Evolvement Constrained Adversarial Learning for Video Style Transfer

Wenbo Li[1*] Longyin Wen[2*] Xiao Bian[3] Siwei Lyu[1]

[1]University at Albany, SUNY [2]JD Finance AI Lab [3]GE Global Research
{wli20,slyu}@albany.edu, lywen.cv.workbox@gmail.com, xiao.bian@ge.com

Abstract. Video style transfer is a useful component for applications such as augmented reality, non-photorealistic rendering, and interactive games. Many existing methods use optical flow to preserve the temporal smoothness of the synthesized video. However, the estimation of optical flow is sensitive to occlusions and rapid motions. Thus, in this work, we introduce a novel evolve-sync loss computed by evolvements to replace optical flow. Using this evolve-sync loss, we build an adversarial learning framework, termed as Video Style Transfer Generative Adversarial Network (VST-GAN), which improves upon the MGAN method for image style transfer for more efficient video style transfer. We perform extensive experimental evaluations of our method and show quantitative and qualitative improvements over the state-of-the-art methods.

1 Introduction

Great artists in history can render scenes with their distinct styles. It is the unique artistic style that differs Van Gogh from Picasso. We wonder if an algorithm can also acquire such styles? For instance, would it be able to re-render the scenes in The Avengers (2012) as if it were the oeuvre of Francis Picabia? Such an interesting question can be formulated as the video style transfer problem as shown in Fig. 1, *i.e.*, given a *style image* (Francis Picabia's Udnie) and a *source video* (a clip from The Avengers), the "synthesizer" should automatically produce a video combining both the style of Udnie and the content of The Avengers. Such an algorithm can find applications in many areas, such as augmented reality, computer games and nonphotorealistic rendering.

Many recent works in the computer vision and computer graphics community have focused on the problem of image style transfer [8, 9, 14, 17, 25]. However, these methods cannot be readily extended to videos, since independently generating each video frame leads to artifacts such as flickering and jagging in the synthesized videos. To this end, existing video style transfer algorithms [1, 3, 24] rely on signals that are estimated by a given motion model such as optical flows computed from adjacent frames to preserve temporal smoothness. We call such signals as model-driven signals. Although more visually pleasing results are achieved with these methods, optical flow estimation methods are known to

* indicates equal contributions.

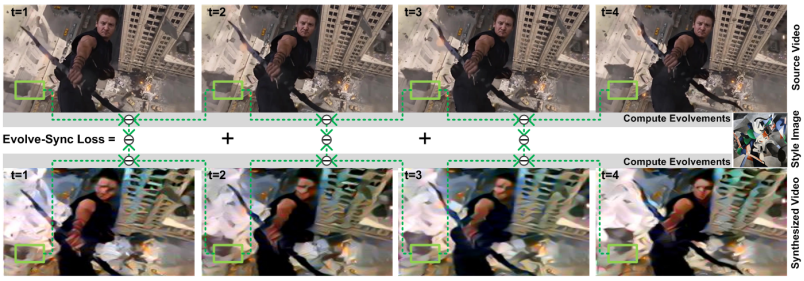

Fig. 1. *Given a video and an image, our algorithm aims to synthesize a video combining the style of the image and the content of the video. To preserve the temporal smoothness of the synthesized video, we use* evolvements *derived from the source and synthesized video, and further compute the evolve-sync loss as the replacement of the optical flow constraints. This loss ensures that the textures at the same location in the image plane of the source and synthesized video evolve synchronously. For the illustration purpose, we only show the order two loss of the evolvements at one patch.*

be sensitive to occlusions and rapid and abrupt motions [23, 27], and such limitations affects the qualities of the synthesized videos. Two recent methods [3, 24] attempt to remedy these problems by introducing occlusion masks to filter out low-confidence optical flow, but the generation of occlusion masks is also error-prone and can lead to further artifacts.

In this work, we aim to exploit model-free signals (against the model-driven ones) in the source video for video style transfer, and synthesize video to match with such signals in the source video to preserve the temporal smoothness. To this end, we introduce **evolvements**, a form of inter-frame variations, as such a model-free signal, the acquisition of which is illustrated in Fig. 1. As the source and synthesized videos are synchronous in time, it is natural to require that the textures in the source and synthesized videos evolve synchronously, which we term as the *evolve-sync assumption*. The evolve-sync assumption is incorporated in our method with the *evolve-sync loss*, which encourages the evolvements from the source domain and those from the synthesized domain to be the same. As we need to preserve the temporal smoothness at both the microscopic and macroscopic levels, we extend the evolve-sync loss to be multi-level by regarding the evolvements as distributions and employing encoders (*e.g.*, a pre-trained CNN) to extract samples from these probability distributions. Thus, the evolve-sync loss encourages samples of different distributions at the corresponding level to be the same. We use the maximum mean discrepancy (MMD) [11] as the distance measure between probability distributions.

The evolve-sync loss can be combined with an image style transfer method to form the basis of video style transfer algorithms. We choose a state-of-the-art image style transfer method, *i.e.*, Markovian Generative Adversarial Network (MGAN) [17], and develop the *Video Style Transfer Generative Adversarial Network* (VST-GAN). MGAN consists of two major components: (i) A Markovian Deconvolutional Adversarial Network (MDAN) denoted by D, and (ii) a gen-

Fig. 2. Saturated vs. Desaturated. *The desaturated color occurs in the synthesized results for the unpaired frames.*

erator that is a feed-forward convolutional neural network and denoted by G. G synthesizes the frames of video according to the content of the source video and the style of the style image, while D plays two roles: it creates real training samples for G with a deconvolutional process driven by the adversarial training, and acts as the adversary to G. Besidess the evolve-sync loss, our modifications to MGAN are presented as follows.

