Coherent Online Video Style Transfer

Dongdong Chen[*1], Jing Liao[2], Lu Yuan[2], Nenghai Yu[1], and Gang Hua[2]

[1]University of Science and Technology of China,
`cd722522@mail.ustc.edu.cn, ynh@ustc.edu.cn`
[2]Microsoft Research Asia,
`{jliao,luyuan,ganghua}@microsoft.com`

Abstract

Training a feed-forward network for fast neural style transfer of images is proven to be successful. However, the naive extension to process video frame by frame is prone to producing flickering results. We propose the first end-to-end network for online video style transfer, which generates temporally coherent stylized video sequences in near real-time. Two key ideas include an efficient network by incorporating short-term coherence, and propagating short-term coherence to long-term, which ensures the consistency over larger period of time. Our network can incorporate different image stylization networks. We show that the proposed method clearly outperforms the per-frame baseline both qualitatively and quantitatively. Moreover, it can achieve visually comparable coherence to optimization-based video style transfer, but is three orders of magnitudes faster in runtime.

1. Introduction

Inspired by the success of work from Gatys et al. [16] on neural style transfer, there have been a surge of recent works [36, 27, 8, 17] addressing the problem of style transfer using deep neural networks.

In their approaches, style transfer is formulated as an optimization problem, *i.e.*, starting with white noise, searching for a new image presenting similar neural activations as the content image and similar feature correlations as the style image. Notwithstanding their impressive results, these methods are very slow in runtime due to the heavy iterative optimization process. To mitigate this issue, many works have sought to speed up the transfer by training feed-forward networks [23, 38, 28, 9, 11, 29]. Such techniques have been successfully applied to a number of popular apps such as Prisma, Pikazo, DeepArt, *etc*.

Extending neural style transfer form image to video may produce new and impressive effects, whose appeal is especially strong in short videos sharing, live-view effects, and movie entertainments. The approaches discussed above, when naively extended to process each frame of the video one-by-one, often lead to flickering and false discontinuities. This is because the solution of the style transfer task is not stable. For optimization-based methods (*e.g.*, [16]), the instability stems from the random initialization and local minima of the style loss function. And for those methods based on feed-forward networks (*e.g.*, [23]), a small perturbation in the content images, *e.g.*, lighting, noises and motions may cause large variations in the stylized results, as shown in Figure 1. Consequently, it is essential to explore temporal consistency in videos for stable outputs.

Anderson et al. [1] and Ruder et al. [35] address the problem of flickers in the optimization-based method by introducing optical flow to constrain both the initialization and the loss function. Although very impressive and smoothing stylized video sequences are obtained, their runtime is quite slow (usually several minutes per frame), making it less practical in real-world applications.

In search for a fast and yet stable solution to video style transfer, we present the first feed-forward network leveraging temporal information for video style transfer, which is able to produce consistent and stable stylized video sequences in near real-time. Our network architecture is constituted by a series of the same networks, which considers two-frame temporal coherence. The basic network incorporates two sub-networks, namely the flow sub-network and the mask sub-network, into a certain intermediate layer of a pre-trained stylization network (*e.g.*, [23, 9]).

The flow sub-network, which is motivated by [43], estimates dense feature correspondences between consecutive frames. It helps all consistent points along the motion trajectory be aligned in feature domain. The mask sub-network identifies the occlusion or motion discontinuity re-
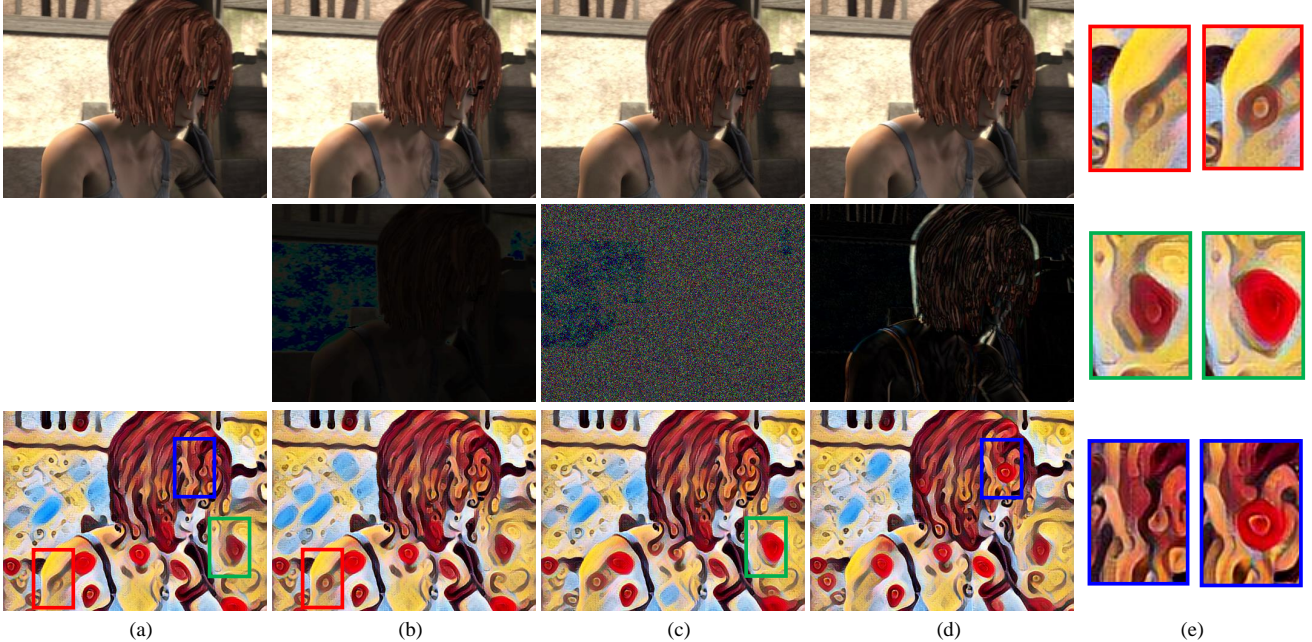

