. Given that Pix2Video and Text2Video-Zero are video processing methods tightly integrated with diffusion models, a direct comparison with other methods would not be equitable. When comparing All-In-One Deflicker and *FastBlend*, All-In-One Deflicker required 5.42 minutes, whereas *FastBlend* needed only 2.27 minutes. This demonstrates that *FastBlend* is significantly faster than All-In-One Deflicker. To further understand why *FastBlend* achieves such notable computational efficiency, we followed the settings from the previous experiments and evaluated the efficiency of different inference modes and batch sizes. The computation times are illustrated in Figure 4. As shown in Figure 4(a), the **Fast Mode** is considerably quicker than the other two inference modes. This speed increase is attributable to the data structure in **Fast Mode**, which substantially reduces time complexity. Additionally, as demonstrated in Figure 4(b), a larger batch size contributes to faster processing. By using a large batch size, *FastBlend* can fully utilize the computing resources of the GPU, thereby achieving high efficiency.

4.6 Parameter Sensitivity

We also conducted experiments to investigate the impact of different sliding window sizes. Figure 5 shows the first and last frames of a 125-frame video. When the sliding window size is set to 30 (Figure 5(b)), the color of the boat in the scene is different because the two frames are far apart. As the sliding window covers the entire video (Figure 5(c)), the color of the small boat becomes consistent. Larger sliding window sizes can improve the long-term consistency of videos but require more computation time.



(a) Input video

(b) Sliding window size is 30

(c) Sliding window covers the whole video

Figure 5: Comparison of different sliding window sizes. The prompt in this example is "a white sailboat, a lake, winter".

5 Conclusion and Future Work

In this paper, we propose a model-free video processing approach named *FastBlend*. This approach can significantly enhance video consistency through patch matching and can be seamlessly integrated with diffusion models to design robust video processing pipelines. By aligning and blending content across different frames, *FastBlend* effectively eliminates flickering in videos. To boost efficiency, we introduce a novel data structure that dramatically reduces the time complexity. Furthermore, we have designed an alignment loss function that facilitates content alignment, thereby improving visual quality. The extensive experimental results have underscored the superiority of *FastBlend*.

Looking ahead, we aim to combine *FastBlend* with additional video processing techniques to design more potent video processing pipelines for various application scenarios. Furthermore, we observe that *FastBlend*, as a standalone video processing algorithm, sometimes produces blurred videos when processing videos with large-scale rapid motion. This is another issue that we aim to optimize and improve upon in future work.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under grant Number 62202170 and Alibaba Group through Alibaba Innovation Research Program.

References

[Bar-Tal *et al.*, 2022] Omer Bar-Tal, Dolev Ofri-Amar, Rafail Fridman, Yoni Kasten, and Tali Dekel. Text2live: Text-driven layered image and video editing. In *European conference on computer vision*, pages 707–723. Springer, 2022.

[Barnes *et al.*, 2009] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3):24, 2009.

[Blattmann *et al.*, 2023a] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

[Blattmann *et al.*, 2023b] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22563–22575, 2023.

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

[Ceylan *et al.*, 2023] Duygu Ceylan, Chun-Hao P Huang, and Niloy J Mitra. Pix2video: Video editing using image diffusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 23206–23217, 2023.

[De Berg, 2000] Mark De Berg. *Computational geometry: algorithms and applications*. Springer Science & Business Media, 2000.

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

[Duan *et al.*, 2023] Zhongjie Duan, Lizhou You, Chengyu Wang, Cen Chen, Ziheng Wu, Weining Qian, and Jun Huang. Diffsynth: Latent in-iteration deflickering for realistic video synthesis. *arXiv preprint arXiv:2308.03463*, 2023.

[Esser *et al.*, 2023] Patrick Esser, Johnathan Chiu, Parmida Atighehchian, Jonathan Granskog, and Anastasis Germanidis. Structure and content-guided video synthesis with diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7346–7356, 2023.

[Fenwick, 1994] Peter M Fenwick. A new data structure for cumulative frequency tables. *Software: Practice and experience*, 24(3):327–336, 1994.

[Gal *et al.*, 2022] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit Haim Bermano, Gal Chechik, and Daniel Cohen-or. An image is worth one word: Personalizing text-to-image generation using textual inversion. In *The Eleventh International Conference on Learning Representations*, 2022.

[Goodfellow *et al.*, 2014] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[Grover and Lin, 2012] Vinod Grover and Yuan Lin. Compiling cuda and other languages for gpus. In *GPU Technology Conference (GTC)*, pages 1–59, 2012.

[Guo *et al.*, 2021] Xiefan Guo, Hongyu Yang, and Di Huang. Image inpainting via conditional texture and structure dual generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14134–14143, 2021.

[Guo *et al.*, 2023] Yuwei Guo, Ceyuan Yang, Anyi Rao, Yaohui Wang, Yu Qiao, Dahua Lin, and Bo Dai. Animatediff: Animate your personalized text-to-image diffusion models without specific tuning. *arXiv preprint arXiv:2307.04725*, 2023.