As noted in [17], generating images using the MDAN model D can be slow, and this becomes more problematic for video synthesis. Thus, we design an accelerating training strategy for VST-GAN. Specifically, we only apply D to every other frame to generate real training samples for G, leading to that the real training samples are unpaired with the synthesized frames. We expect that G can synthesize desirable textures for the unpaired frames. However, we observe the desaturated color (similar problem arose in [31]) in the synthesized results of the unpaired frames, which is illustrated in Fig. 2. We therefore modify G by adding a convolutional recurrent layer as its final output layer, which alleviates the desaturation problem, as the recurrent connection makes it possible to propagate the saturation of the paired frames to the unpaired ones.

The main contributions of our work can be summarized as follows: we introduce the evolve-sync loss, which is based on the evolvement that is more reliable than the estimated optical flow in preserving the temporal smoothness of the synthesized video. Applying the evolve-sync loss at both the microscopic and microscopic levels, we develop VST-GAN, an adversarial learning framework for video style transfer, by adapting the MGAN image style transfer method. Specifically, we add a convolutional recurrent layer as the output layer to resolve the desaturation problem in the synthesized video, which is caused by the trade-off between the training speed and the sufficiency of the real samples. Experimental results demonstrate the effectiveness of the evolve-sync loss and VST-GAN.

2 Related Works

Image and Video Style Transfer. There has been an extensive literature on image style transfer methods, which synthesize images based on sampling low-level features in the given source and style images. The extensions to video style transfer [2, 12, 18, 32] rely on optical flow to maintain the temporal smoothness of sampling. See [16] for a comprehensive survey.

Recently, deep neural networks have been proved effective for both image [4, 7–9, 13, 14, 17, 25] and video [1, 3, 24] style transfer. Gatys *et al.* [8, 9] used the

convolutional neural network (CNN) to model the patch statistics with a global Gaussian model of the higher-level feature vectors (*e.g.*, activations of CNN), and transferred the style by minimizing the feature reconstruction loss in an iterative deconvolutional process. Two follow-up works, *i.e.*, Johnson *et al.* [14] and Ulyanov *et al.* [25], proposed fast implementations of Gatys *et al.*'s method. Both methods employed precomputed decoders trained with a perceptual style loss and obtained significant runtime benefits. In contrast to these three works, Li and Wand [17] argued that real-world contextually related patches do not always comply with a Gaussian distribution, but a complex nonlinear manifold, and proposed MGAN, where a feed-forward generator is adversarially learned to project the contextually related patches to the manifold of patches.

Anderson *et al.* [1] extended Gatys *et al.*'s method to video style transfer. To preserve the temporal smoothness of the synthesized video, they used optical flow to initialize the style transfer optimization, and incorporated the flow explicitly into the loss function. To further reduce artifacts at the boundaries and occluded regions, Ruder *et al.* [24] introduced masks to filter out optical flows with low confidences in the loss function. Chen *et al.* [3, 5] extended Johnson *et al.*'s method to a feed-forward network for video style transfer. To preserve the temporal smoothness, this method first obtained the current result via a learned flow, and then reduced the artifacts at the occluded regions by fusing the warped result with the independently synthesized result via a learned occlusion mask. In summary, all existing video style transfer methods rely on using optical to preserve temporal smoothness, and use the occlusion mask to stabilize the results. As such, these methods suffer from the common problems in estimating optical flow, *i.e.*, the sensitivity to occlusion and abrupt motion in video.

Generative Adversarial Network (GAN). GANs [10] have achieved impressive results for various tasks in image processing, such as style transfer [17], generation [6], editing [33], representation learning [21], and translation [34], *etc*. The key to GANs' success is the idea of an adversarial loss that forces the generated images/videos to be indistinguishable from the real ones. Only a few works develop GANs for videos, *i.e.*, generation [29, 28, 30] and prediction [20, 26]. The real samples of the existing GANs for video generation and prediction are available. However, this is not the case for video style transfer. This is because the qualified real samples for this task should contain both the desirable style and the required content. In VST-GAN, we generate such samples with the deconvolutional model in MGAN constrained by the evolve-sync loss. However, the iterative deconvolutional optimization for videos is slow. Thus, we design a strategy to accelerate the training process of the GAN framework for video style transfer while maintaining the quality of the synthesized videos.

3 Overview

We first formally define the video style transfer problem as following: given a source video $\mathcal{X} = \{X_1, \cdots, X_i, \cdots\}$ and a style image S, we aim to produce a video $\mathcal{Y} = \{Y_1, \cdots, Y_i, \cdots\}$ with the style of S, and the content of $\mathcal{X}$.

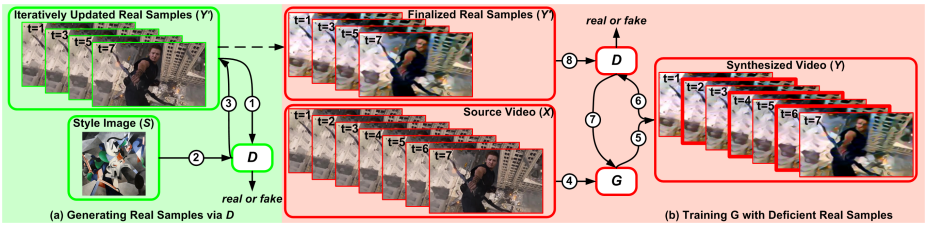

Fig. 3. Pipeline. *Our method first uses a MDAN model D to generate real samples every other frame within an iterative deconvolutional process. $\mathcal{Y}'$ is initialized with the downsampled source video $\mathcal{X}'$. The generated real samples are used to train G using a GAN model. Then, G transfers the image style to the whole video. The numbers on the arrows indicate the order of operations, which are explained in detail in § 3.*

We design VST-GAN, an adversarial learning framework based on MGAN [17], to build a video style transfer algorithm incorporating the evolve-sync loss, with an aim to preserve temporal smoothness of frames without using optical flow. VST-GAN consists of a deconvolutional model D, and a feed-forward generator G. Both D and G are integrated with the evolve-sync loss. Fig. 3 illustrates the overall pipeline of our method, which includes two steps:

Step (i) D generates real samples for G within a deconvolutional process that is constrained by the evolve-sync loss and driven by the adversarial training. Considering the efficiency issue, we accelerate the generation process by applying it to every other frame. In Fig. 3 (a), steps ① and ② correspond to the convolutional forward pass, where D determines how real $\mathcal{Y}'$ is. Step ③ represents the deconvolutional backward pass, where D acts as the generator and the losses are back-propagated to pixels of $\mathcal{Y}'$.