Figure 1. The image stylization network (*e.g.*, [23]), will amplify image variances caused by some unnoticeable changes in inputs. Upper row shows the four inputs: (a) the original one, (b) 5% lighter than (a), (c) Gaussian noises ($\mu = 0, \sigma = 1e - 4$) added to (a); and (d) the next frame of (a) with subtle motions. The middle rows show the absolute difference between (a) and other three inputs. For better visualization, these differences are boosted by $3\times$. The bottom row shows the corresponding stylization results. (e) shows close-up views of some flickering regions.

gions. It helps adaptively blend feature maps from previous frames and the current frame to avoid ghosting artifacts. The entire architecture is trained end-to-end, and minimizes a new loss function, jointly considering stylization and temporal coherence.

There are two kinds of temporal consistency in videos, as mentioned in [35]: long-term consistency and short-term consistency. Long-term consistency is more appealing since it produces stable results over larger periods of time, and even can enforce consistency of the synthesized frames before and after the occlusion. This constraint can be easily enforced in optimization-based methods [35]. Unfortunately, it is quite difficult to incorporate it in feed-forward networks, due to limited batch size, computation time and cache memory. Therefore, short-term consistency seems to be more affordable by feed-forward network in practice.

Therefore, our solution is a kind of compromise between consistency and efficiency. Our network is designed to mainly consider short-term relationship (only two frames), but the long-term consistency is partially achieved by propagating the short-term ones. Our network may directly leverage the composite features obtained from the previous frame, and combine it with features at the current frame for the propagation. In this way, when the point can be traced along motion trajectories, the feature can be propagated until the tracks end.

This approximation may suffer from shifting errors in propagation, and inconsistency before and after the occlusion. Nevertheless, in practice, we do not observe obvious ghosting or flickering artifacts through our online method, which is necessary in many real applications. In summary, our proposed video style transfer network is unique in the following aspects:

- Our network is the first network leveraging temporal information that is trained end-to-end for video style transfer, which successfully generates stable results.

- Our feed-forward network is about thousands of times faster compared to optimization-based style transfer in videos [1, 35], reaching 15 fps on modern GPUs.

- Our method enables online processing, and is cheap in both learning and inference, since we achieve the good approximation of long-term temporal coherence by propagating short-term one.

- Our network is general, and successfully applied to several existing image stylization networks, including per-style-per-network [23] or mutiple-style-per-network [9].

2. Related Work

2.1. Style Transfer for Images and Videos

Traditional image stylization work mainly focus on texture synthesis based on low-level features, which uses non-

parametric sampling of pixels or patches in given source texture images [13, 20, 12] or stroke databases [30, 19]. Their extension to video mostly uses optical flow to constrain the temporal coherence of sampling [4, 18, 31]. A comprehensive survey can be found in [25].

Recently, with the development of deep learning, using neural networks for stylization becomes an active topic. Gatys et al. [16] first propose a method of using pre-trained Deep Convolutional Neural Networks (CNN) for image stylization. It generates more impressive results compared to traditional methods because CNN provides more semantic representations of styles. To further improve the transfer quality, different complementary schemes have been proposed, including face constraints [36], Markov Random Field (MRF) prior [27], user guidance [8] or controls [17]. Unfortunately, these methods based on an iterative optimization are computationally expensive in run-time, which imposes a big limitation in real applications. To make the run-time more efficient, some work directly learn a feed-forward generative network for a specific style [23, 38, 28] or multiple styles [9, 11, 29] which are hundreds of times faster than optimization-based methods.

Another direction of neural style transfer [16] is to extend it to videos. Naive solution that independently processes each frame produces flickers and false discontinuities. To preserve temporal consistency, Alexander et al. [1] use optical flow to initialize the style transfer optimization, and incorporate flow explicitly into the loss function. To further reduce ghosting artifacts at the boundaries and occluded regions, Ruder et al. [35] introduce masks to filter out the flow with low confidences in the loss function. This allows to generate consistent and stable stylized video sequences, even in cases with large motion and strong occlusions. Notwithstanding their demonstrated success in video style transfer, it is very slow due to the iterative optimization. Feed-forward networks [23, 38, 28, 9, 11, 29] have proven to be efficient in image style transfer. However, we are not aware of any work that trains a feed-forward network that explicitly takes temporal coherence into consideration in video style transfer.

2.2. Temporal Coherence in Video Filter

Video style transfer can be viewed as applying one kind of artistic filter on videos. How to preserve the temporal coherence is essential and has been considered in previous video filtering work. One popular solution is to temporally smooth filter parameters. For instance, Bonneel et al. [2] and Wang et al. [39] transfer the color grade of one video to another by temporally filtering the color transfer functions.

Another solution is to extend the filter from 2D to 3D. Paris et al. [32] extend the Gaussian kernel in bilateral filtering and mean-shift clustering to the temporal domain for many applications of videos. Lang et al. [26] also extend

the notion of smoothing to the temporal domain by exploiting optical flow and revisit optimization-based techniques such as motion estimation and colorization. These temporal smoothing and 3D extension methods are specific to their applications, and cannot generalize to other applications, such as stylization.

A more general solution considering temporal coherence is to incorporate a post-processing step which is blind to filters. Dong et al. [10] segment each frame into several regions and spatiotemporally adjust the enhancement (produced by unknown image filters) of regions of different frames; Bonneel et al. [3] filter videos along motion paths using a temporal edge-preserving filter. Unfortunately, these post-processing methods fracture texture patterns, or introduce ghosting artifacts when applied to the stylization results due to high demand of optical flow.