[Hertz *et al.*, 2022] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-or. Prompt-to-prompt image editing with cross-attention control. In *The Eleventh International Conference on Learning Representations*, 2022.

[Ho and Salimans, 2021] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

[Hu *et al.*, 2021] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2021.

[Jamriška *et al.*, 2019] Ondřej Jamriška, Šárka Sochorová, Ondřej Texler, Michal Lukáč, Jakub Fišer, Jingwan Lu, Eli Shechtman, and Daniel Sỳkora. Stylizing video by example. *ACM Transactions on Graphics (TOG)*, 38(4):1–11, 2019.

[Khachatryan *et al.*, 2023] Levon Khachatryan, Andranik Movsisyan, Vahram Tadevosyan, Roberto Henschel, Zhangyang Wang, Shant Navasardyan, and Humphrey Shi. Text2video-zero: Text-to-image diffusion models are zero-shot video generators. *arXiv preprint arXiv:2303.13439*, 2023.

[Lei *et al.*, 2023] Chenyang Lei, Xuanchi Ren, Zhaoxiang Zhang, and Qifeng Chen. Blind video deflickering by neural filtering with a flawed atlas. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10439–10448, 2023.

[Li *et al.*, 2022] Haoying Li, Yifan Yang, Meng Chang, Shiqi Chen, Huajun Feng, Zhihai Xu, Qi Li, and Yueting Chen. Srdiff: Single image super-resolution with diffusion probabilistic models. *Neurocomputing*, 479:47–59, 2022.

[Liu *et al.*, 2021] Steven Liu, Xiuming Zhang, Zhoutong Zhang, Richard Zhang, Jun-Yan Zhu, and Bryan Russell. Editing conditional radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5773–5783, 2021.

[Luebke, 2008] David Luebke. Cuda: Scalable parallel programming for high-performance scientific computing. In *2008 5th IEEE international symposium on biomedical imaging: from nano to macro*, pages 836–838. IEEE, 2008.

[Mou *et al.*, 2023] Chong Mou, Xintao Wang, Liangbin Xie, Jian Zhang, Zhongang Qi, Ying Shan, and Xiaohu Qie. T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models. *arXiv preprint arXiv:2302.08453*, 2023.

[Mount, 2010] David M Mount. Ann: A library for approximate nearest neighbor searching. *http://www. cs. umd. edu/˜ mount/ANN/*, 2010.

[Ouyang *et al.*, 2023] Hao Ouyang, Qiuyu Wang, Yuxi Xiao, Qingyan Bai, Juntao Zhang, Kecheng Zheng, Xiaowei Zhou, Qifeng Chen, and Yujun Shen. Codef: Content deformation fields for temporally consistent video processing. *arXiv preprint arXiv:2308.07926*, 2023.

[Qi *et al.*, 2023] Chenyang Qi, Xiaodong Cun, Yong Zhang, Chenyang Lei, Xintao Wang, Ying Shan, and Qifeng Chen. Fatezero: Fusing attentions for zero-shot text-based video editing. *arXiv preprint arXiv:2303.09535*, 2023.

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

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

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

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

[Schuhmann *et al.*, 2022] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *arXiv preprint arXiv:2210.08402*, 2022.

[Singer *et al.*, 2022] Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, et al. Make-a-video: Text-to-video generation without text-video data. In *The Eleventh International Conference on Learning Representations*, 2022.

[Song *et al.*, 2020] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2020.

[Teed and Deng, 2020] Zachary Teed and Jia Deng. 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*, pages 402–419. Springer, 2020.

[Wang *et al.*, 2021] Xintao Wang, Liangbin Xie, Chao Dong, and Ying Shan. Real-esrgan: Training real-world blind super-resolution with pure synthetic data. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1905–1914, 2021.

[Xing *et al.*, 2023] Zhen Xing, Qijun Feng, Haoran Chen, Qi Dai, Han Hu, Hang Xu, Zuxuan Wu, and Yu-Gang Jiang. A survey on video diffusion models. *arXiv preprint arXiv:2310.10647*, 2023.

[Yang *et al.*, 2022] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications. *ACM Computing Surveys*, 2022.

[Yang *et al.*, 2023] Shuai Yang, Yifan Zhou, Ziwei Liu, and Chen Change Loy. Rerender a video: Zero-shot text-guided video-to-video translation. *arXiv preprint arXiv:2306.07954*, 2023.

[Yu *et al.*, 2021] Sihyun Yu, Jihoon Tack, Sangwoo Mo, Hyunsu Kim, Junho Kim, Jung-Woo Ha, and Jinwoo Shin. Generating videos with dynamics-aware implicit generative adversarial networks. In *International Conference on Learning Representations*, 2021.

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

[Zhou *et al.*, 2022] Shangchen Zhou, Kelvin Chan, Chongyi Li, and Chen Change Loy. Towards robust blind face restoration with codebook lookup transformer. *Advances in Neural Information Processing Systems*, 35:30599–30611, 2022.