Step (ii) Given the unpaired training samples, G is trained to transfer the style of S to the generated video, using D as the adversary in the manner of GAN. However, the original generator in MGAN suffers from the lack of real samples, which can cause the desaturation effect (or grey image tone) in the generated videos (see an example in Fig. 2). We therefore modify G by adding a convolutional recurrent layer as its final output layer, which reduces the desaturation issue effectively. In Fig. 3 (b), in the runtime of updating G, $\mathcal{X}$ is fed into G (step ④) to generate $\mathcal{Y}$ (step ⑤). Then, D determines how real is the synthesized video $\mathcal{Y}$ (step ⑥), and the losses are back-propagated to update G (step ⑦). During the updating of D, $\mathcal{Y}$ and $\mathcal{Y}'$ are used as real and fake samples[1] to train D (steps ⑥ and ⑧), respectively.

The evolve-sync loss is based on a more reliable signal than the optical flow estimated from the input video, thus it can better preserve the temporal smoothness. Our accelerating training strategy and the added convolutional recurrent structure effectively reduce the training complexity of VST-GAN.

[1] The naming fashion of real and fake samples follow the convention of GAN: the output $\mathcal{Y}$ of G is considered to be fake, while the precomputed $\mathcal{Y}'$ is real.

4 The Evolve-Sync Loss

One basic requirement of video style transfer is to preserve the temporal smoothness between generated frames, as human visual systems are sensitive to the flickering artifacts. This means that the simple approach of generating each frame independently using existing image style transfer algorithms is not effective, as it will lead to visually displeasing results due to two factors. First, as many image style transfer methods (*e.g.*, [8, 9]) are iterative, their results are affected by different initializations and the local minima of the style loss function. Second, a small perturbation in the source images may cause large variations in the synthesized results that are not temporally smooth.

As such, in order to generate temporally smooth frames with spatially rich style patterns, existing methods [1, 3, 24] modify the image style transfer algorithms by incorporating optical flows estimated from the source video as supervisory signals. The reliability of the estimated optical flow is often problematic due to the problems related with the common optical flow algorithms, *i.e.*, sensitivity to occlusions and rapid motions. This motivates us to turn to a different source of model-free signal directly acquired from the source video itself to capture inter-frame variations, which we term as *evolvement*. Given two frames X_i and $X_{i-k} \in \mathbb{R}^{h\times w\times 3}$, we define the evolvement from X_{i-k} to X_i as a distribution $\mathcal{E}(X_{i-k}, X_i)$. Fig. 4 illustrates the sampling process from evolvements. We compute an evolvement sample $\mathcal{E}(X_{i-k}, X_i)_m \sim \mathcal{E}(X_{i-k}, X_i)$ as:

$$\mathcal{E}(X_{i-k}, X_i)_m = z(|g(X_i)_m - g(X_{i-k})_m|), \tag{1}$$

where $g(\cdot)$ denotes an encoder function that extracts samples from evolvements. The standardization function is represented as $z(x) = \frac{x-\mu}{\sigma}$, where the input x is a 2D matrix, and μ and σ are the mean and standard deviation of elements in x, respectively. Index m indicates the mth sample generated by $g(\cdot)$.

Our method is based on the *evolve-sync assumption*, which states that $\mathcal{X}$ and the synthesized video $\mathcal{Y}$ are synchronous in time, so their evolvements, $\mathcal{E}(X_{i-k}, X_i)$ and $\mathcal{E}(Y_{i-k}, Y_i)$, can be viewed as two synchronized signals. As seen in Fig. 4, the brighter a pixel in an evolvement sample is, the more drastic variation occurs at that pixel. The rationality behind the evolve-sync assumption can be understood by contradiction: if it does not hold for a certain pixel, it means that the drastic variation occurs in $\mathcal{E}(X_{i-k}, X_i)$ while the mild variation occurs in $\mathcal{E}(Y_{i-k}, Y_i)$, or vice versa. This suggests that the content at that location has not been properly preserved, which contradicts the problem formulation in § 3.

Given $\mathcal{X}$ with a certain temporal smoothness degree, preserving the evolve-sync is equivalent to forcing the temporal smoothness of $\mathcal{Y}$ to be the same as that of $\mathcal{X}$. Consequently, we introduce the *evolve-sync loss* L_{es} to enforce the evolve-sync assumption in $\mathcal{Y}$ that measures the distance between $\mathcal{E}(X_{i-k}, X_i)$ and $\mathcal{E}(Y_{i-k}, Y_i)$. To this end, we employ the Maximum Mean Discrepancy [11] as the metric between two probability distributions:

$$L_{es}(\mathcal{F}, \mathcal{X}, \mathcal{Y}) = \sum_{|i-j|<\delta} \sup_{f\in\mathcal{F}} (\mathbf{E}_{x\sim\mathcal{E}(X_i,X_j)}[f(x)] - \mathbf{E}_{y\sim\mathcal{E}(Y_i,Y_j)}[f(y)]), \tag{2}$$

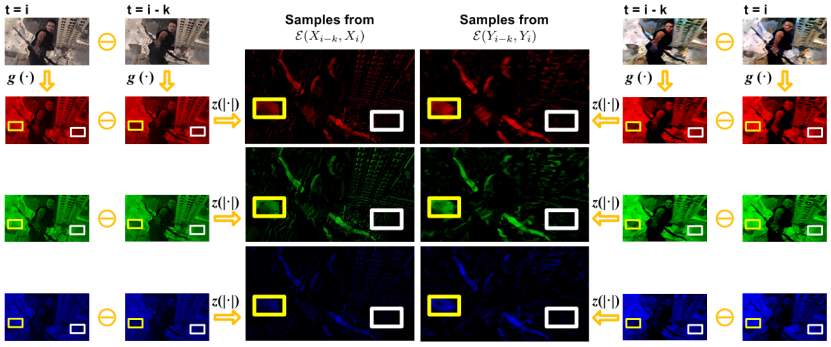

Fig. 4. Illustration of the computation of evolvement and evolve-sync assumption. *$g(\cdot)$ represents an encoder, which splits the image in R, G and B color channels herein. $z(\cdot)$ represents a standardization function meaning subtracting mean and dividing by standard deviation. $\mathcal{E}(X_{i-k}, X_i)$ represents the evolvement from frame X_{i-k} to X_i. The yellow/white boxes highlight a spot in the image plane where drastic/mild variations occur.*

where δ is a preset parameter determining the order of L_{es}, $\mathcal{F}$ is a Gaussian kernel and we set to $\delta = 3$ in our experiments.