As for stylization, previous methods (including traditional ones [4, 18, 31, 42] and neural ones [1, 35]) rely on optical flow to track motions and keep coherence in color and texture patterns along the motion trajectories. Nevertheless, how to add flow constraints to feed-forward stylization networks has not been investigated before.

2.3. Flow Estimation

Optical flow is known as an essential component in many video tasks. It has been studied for decades and numerous approaches has been proposed [21, 5, 40, 6, 41, 34]). These methods are all hand-crafted, which are difficult to be integrated in and jointly trained in our end-to-end network.

Recently, deep learning has been explored to solving optical flow. FlowNet [15] is the first deep CNNs designed to directly estimate the optical flow and achieve good results. Later, its successors focused on accelerating the flow estimation [33], or achieving better quality [22]. Zhu et al. [43] recently integrate the FlowNet [15] to image recognition networks and train the network end-to-end for fast video recognition. Our work is inspired by their idea of applying FlowNet to existing networks. However, the stylization task, different from the recognition one, requires some new factors to be considered in network designing, such as the loss function, and feature composition, *etc*.

3. Method

3.1. Motivation

When the style transfer for consecutive frames is applied independently (*e.g.*, [23]), subtle changes in appearance (*e.g.*, lighting, noise, motion) would result in strong flickering, as shown in Figure 1. By contrast, in still-image style transfer, such small changes in the content image, especially on flat regions, may be necessary to generate spatially rich and varied stylized patterns, making the result more impressive. Thus, how to keep such spatially rich and
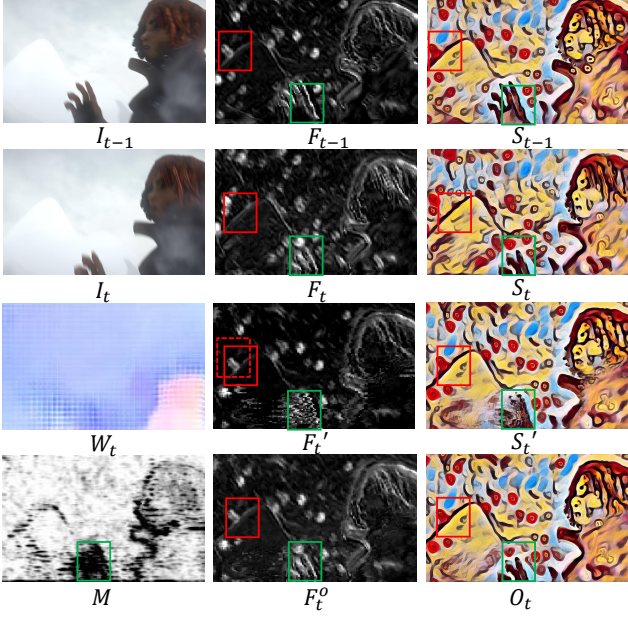
Figure 2. Visualization of two-frame temporal consistency. Two inputs I_{t-1}, I_t pass the stylization network [23] to obtain feature maps F_{t-1}, F_t, and stylized results S_{t-1}, S_t. We may notice discontinuities between S_{t-1} and S_t in *Red* and *Green* rectangles. The third row shows the warped feature map F_t' and styled result S_t' through flow W_t. We can see texture patterns (*Red* rectangle) are successfully traced from $t-1$ to t, but ghosting occurs at occluded regions (*Green* rectangle) of S_t'. The occlusion mask is shown as M. In these false regions, F_t' (also S_t') is replaced with F_t (also S_t) to get the composite features F_t^o and result O_t.

interesting texture patterns, while preserving the temporal consistency in videos is worthy of a more careful study.

For simplicity, we start by exploring temporal coherence between two frames. Our intuition is to warp the stylized result from the previous frame to the current one, and adaptively fuse both together. In other words, some traceable points/regions from the previous frame keep unchanged, while some untraceable points/regions use new results occurring at the current frame. Such an intuitive strategy strikes two birds in one stone: 1) it makes sure stylized results along the motion paths to be as stable as possible; 2) it avoids ghosting artifacts for occlusions or motion discontinuities. We show the intuitive idea in Figure 2.

The strategy outlined above only preserves the short-term consistency, which can be formulated as the problem of propagation and composition. The issue of propagation relies on good and robust motion estimation. Instead of optical flow, we are more inclined to estimate flow on deep features, similar to [43], which may neglect noise and small appearance variations and hence lead to more stable motion estimation. This is crucial to generate stable stylization videos, since we desire appearance in stylized video frames not to be changed due to such variations. The issue of composition is also considered in the feature domain instead of
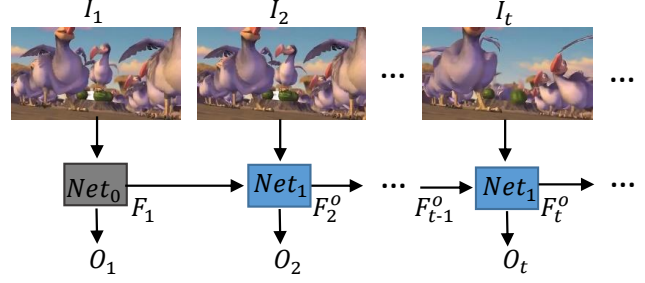

Figure 3. system overview.

pixel domain, since it can further avoid seam artifacts.

To further obtain the consistency over long periods of time, we seek a new architecture to propagate short-term consistency to long-term. The pipeline is shown in Figure 3. At $t-1$, we obtain the composite feature maps F_{t-1}^o, which are constrained by two-frame consistency. At t, we reuse F_{t-1}^o for propagation and composition. By doing so, we expect all traceable points to be propagated as far as possible in the entire video. Once the points are occluded or the tracking get lost, the composite features will keep values independently computed at the current frame. In this way, our network only needs to consider two frames every time, but still approaches long-term consistency.

3.2. Network Architecture

In this section, we explain the details of our proposed end-to-end network for video style transfer. Given the input video sequence $\{I_t|t=1...n\}$, the task is to obtain the stylized video sequence $\{O_t|t=1...n\}$. The overall system pipeline is shown in Figure 3. At the first frame I_1, it uses existing stylization network (*e.g.*, [23]) denoted as Net_0 to produce the stylized result. Meanwhile, it also generates the encoded features F_1 as the input of our proposed network Net_1 at the second frame I_2. The process is iterated over the entire video sequence. Starting from the second frame I_2, we use Net_1 rather than Net_0 for style transfer.

The proposed network structure Net_1 incorporating two-frame temporal coherence is presented in Figure 4. It consists of three main components: the style sub-network, the flow sub-network, and the mask sub-network.

Style Sub-network. We adopt the pre-trained image style transfer network of Johnson et al. [23] as our default style sub-network, since it is often adopted as the basic network structure for many follow-up work (*e.g.*, [11, 9]). This kind of network looks like auto-encoder architecture, with some strided convolution layers as the encoder and fractionally strided convolution layers as the decoder, respectively. Such architectures allow us to insert the flow sub-network and the mask sub-network between the encoder and the decoder. In Section 4.4, we provide the detailed analysis on which layer is better for the integration of our sub-networks.
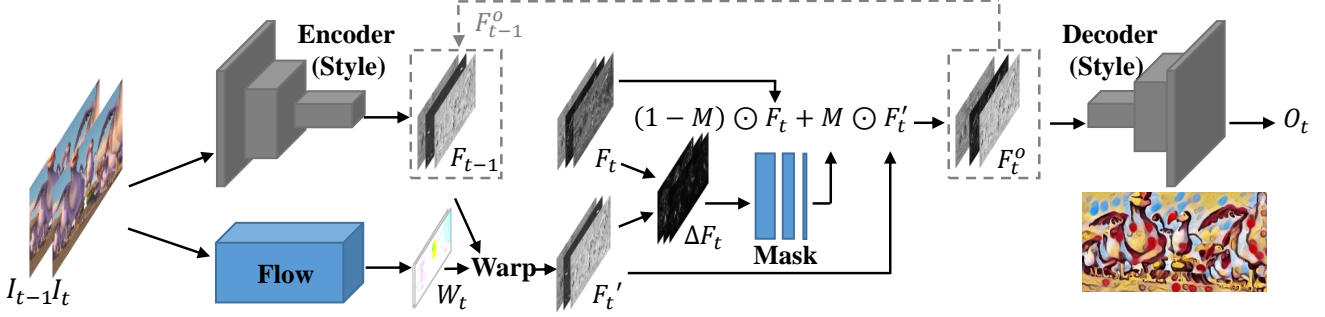
4

Figure 4. Our network architecture consists of three main components: the pretrained style sub-network, which is split into two parts: an encoder and a decoder; the flow sub-network to predict intermediate feature flow; and the mask sub-network to regress the composition mask.

Flow Sub-network. As a part for temporal coherence, the flow sub-network is designed to estimate the correspondences between two consecutive frames I_{t-1} and I_t, and then warp the convolutional features. We adopt *FlowNet* (the "Simple" version) [15] as our flow sub-network by default. It is pre-trained on the synthetic *Flying Chairs* dataset [15] for optical flow task, and should be fine-tuned to produce feature flow suitable for our task.

The process is similar to [43], which uses it for video recognition. Two consecutive frames I_{t-1}, I_t are firstly encoded into feature maps F_{t-1}, F_t respectively by the encoder. W_t is the feature flow generated by the flow sub-network and bilinearly resized to the same spatial resolution as F_{t-1}. As the values of W_t are in general fractional, we warp F_{t-1} to F'_t via bilinear interpolation:

$$F'_t = \mathcal{W}^t_{t-1}(F_{t-1}) \tag{1}$$

where $\mathcal{W}^t_{t-1}(\cdot)$ denotes the function that warps features from $t-1$ to t using the estimated flow field W_t, namely $F'_t(p) = F_{t-1}(p + W_t(p))$, where p denotes spatial location in feature map and flow.

Mask Sub-network. Given the warped feature F'_t and the original feature F_t, the mask sub-network is employed to regress the composition mask M, which is then adopted to compose both features F'_t and F_t. The value of M varies from 0 to 1. For traceable points/regions by the flow (*e.g.*, static background), the value in the mask M tends to be 1. It suggests that the warped feature F'_t should be reused so as to keep coherence. On the contrary, at occlusion or false flow points/regions, the value in the mask M is 0, which suggests F_t should be adopted. The mask sub-network architecture consists of three convolutional layers with stride one. Its input is the absolute difference of two feature maps

$$\Delta F_t = |F_t - F'_t|, \tag{2}$$

and the output is a single channel mask M, which means all feature channels would share the same mask in the later

composition. Here, we obtain the composite features F^o_t by linear combination of F_t and F'_t:

$$F^o_t = (1 - M) \odot F_t + M \odot F'_t \tag{3}$$

where $\odot$ represents element-wise multiplication.