We aim to preserve the temporal smoothness of $\mathcal{Y}$ at the microscopic level where the synthesized textures are temporally continuous, and at the macroscopic level where the synthesized textures and the video content are synchronized. To this end, we use two encoders for each level (i) the microscopic encoder $g_1(\cdot)$, which splits the image in R, G and B color channels for the microscopic level, and (ii) the macroscopic encoder $g_2(\cdot)$, which is a pretrained VGG network (sampled from *Relu3_1*). As such, the overall evolve-sync loss is given as:

$$L_{es}(\mathcal{G}, \mathcal{F}, \mathcal{X}, \mathcal{Y}) = \sum_{r=1}^{|\mathcal{G}|} \alpha_r \cdot \sum_{|i-j|<\delta} \sup_{f \in \mathcal{F}} (\mathbf{E}_{x_r \sim \mathcal{E}(X_i, X_j)}[f(x)] - \mathbf{E}_{y_r \sim \mathcal{E}(Y_i, Y_j)}[f(y)]), \quad (3)$$

where x_r and y_r are determined by $g_r(\cdot)$, and we set $\alpha_1 = 0.005$ and $\alpha_2 = 100$.

5 Video Style Transfer GAN (VST-GAN)

In this section, we describe the architecture of VST-GAN in § 5.1 and the training of VST-GAN in § 5.2.

5.1 Architecture

We build VST-GAN by adapting Markovian GAN (MGAN) [17], a state-of-the-art image style transfer framework that does not rely on the implicit assumption that the real-world textures comply with a Gaussian distribution. We show the architecture of VST-GAN in Fig. 5, where grey blocks indicate the intrinsic

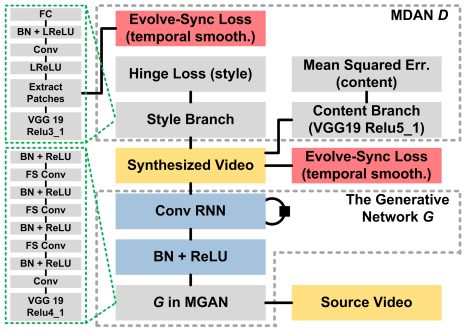

Fig. 5. Architecture of VST-GAN. *Grey blocks indicate the intrinsic architecture of MGAN, and blocks with other colors indicate the input and our modifications.*

architecture of MGAN, and blocks with other colors indicate our modifications. MGAN consists of two major components: (i) A Markovian Deconvolutional Adversarial Network (MDAN) denoted by D, and (ii) a feed-forward generator denoted by G. D plays two roles: it creates real training samples for G with a deconvolutional process that is driven by the adversarial training, and acts as the adversary to G.

MDAN D. As shown in Fig. 5, D has the *style branch* and the *content branch*. The style branch learns to distinguish the feature patches extracted from the feature maps output by VGG19 *Relu3_1* of the source video from those of the synthesized videos. D outputs a classification score $s = 1$ or 0 for each patch, indicating how "real" the patch is (with $s = 1$ being sampled from the style image S, or real patch). For each patch sampled from the synthesized frame, we minimize its style loss (*i.e.*, $1 - s$). Like Radford *et al.*[22], we use batch normalization (BN) and leaky ReLU (LReLU) to improve the training of D. The content branch encourages the content of the synthesized image to be similar to that of the source image, and is constructed from VGG19 features on the same image from higher and more abstract layer *Relu5_1*. The content dissimilarity is measured by a content loss given by the mean squared error between two feature maps obtained from the source video and the synthesized one, respectively. When using D to generate real samples for G, the deconvolution process back-propagates both the style and content loss to pixels. When D acts as the adversary to G, the style and content loss are back-propagated to train G.

The Generative Network G. D requires many iterations and a separate run for each source image, so Li and Wand [17] further developed G, which consists of a pre-trained VGG encoder and a decoder. The VGG encoder of G takes the source image as input, and outputs a feature map from *Relu4_1*. The decoder of G takes the output of the encoder, and decodes an image through a ordinary convolution followed by a cascade of fractional-strided convolutions (FS Conv in Fig. 5). Note that the content loss is used to measure the content dissimilarity between the synthesized image and its corresponding real sample. Although

being trained with fixed-size input, G can be naturally extended to images of arbitrary sizes. VGG encoders in MGAN are fixed during training.

When adapting MGAN to the video style transfer, we make two major modifications to its architecture. First, in order to preserve the temporal smoothness at both the microscopic and macroscopic level, we integrate D with the proposed evolve-sync loss at two levels, *i.e.*, the synthesized video (microscopic) and the VGG encoder of the style branch (macroscopic).

Unfortunately, D has a slow running time – it takes D nearly 4 hours to synthesize a 50-frame video on a single Titan X GPU – which is problematic to generate videos with more frames. Thus, we only apply D every other frame to generate real samples for G, and train G with such unpaired samples. This way, there will be a half of frames without the corresponding real samples, so these frames will not be used to compute the content loss (inherited from standard regression problems). Since such a content loss encourages conservative predictions, it makes G generate synthesized frames with desaturation artifacts (Fig. 2). To alleviate the desaturation problem, we further modify MGAN by adding a convolutional recurrent layer as the final output layer of G, as the recurrent connection makes it possible to smooth the saturation of consecutive frames.