Summary of Net_1. Figure 4 summarizes our network Net_1 designed for two frames. Given two input frame I_{t-1}, I_t, they are fed into the encoder of fixed style sub-network, generating convolutional feature maps F_{t-1}, F_t. This first step is different in inference, where F_{t-1} will not be computed from I_{t-1}, and instead borrowed from the obtained composite features F^o_{t-1} at $t-1$. It is illustrated by the dot lines in Figure 4. On the other branch, both frames I_{t-1}, I_t are fed into the flow sub-network to compute feature flow W_t, which warps the features F_{t-1} (F^o_{t-1} used in inference instead) to F'_t. Next, the difference ΔF_t between F_t and F'_t is fed into the mask sub-network, generating the mask M. New features F^o_t are achieved by linear combination of F_t and F'_t weighted by the mask M. Finally, F^o_t is fed into the decoder of the style sub-network, generating the stylized result O_t at frame t. For the inference, F^o_t is also the output for the next frame $t+1$. Since both flow and mask sub-networks learn relative flow W_t and mask M_t between any two frames, it is not necessary for our training to incorporate historic information (*e.g.*, F^o_{t-1}) as well as the inference. It can make our training be simple.

3.3. The Loss Function

To train both the flow and mask sub-networks, we define the loss function by enforcing three terms: the coherence term $\mathcal{L}_{cohe}$, the occlusion term $\mathcal{L}_{occ}$, and the flow term $\mathcal{L}_{flow}$. The coherence term $\mathcal{L}_{cohe}$ penalizes the inconsistencies between stylized results of two consecutive frames.

$$\mathcal{L}_{cohe}(O_t, S_{t-1}) = M^g \odot ||O_t - \mathcal{W}^t_{t-1}(S_{t-1})||^2, \tag{4}$$

where S_{t-1} is the stylized result produced independently at $t-1$. The warping function $\mathcal{W}^t_{t-1}(\cdot)$ uses the ground-truth

flow W_t^g. M^g is the ground-truth mask, where 1 represents consistent points/regions and 0 represents untraceable ones. It encourages the stylized result O_t to be consistent with S_{t-1} in the traceable points/regions.

On the contrary, in the untraceable regions (e.g. occlusions), the occlusion term $\mathcal{L}_{occ}$ enforces O_t to be close to the independently stylized result S_t at frame I_t:

$$\mathcal{L}_{occ}(O_t, S_t) = (1 - M^g) \odot ||O_t - S_t||^2. \qquad (5)$$

Besides, we add a term to constrain the feature flow:

$$\mathcal{L}_{flow} = ||W_t - W_t^g \downarrow||^2. \qquad (6)$$

Here we use the down-scaled version of the ground-truth optical flow $W_t^g \downarrow$, which is re-scaled to the same size of W_t, to serve as the guidance for feature flow estimation.

In summary, our loss function to train flow and mask sub-networks is the weighted aevarge of three terms.

$$\mathcal{L} = \alpha \mathcal{L}_{cohe} + \beta \mathcal{L}_{occ} + \lambda \mathcal{L}_{flow}, \qquad (7)$$

where $\alpha = 1e5$, $\beta = 2e4$ and $\lambda = 20$ by default.

Note that our loss function discards the content and style loss for training the original style network, because the pre-trained style sub-network is fixed during the training period of the flow and mask sub-networks. We believe that S_t (or S_{t-1}) itself can provide sufficient style supervision in learning. One extra benefit is that we can directly leverage other trained still-image style models and apply it to videos directly. In this sense, our proposed framework is general.

4. Experiments

4.1. Dataset Set-up

Our task requires a big video dataset with varied types of motions and ground-truth optical flow. However, existing datasets are quite small, *e.g.*, the synthetic *MPI Sintel* dataset [7] (only has 1,064 frames totally). Instead, we collect ten short videos (eight animation movies episode of *Ice Age*, and two real videos from YouTube), around $28,000$ frames together as our training dataset.

To obtain approximated ground-truth flow W^g between every two consecutive frames in these videos, we use Deep-Flow2 [41] to compute the bidirectional optical flow and use the backward flow as the ground-truth.

As for the ground-truth of the composition mask M^g, we adopt the methods used in [35, 37] to detect occlusions and motion boundaries. We mask out two types of pixels, being set to 0 in M^g: 1) the occlusion pixels achieved by cross-checking the forward and backward flows; 2) the pixels at motion boundaries with large gradients of flow, which are often less accurate and may result in ghosting artifacts in composition. All other pixels in M^g are set to 1.

Methods	stability error e_{stab}				runtime
	Muse	*Candy*	*Scream*	*Udnie*	(fps)
Johnson et al. [23]	0.0199	0.0240	0.0048	0.0108	38.17
[23]+Ours	**0.0121**	**0.0105**	**0.0034**	**0.0076**	**15.07**
[23]+Ours††	0.0135	0.0120	0.0036	0.0079	15.07
Dong et al. [9]	0.0159	0.0181	0.0035	0.0059	20.6
[9] + Ours	**0.0126**	**0.0131**	**0.0030**	**0.0048**	**7.35**

Table 1. Comparison of different methods on stability error and runtime (GPU Titan X). Compared to the per-frame processing baseline [23] or [9], our method can obtain much lower stability loss while only $2.5 \sim 2.8\times$ slower. Compared to fixed flow sub-network (indicated by ††), our fine-tuned flow sub-network achieves better coherence.

We use the *MPI Sintel* [7] as the test dataset, which is widely adopted for optical flow evaluation. It contains 23 short videos and is labeled with ground-truth flow and occlusion mask. The dataset covers various types of real scenarios, such as large motions and motion blurs.

4.2. Implementation details

In our experiments, we adopt two types of pre-trained style network (per-style-per-net [23][1], multiple-style-per-net [9][2]) as our fixed style sub-network. We train the flow sub-network and mask sub-network on the video dataset described in Section 4.1. All videos have the image resolutions of 640×360. The network is trained with a batch size of 1 (frame pair) for 100k iterations. And the Adam optimization method [24] is adopted with the initial learning rate of $1e-4$ and decayed by 0.8 at every $5k$ iterations.