5.2 Training

The training process of VST-GAN includes two steps: (i) generating real training samples for G using D on every other frame, and (ii) training G adversarilly against D with the unpaired training samples.

Generate Real Samples via D. In order to train G adversarially, we need qualified real samples that contain both the style of S and the content of $\mathcal{X}$. As such real samples are not accessible to us during training, we generate them using D on every other frame of the videos as described in § 3. We denote the downsampled source video as $\mathcal{X}' = \{X_1, X_3, \cdots, X_{i-2}, X_i, \cdots\}$ and denote its corresponding real samples as $\mathcal{Y}' = \{Y'_1, Y'_3, \cdots, Y'_{i-2}, Y'_i, \cdots\}$. Then, we perform deconvolution with D iteratively to update $\mathcal{Y}'$ (initialized with $\mathcal{X}'$), so that the following loss is minimized:

$$\hat{\mathcal{Y}}' = \underset{\mathcal{Y}'}{\operatorname{argmin}} \sum_{Y'_i \in \mathcal{Y}'} [L_t(\Phi_t(Y'_i), \ell_{real}) + L_c(\Phi_c(X_i), \Phi_c(Y'_i)) + \omega \Upsilon(Y'_i)] + L_{es}(\mathcal{G}, \mathcal{F}, \mathcal{X}', \mathcal{Y}'), \tag{4}$$

where L_t denotes the style loss. L_c denotes the content loss, which is a mean squared error. Φ_t and Φ_c denote the VGG encoder in the style and content branch, respectively. L_{es} denotes the evolve-sync loss defined in (3). The regularizer Υ is a smoothness prior for pixels [19]. We sample patches from $\Phi_t(Y'_i)$, and compute L_t as the hinge loss with their labels fixed to one, *i.e.*, $\ell_{real} = 1$:

$$L_t(\Phi_t(Y'_i), \ell_{real}) = \frac{1}{N} \sum_{j=1}^{N} \max(0, 1 - \ell_{real} \cdot s_j), \tag{5}$$

where s_j denotes the score (output by D) of the jth patch, and N is the total number of sampled patches in $\Phi_t(Y_i')$.

The model D is trained in tandem: its parameters are randomly initialized, and then updated after each deconvolution, so it improves as $\mathcal{Y}'$ improves. The objective of updating D is:

$$\hat{D} = \underset{D}{\operatorname{argmin}} L_t(\Phi_t(S), \ell_{real}) + \sum_{Y_i' \in \mathcal{Y}'} L_t(\Phi_t(Y_i'), \ell_{fake}). \tag{6}$$

$\ell_{real} = 1$ and $\ell_{fake} = 0$. Like [17], we set $\omega = 0.00001$ in (4), and minimize (4) and (6) using back-propagation with ADAM [15] (learning rate 0.02, momentum 0.5). The optimization in (6) is memory intensive. To make it feasible and efficient for a machine with a Titan X GPU with 12GB onboard memory, we divide X' into multiple non-overlapped segments of 3 frames, and synthesize frames within one segment after another. In this way, L_{es} in (4) will only preserve the temporal smoothness within each segment. In order to preserve the inter-segment smoothness, we use the last 2 frames of the previous segment to compute L_{es}, and leave these 2 frames unchanged during the optimization for the current segment. The segment size can be adaptively enlarged with increased GPU memory capacity.

Train G Against D with Unpaired Real Samples. Given the unpaired real samples $\mathcal{Y}'$, we aim to train G against D in a GAN model. G takes $\mathcal{X}$ as input and outputs the synthesized video $\mathcal{Y} = \{ Y_1, \cdots, Y_i, \cdots \}$, with $Y_i = G(X_i)$. Thus, our objective herein is as follows:

$$\begin{aligned} L(G, D, \mathcal{X}, \mathcal{Y}, \mathcal{Y}') = & \sum_{Y_i \in \mathcal{Y}} [L_t(\Phi_t(Y_i), \ell_{real}) + \omega \Upsilon(Y_i)] + L_{es}(\mathcal{G}, \mathcal{F}, \mathcal{X}, \mathcal{Y}) + \\ & \sum_{Y_i' \in \mathcal{Y}'} [L_t(\Phi_t(Y_i'), \ell_{real}) + L_c(\Phi_c(Y_i), \Phi_c(Y_i'))]. \end{aligned} \tag{7}$$

We therefore aim to solve:

$$\hat{G} = \arg \min_G \max_D L(G, D, \mathcal{X}, \mathcal{Y}, \mathcal{Y}'), \tag{8}$$

where D and G are trained from scratch using back-propagation with ADAM (learning rate 0.02, momentum 0.5). Same notations as those in (7) can be found in (4), (5) and (6). Note that L_c is only valid for the paired frames.

6 Experiments

Implementation Details. We implement VST-GAN and MGAN using Tensorflow, and conduct the experiments on a computer with an Intel Xeon X5570 CPU with 16 cores of 2.93GHz each, 94.4GB memory, and one NVIDIA TITAN X GPU with 12GB onboard memory. For the real sample generation process, D is trained for each segment (3 frames) for $3,000$ iterations. With a batch size of 3, G is trained for $20,000$ iterations. For a 50-frame video, it takes D approximately 2 hours to generate the real samples, and approximately a further 1 hour to train G.