4.3. Quantitative and Qualitative Comparison

For video style transfer, runtime and temporal consistency are two key criteria. Runtime uses the frame rate of inference. The temporal consistency is measured by

$$e_{stab}(O_t, O_{t-1}) = M^g \odot ||O_t - \mathcal{W}_{t-1}^t(O_{t-1})||^2, \quad (8)$$

where the stability error $e_{stab}(O_t, O_{t-1})$ measures the coherence loss (in Equation (4)) between two results O_t and O_{t-1}. Here, we only evaluate stability of results on traceable regions. Lower stability error indicates more stable result. For the entire video, we use the average error instead.

Quantitative Results. To validate the effectiveness of our method, we test and compare using two existing stylization networks [23, 9]. The baseline for comparison is to apply their networks to process each frame independently. As shown in Table 1, for all the four styles, our method obtains much lower stability error than the baseline [23, 9]. As

---

[1]In our experiment, we adopt the released model of [23]. It uses two stride-2 convolutions to down-scale the input followed by five residual blocks and then two convolutional layers with stride-1/2 to up-scale, but its channel number of all convolutional layers is half of [23].

[2]We slightly modified the StyleBank model [9], whose encoder and decoder sub-networks adopted the same structures as [23], but the stylebank layer is inserted after the third residual block.
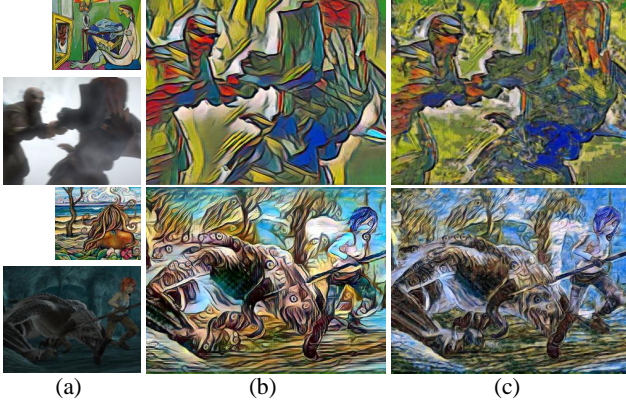
Figure 5. Comparison of our results (b) and results of [3] (c) on the same inputs (a). Their post-processing scheme results in ghosting and blurring artifacts for video style transfer task.

for the runtime, our method is around $2.5 \sim 2.8\times$ slower than the baseline, because our network may need extra computation in both flow and mask sub-networks. Nevertheless, our method is still near real-time (15 fps in Titan X).

As a reference, we also test the optimization method [35] with the *Candy* style on our test database. Ours is with slightly larger temporal coherence error compared to theirs (0.0067), because our network is trained for all videos while theirs is optimized for one. As for the speed, ours is thousands of times faster than theirs (0.0089 fps).

Qualitative Results. In Figure 6, we show three examples with kinds of representative motions to visually compare our results with per-frame processing models [9, 23]. These results clearly show that our methods successfully reduce temporal inconsistency artifacts which appear in these per-frame models. In the nearly static scene (*First Row*), ours can keep the scene unchanged after stylization while the per-frame models fail. As for the scenes with motions, including both camera motions (*Second Row*) and object motions (*Third Row*), our method keeps the coherence between two frames except for the occluded regions. (*The comparisons in our supplementary video* [3] *are highly recommended for better visualization.*)

We further compare our method with a post-processing method [3], which is applied to the stylized results produced by per-frame model [23]. As shown in Figure 5, the results produced from the post-processing method [3] look not so clear as ours, and produces ghosting artifacts. This is because optimizing temporal coherence after stylization may not be able to obtain the global optima for both temporal coherence and stylization.

$A + B$ with [23]	e_{stab}	$A + B$ with [9]	e_{stab}
Scream + Scream	0.0034	*Scream + Scream*	0.0031
Scream + Candy	0.0042	*Scream + **multiple***	0.0032
Candy + Scream	0.0137	*Candy + Candy*	0.0121
Candy + Candy	0.0105	*Candy + **multiple***	0.0123

Table 2. Cross comparison of transferring flow and mask sub-networks of A to B. A represents the style of pretrained style sub-network, and B is the style which flow and mask sub-networks are trained for. In [9], B can be multiple styles.

4.4. Ablation Study

Layer Choice for Feature Composition. To study which layer of the style sub-network is the best for our feature propagation and composition, we try different layers for integration. For the basic style network [23], we find 5 intermediate feature layers from input to output (respectively with 1,1/2,1/4,1/2,1 times of original resolution), which allow our flow and mask sub-networks being integrated. The five settings are trained and tested on the same database and with the same style.

In this experiment, we measure the sharpness of their stylization results by *Perceptual Sharpness Index* (PSI) [14], in addition to the stability error (in Equation (8)). Table 3 clearly shows that the stability is improved from input to output layers, while the sharpness decreases. It may result from the observation that the stylization networks (*e.g.*, [23]) will amplify the image variances as shown in Figure 1. When feature flow estimation and composition happen closer to the input layer, small inconsistencies in composite features would also be amplified, causing incoherent results. When they happen closer to the output layer, blending already amplified differences become more difficult and may introduce strong ghosting artifacts. To strike for a balance between stability and image sharpness, we recommend to integrate our sub-networks into the middle layer of stylization networks, *i.e.*, r1/4(E). In this layer, the image content is compressed as much as possible, which may be beneficial to robust flow estimation and feature composition.