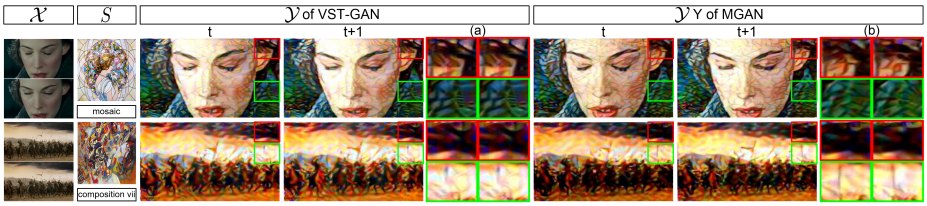

Fig. 6. Qualitative comparison with MGAN [17] *The marked regions highlight that the temporal smoothness of our results is higher. The dilated marked regions are shown in (a) and (b). This figure is best viewed in color.*

Datasets. We use 8 classical style images, *i.e.*, *starry night*, *the scream*, *udnie*, *la muse*, *wave*, *composition vii*, *mosaic*, and *candy*, several of which are used in [3] or [24]. For the source videos, we choose 8 videos with diverse contents, including natural scenes, action scenes, close-up portraits, *etc*. Lengths of these videos vary from 40 to 300 frames, with 91 frames on average. All videos have the image resolutions of 640×360 and were captured at 23 frame per second.
Compared Methods. We compare VST-GAN with ASTV [24], a state-of-the-art neural network based video style transfer method. ASTV uses optical flow and occlusion mask to preserve temporal smoothness in the synthesized video, so it suffers from the common problems in estimating optical flow, *i.e.*, the sensitivity to occlusion and abrupt motion in video. We also create a baseline method based on MGAN [17], which uses image style transfer method in [17] to create individual frames independently. Comparison with these baseline methods demonstrates the advantage of our method in preserving temporal smoothness.

6.1 Qualitative Comparison

In Fig. 6, we show two consecutive frames from two synthesized videos produced by VST-GAN and MGAN, respectively, with two highlighted regions in each frame. The close-up regions demonstrate the effectiveness of evolve-sync loss in preserving the temporal smoothness. As mentioned in the beginning of § 4, the image style transfer methods (*e.g.*, MGAN) are ineffective in preserving the temporal smoothness, which is evident from comparing the two close-up regions.

We further compare the synthesis results using VST-GAN with ASTV [24] in Fig. 7, with four video clips with a variety of challenging factors including camera motions, rapid object motions, and occlusions, *etc*. Fig. 7 (a) shows an epic natural scene with lateral camera motions. Note that artifacts emerge as the new content enters the scene at the top right corner. Considering the contextual information on the left side, the sky color within the marked region is supposed to be either blue, white or yellow. However, the actual color is brownish-grey, which is partially caused by optical flow's intrinsic limitation. Specifically, since the camera view moves from left to right, the estimated direction of the optical flow is the opposite. This leads to an ill-posed problem that the new content at the top right corner solely depends on the pixel values along image's right

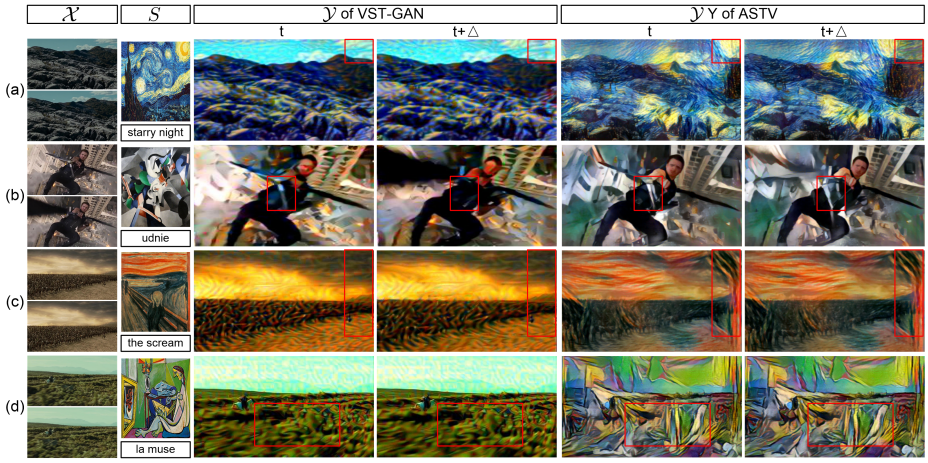

Fig. 7. Qualitative comparison of the results of VST-GAN with those of ASTV [24] *Row (a) corresponds to a scene with the rapid camera motion, and we highlight the newly entering regions where the artifacts appear for ASTV's result. Row (b) displays a video associated with the rapid object motions and occlusions. The marked regions show that the artifacts appear after rapid motions and occlusions for ASTV's result. Row (c) presents an example where the ghosting salient content exists in ASTV's result but not in $\mathcal{X}$. Row (d) illustrates an example where the content of $\mathcal{X}$ is not preserved properly by ASTV. This figure is best viewed in color.*

border. Thus, colors for the sky region (blue, white and yellow) on the left side have no effect on the synthesized video. Fig. 7 (b) displays a video in which the arrow moves rapidly and its movement causes the occlusion. Artifacts arise in the videos synthesized by ASTV due to the instability of the estimated optical flow in the presence of rapid motions and occlusions. On the other hand, because we introduce the evolve-sync loss to replace optical flow, VST-GAN is not affected by problems of the optical flow estimation.

In Fig. 7 (c), we present an example where ASTV synthesized video contains some "ghost" salient content that does not exist in the source video. The preservation of the source video content is worse in the ASTV's result in Fig. 7 (d), *e.g.*, the contextually related patches that constitute the grassland in $\mathcal{X}$ become unrelated in $\mathcal{Y}$. This is because ASTV [24] models textures in the style image with a Gaussian distribution. As a result, the synthesized video does not further improve once two distribution matches and the synthesis quality of local image regions cannot be guaranteed. In contrast, VST-GAN preserves the content more properly by relaxing the above assumption to that the textures follow a complicated non-linear manifold. Furthermore, the adversarial training of VST-GAN can recognize such a manifold with its discriminative network, and strengthen its generative power with a projection on the manifold.