Fixed Flow Sub-network. In our experiment, *FlowNet* [15] is adopted as our flow sub-network. Original *Flownet* is trained in image domain for optical flow. It needs to be fine tuned on our task, since the flow would be further improved by jointly learning stylization and temporal coherence. Here, we compare fixed and fine-tuned flow sub-network. As shown in Table 1, fixed flow sub-network obtains less temporally coherent results than fine-tuned one.

Transferability. To know whether our trained flow and mask sub-networks can be used to a new style (not appearing in training), we conduct two experiments respectively on per-style-per-net [23] and multiple-style-per-net [9]. In per-style-per-net [23], we use two different styles, named
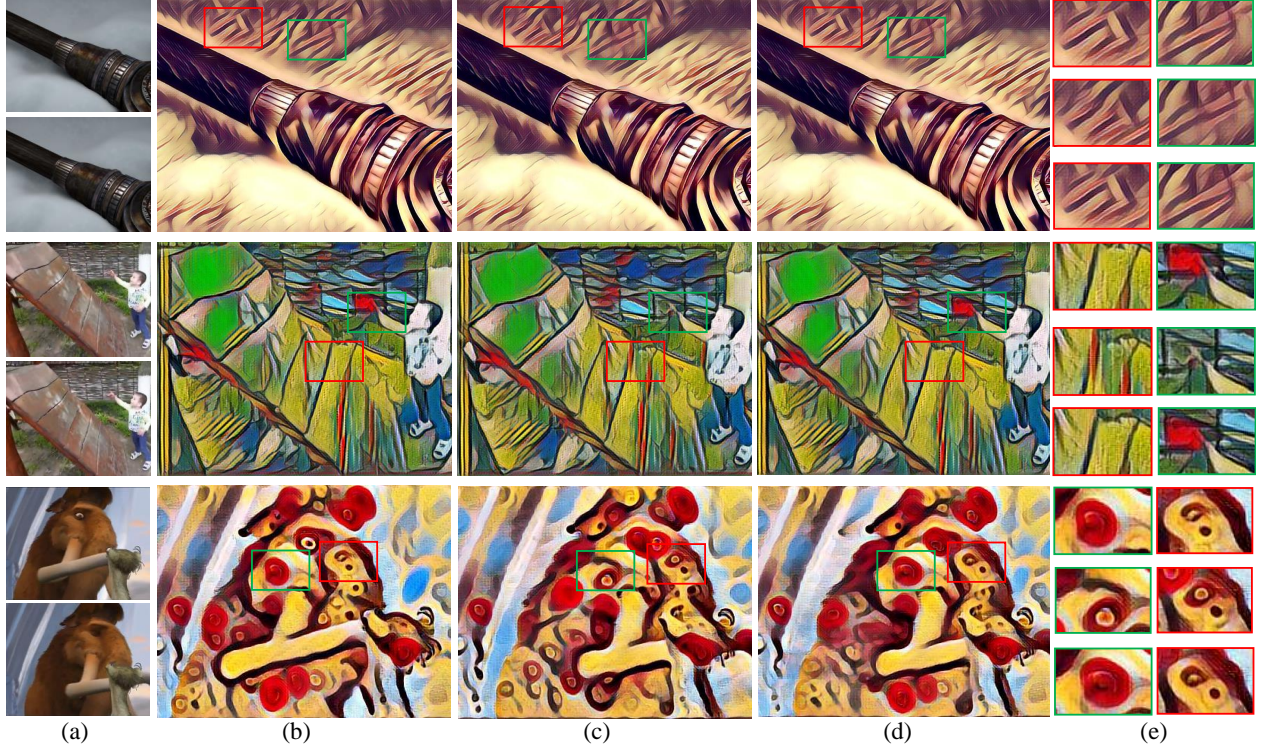
Figure 6. Qualitative comparison results: (a) consecutive frames pair (top: frame t, bottom: frame $t + 1$); (b) stylization result of frame t; (c) stylization result of frame $t + 1$ from baseline; (d) stylization result of frame $t + 1$ from our method; (e) top to down: dilated marked regions corresponding to (b),(c),(d) respectively. The top row are with [9] on a nearly static scene, and bottom two rows are with [23] on a scene with camera motion or object motion. Compared to baseline of [23, 9], our results are all more temporal coherent.

	Baseline	r1(E)	r1/2(E)	r1/4(E)	r1/2(D)	r1(D)
e_{stab}	0.0199	0.0187	0.0180	0.0121	0.0058	0.0038
PSI	0.4851	0.4846	0.4839	0.4825	0.4187	0.4086

Table 3. Layer choice for feature composition. r1, r1/2, r1/4 represent different layers whose feature map resolution is $1, 1/2, 1/4\times$ of the original image, and E and D represent encoder and decoder respectively. The top table shows stability error e_{stab} and PSI for different settings. One visual example is shown on the bottom row.

as A and B for cross experiments. One combination is style sub-network learned from A, and our flow and mask sub-networks learned from B. The other combination is reversed. As shown in Table 2 (*First Column*), it is hard to preserve the original stability when our sub-networks trained on one style are applied to another. By contrast, in multiple-style-per-net [9], our trained sub-networks can be directly used to two new styles without re-training, while preserving the original stability, as shown in Table 2 (*Second Column*). The observation suggests that our sub-

networks learned with multiple-style-per-net [9] can be independent of styles, which is beneficial to real applications.

5. Conclusion and Discussion

In this paper, we present the first end-to-end training system by incorporating temporal coherence for video style transfer, which can speed up existing optimization-based video style transfer ([1, 35]) by thousands of times, and achieve near real-time speed on modern GPUs. Moreover, our network achieves the long-term temporal coher-

ence through the propagation of the short-term ones, which enables our model for online processing. It can be successfully employed in existing stylization networks [23, 9], and can even be directly used for new styles without re-training. Our method can produce stable and visually appealing stylized videos in the presence of camera motions, object motions, and occlusions.