6.2 Quantitative Comparison

Evaluation Metrics. Existing methods [3, 24] measure the temporal smoothness of the synthesized video using the ground-truth optical flow and occlusion mask. Specifically, they warp the i-th frame in the synthesized video to be synchronized with the ground truth flow and compute the difference with the $(i-1)$-th synthesized frame in non-occluded regions. Although this metric is straightforward to compute, it has two drawbacks. First, it restricts the choice of the evaluated videos to those with ground truth optical flow, which are very

Sequence	Method	AESL					
		2-order	4-order	6-order	8-order	10-order	12-order
starry night	ASTV [24]	**45.22**	**132.74**	221.95	301.32	366.93	463.47
	MGAN [17]	97.67	245.43	344.66	443.56	523.67	598.88
	VST-GAN (ours)	60.42	143.62	**220.32**	**289.87**	**344.98**	**412.83**
	VST-GAN w/o ESL	72.13	181.80	274.85	379.43	440.05	533.49
	VST-GAN w/o RNN	65.36	175.23	271.84	347.59	375.17	422.74
the scream	ASTV [24]	32.45	105.31	178.48	237.00	308.06	368.63
	MGAN [17]	91.83	220.06	292.33	362.17	420.37	478.57
	VST-GAN (ours)	**31.33**	**96.74**	**139.05**	**179.07**	**204.43**	**240.20**
	VST-GAN w/o ESL	65.65	143.11	197.96	249.81	293.99	348.53
	VST-GAN w/o RNN	47.62	108.26	146.45	193.29	221.19	252.69
udnie	ASTV [24]	**48.38**	84.83	120.23	154.45	187.89	220.71
	MGAN [17]	81.14	121.00	149.36	173.43	226.53	252.21
	VST-GAN (ours)	48.69	**72.36**	**93.31**	**112.74**	**131.06**	**148.46**
	VST-GAN w/o ESL	81.00	116.75	148.03	177.20	204.52	230.32
	VST-GAN w/o RNN	54.60	89.00	119.84	148.73	176.09	202.08
la muse	ASTV [24]	84.18	266.25	459.09	650.63	838.49	1021.58
	MGAN [17]	193.32	493.94	737.08	940.96	1120.69	1282.29
	VST-GAN (ours)	**79.40**	**234.34**	**322.61**	**396.55**	**480.32**	**541.78**
	VST-GAN w/o ESL	131.70	309.85	460.34	595.70	721.95	841.39
	VST-GAN w/o RNN	96.43	247.61	325.74	413.65	493.38	568.11
wave	ASTV [24]	59.97	147.53	233.54	321.63	413.23	506.86
	MGAN [17]	180.76	434.75	636.53	813.08	978.61	1134.02
	VST-GAN (ours)	**58.21**	**138.57**	**193.25**	**254.14**	**306.07**	**361.79**
	VST-GAN w/o ESL	130.35	314.50	466.62	603.03	732.35	855.11
	VST-GAN w/o RNN	123.84	294.62	429.18	546.39	656.17	759.13
comp. vii	ASTV [24]	**32.92**	**99.67**	172.53	234.40	305.62	345.03
	MGAN [17]	94.61	222.30	304.92	357.22	417.04	456.96
	VST-GAN (ours)	39.23	104.42	**157.65**	**208.61**	**254.18**	**293.65**
	VST-GAN w/o ESL	92.03	245.12	373.81	470.22	578.90	642.62
	VST-GAN w/o RNN	65.00	159.93	239.70	303.14	354.22	413.45
mosaic	ASTV [24]	**30.44**	103.48	187.32	275.91	364.91	451.67
	MGAN [17]	99.35	260.26	400.07	528.03	645.29	751.34
	VST-GAN (ours)	36.84	**98.88**	**157.79**	**214.84**	**253.38**	**301.66**
	VST-GAN w/o ESL	63.13	179.14	289.34	395.64	497.12	591.47
	VST-GAN w/o RNN	53.84	144.03	225.46	301.84	374.40	442.21
candy	ASTV [24]	29.98	90.81	148.01	200.58	249.69	296.37
	MGAN [17]	33.88	84.39	126.82	164.90	200.51	233.95
	VST-GAN (ours)	**19.26**	**52.34**	**87.74**	**110.95**	**144.60**	**170.47**
	VST-GAN w/o ESL	31.07	76.31	114.84	149.95	182.66	213.52
	VST-GAN w/o RNN	23.67	67.95	107.51	143.22	175.31	203.78

Table 1. *Comparison on temporal smoothness for synthetic videos using ours and state-of-the-art video style transfer methods.*

difficult to generate and scarce in number. Second, it does not allow for the evaluation of long-term temporal smoothness due to the lack of long-term ground

truth optical flow. To this end, we use the *averaging evolve-sync loss* (AESL) (averaged by the video length) as a new metric that is free of the optical flow, occlusion mask and the short-term restrictions. We compute the multi-order AESL to evaluate the temporal smoothness for short (order 2 and 4)/medium (order 6 and 8)/long-term (order 10 and 12).

Comparing with the State-of-the-art Methods. The comparison results are presented in Table 1. These result show that VST-GAN outperforms MGAN significantly in terms of the temporal smoothness of the synthesized videos. The comparison between VST-GAN and ASTV based on AESL of order 2 and 4 suggests comparable performance of our method using the evolve-sync loss to those based on optical flow and occlusion mask in preserving the short-term temporal smoothness. In addition, the evolve-sync loss is more effective than optical flow in preserving medium/long-term temporal smoothness, which is demonstrated by the comparison based on AESL of order 6, 8, 10 and 12. This is due the lack of long-term optical flows, which cannot be reliably estimated using current methods. In contrast, the high-order evolve-sync loss can be more easily computed and compared.

Effects of the Evolve-sync Loss. To investigate the impact of the evolve-sync loss on preserving the temporal smoothness, we remove it from the objective of training G (7). As a result, we observe significant increase in AESL of all orders, which indicates the retrogression on the preservation of the temporal smoothness. Nonetheless, VST-GAN still preserves the temporal smoothness better than MGAN even without the use of the evolve-sync loss. This is because we maintain the evolve-sync loss in D for generating real training samples, which further demonstrates the effectiveness of the evolve-sync loss.

Effects of the Recurrent Structure. We remove the convolutional recurrent layer from VST-GAN to study its impact on preserving the temporal smoothness. Consequently, the AESL increases slightly, but the increment is much smaller compared to that after removing the evolve-sync loss. This indicates that the recurrent structure is also useful for preserving the temporal smoothness, but its impact is less prominent than that of the evolve-sync loss.

Runtime Efficiency. The runtime speed of our VST-GAN in synthesizing videos is 18.18 fps, which is comparable to the image style transfer method MGAN (19.33 fps), and much efficient than the deconvolutional video style transfer method ASTV (0.03 fps).

7 Conclusion

In this work, we propose VST-GAN as an adversarial learning framework for video style transfer based on the evolve-sync loss. We show that the evolve-sync loss is able to preserve the temporal smoothness effectively without using optical flow. Our accelerating training strategy and the convolutional recurrent structure significantly reduce the training complexity of VST-GAN. Experimental evaluations show that VST-GAN outperforms the state-of-the-art methods based on optical flow in both running time efficiency and visual quality.

References

1. Anderson, A.G., Berg, C.P., Mossing, D.P., Olshausen, B.A.: Deepmovie: Using optical flow and deep neural networks to stylize movies. CoRR **abs/1605.08153** (2016)
2. Bousseau, A., Neyret, F., Thollot, J., Salesin, D.: Video watercolorization using bidirectional texture advection. TOG **26**(3), 104 (2007)
3. Chen, D., Liao, J., Yuan, L., Yu, N., Hua, G.: Coherent online video style transfer. In: ICCV (2017)
4. Chen, D., Yuan, L., Liao, J., Yu, N., Hua, G.: Stylebank: An explicit representation for neural image style transfer. In: CVPR. pp. 1897–1906 (2017)
5. Chen, D., Yuan, L., Liao, J., Yu, N., Hua, G.: Stereoscopic neural style transfer. In: CVPR. pp. 1–9 (2018)
6. Denton, E.L., Chintala, S., Szlam, A., Fergus, R.: Deep generative image models using a laplacian pyramid of adversarial networks. In: NIPS. pp. 1486–1494 (2015)
7. Fan, Q., Chen, D., Yuan, L., Hua, G., Yu, N., Chen, B.: Decouple learning for parameterized image operators. In: ECCV. pp. 455–471 (2018)
8. Gatys, L.A., Ecker, A.S., Bethge, M.: Texture synthesis using convolutional neural networks. In: NIPS. pp. 262–270 (2015)
9. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: CVPR. pp. 2414–2423 (2016)
10. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A.C., Bengio, Y.: Generative adversarial nets. In: NIPS. pp. 2672–2680 (2014)
11. Gretton, A., Borgwardt, K.M., Rasch, M.J., Schölkopf, B., Smola, A.J.: A kernel two-sample test. JMLR **13**, 723–773 (2012)
12. Hays, J., Essa, I.A.: Image and video based painterly animation. In: NPAR. pp. 113–120 (2004)
13. He, M., Chen, D., Liao, J., Sander, P.V., Yuan, L.: Deep exemplar-based colorization. TOG **37**(4), 47:1–47:16 (2018)
14. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: ECCV. pp. 694–711 (2016)
15. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR **abs/1412.6980** (2014)
16. Kyprianidis, J.E., Collomosse, J.P., Wang, T., Isenberg, T.: State of the "art": A taxonomy of artistic stylization techniques for images and video. TVCG **19**(5), 866–885 (2013)
17. Li, C., Wand, M.: Precomputed real-time texture synthesis with markovian generative adversarial networks. In: ECCV. pp. 702–716 (2016)
18. Lu, J., Sander, P.V., Finkelstein, A.: Interactive painterly stylization of images, videos and 3d animations. In: SI3D. pp. 127–134 (2010)
19. Mahendran, A., Vedaldi, A.: Understanding deep image representations by inverting them. In: CVPR. pp. 5188–5196 (2015)
20. Mathieu, M., Couprie, C., LeCun, Y.: Deep multi-scale video prediction beyond mean square error pp. 1–14 (2016)
21. Mathieu, M., Zhao, J.J., Sprechmann, P., Ramesh, A., LeCun, Y.: Disentangling factors of variation in deep representation using adversarial training. In: NIPS. pp. 5041–5049 (2016)
22. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. CoRR **abs/1511.06434** (2015)

23. Revaud, J., Weinzaepfel, P., Harchaoui, Z., Schmid, C.: Epicflow: Edge-preserving interpolation of correspondences for optical flow. In: CVPR. pp. 1164–1172 (2015)
24. Ruder, M., Dosovitskiy, A., Brox, T.: Artistic style transfer for videos. In: GCPR. pp. 26–36 (2016)
25. Ulyanov, D., Lebedev, V., Vedaldi, A., Lempitsky, V.S.: Texture networks: Feed-forward synthesis of textures and stylized images. In: ICML. pp. 1349–1357 (2016)
26. Vondrick, C., Pirsiavash, H., Torralba, A.: Generating videos with scene dynamics. In: NIPS. pp. 613–621 (2016)
27. Weinzaepfel, P., Revaud, J., Harchaoui, Z., Schmid, C.: Deepflow: Large displacement optical flow with deep matching. In: ICCV. pp. 1385–1392 (2013)
28. Xu, T., Zhang, P., Huang, Q., Zhang, H., Gan, Z., Huang, X., He, X.: Attngan: Fine-grained text to image generation with attentional generative adversarial networks pp. 1–9 (2018)
29. Zhang, H., Xu, T., Li, H.: Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In: ICCV. pp. 5908–5916 (2017)
30. Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., Metaxas, D.N.: Stackgan++: Realistic image synthesis with stacked generative adversarial networks. CoRR **abs/1710.10916** (2017)
31. Zhang, R., Isola, P., Efros, A.A.: Colorful image colorization. In: ECCV. pp. 649–666 (2016)
32. Zhang, S., Li, X., Hu, S., Martin, R.R.: Online video stream abstraction and stylization. TMM **13**(6), 1286–1294 (2011)
33. Zhu, J., Krähenbühl, P., Shechtman, E., Efros, A.A.: Generative visual manipulation on the natural image manifold. In: ECCV. pp. 597–613 (2016)
34. Zhu, J., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: ICCV