There are still some limitations in our method. For instance, limited by the accuracy of ground-truth optical flow (given by DeepFlow2 [41]), our results may suffer from some incoherence where the motion is too large for the flow to track. And after propagation over a long period, small flow errors may accumulate, causing blurriness. These open questions are interesting for further exploration in the future work.

Acknowledgement

References

[1] A. G. Anderson, C. P. Berg, D. P. Mossing, and B. A. Olshausen. Deepmovie: Using optical flow and deep neural networks to stylize movies. *arXiv preprint arXiv:1605.08153*, 2016.

[2] N. Bonneel, K. Sunkavalli, S. Paris, and H. Pfister. Example-based video color grading. *ACM Trans. Graph.*, 32(4):39–1, 2013.

[3] N. Bonneel, J. Tompkin, K. Sunkavalli, D. Sun, S. Paris, and H. Pfister. Blind video temporal consistency. *ACM Trans. Graph. (Proc. of SIGGRAPH Asia)*, 34(6):196, 2015.

[4] A. Bousseau, F. Neyret, J. Thollot, and D. Salesin. Video watercolorization using bidirectional texture advection. In *ACM Trans. Graph. (Proc. of SIGGRAPH)*, volume 26, page 104. ACM, 2007.

[5] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *Proc. ECCV*, pages 25–36. Springer, 2004.

[6] T. Brox and J. Malik. Large displaczement optical flow: descriptor matching in variational motion estimation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(3):500–513, 2011.

[7] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *Proc. ECCV*, pages 611–625. Springer, 2012.

[8] A. J. Champandard. Semantic style transfer and turning two-bit doodles into fine artworks. *arXiv preprint arXiv:1603.01768*, 2016.

[9] D. Chen, L. Yuan, J. Liao, N. Yu, and G. Hua. Stylebank: An explicit representation for neural image style transfer. In *Proc. CVPR*, 2017.

[10] X. Dong, B. Bonev, Y. Zhu, and A. L. Yuille. Region-based temporally consistent video post-processing. In *Proc. CVPR*, pages 714–722, 2015.

[11] V. Dumoulin, J. Shlens, and M. Kudlur. A learned representation for artistic style. *arXiv preprint arXiv:1610.07629*, 2016.

[12] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *Proc. ACM SIGGRAPH*, pages 341–346. ACM, 2001.

[13] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *Proc. ICCV*, volume 2, pages 1033–1038. IEEE, 1999.

[14] C. Feichtenhofer, H. Fassold, and P. Schallauer. A perceptual image sharpness metric based on local edge gradient analysis. *IEEE Signal Processing Letters*, 20(4):379–382, 2013.

[15] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. *arXiv preprint arXiv:1504.06852*, 2015.

[16] L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

[17] L. A. Gatys, A. S. Ecker, M. Bethge, A. Hertzmann, and E. Shechtman. Controlling perceptual factors in neural style transfer. *arXiv preprint arXiv:1611.07865*, 2016.

[18] J. Hays and I. Essa. Image and video based painterly animation. In *Proc. of NPAR*, pages 113–120. ACM, 2004.

[19] A. Hertzmann. Painterly rendering with curved brush strokes of multiple sizes. In *Proc. ACM SIGGRAPH*, pages 453–460. ACM, 1998.

[20] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In *Proc. ACM SIGGRAPH*, pages 327–340. ACM, 2001.

[21] B. K. Horn and B. G. Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

[22] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. *arXiv preprint arXiv:1612.01925*, 2016.

[23] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. *arXiv preprint arXiv:1603.08155*, 2016.

[24] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[25] J. E. Kyprianidis, J. Collomosse, T. Wang, and T. Isenberg. State of the" art: A taxonomy of artistic stylization techniques for images and video. *IEEE Trans. on Visualization and Computer Graphics*, 19(5):866–885, 2013.

[26] M. Lang, O. Wang, T. Aydin, A. Smolic, and M. Gross. Practical temporal consistency for image-based graphics applications. *ACM Trans. Graph. (Proc. of SIGGRAPH)*, 31(4):34, 2012.

[27] C. Li and M. Wand. Combining markov random fields and convolutional neural networks for image synthesis. *arXiv preprint arXiv:1601.04589*, 2016.

[28] C. Li and M. Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. *arXiv preprint arXiv:1604.04382*, 2016.

[29] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang. Diversified texture synthesis with feed-forward networks. In *Proc. CVPR*, 2017.

[30] P. Litwinowicz. Processing images and video for an impressionist effect. In *Proc. ACM SIGGRAPH*, pages 407–414. ACM, 1997.

[31] J. Lu, P. V. Sander, and A. Finkelstein. Interactive painterly stylization of images, videos and 3d animations. In *Proc. of I3D*, pages 127–134. ACM, 2010.

[32] S. Paris, S. W. Hasinoff, and J. Kautz. Local laplacian filters: Edge-aware image processing with a laplacian pyramid. *ACM Trans. Graph. (Proc. of SIGGRAPH)*, 30(4):68, 2011.

[33] A. Ranjan and M. J. Black. Optical flow estimation using a spatial pyramid network. *arXiv preprint arXiv:1611.00850*, 2016.

[34] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *Proc. CVPR*, pages 1164–1172, 2015.

[35] M. Ruder, A. Dosovitskiy, and T. Brox. Artistic style transfer for videos. In *Proc. GCPR*, pages 26–36. Springer, 2016.

[36] A. Selim, M. Elgharib, and L. Doyle. Painting style transfer for head portraits using convolutional neural networks. *ACM Trans. Graph. (Proc. of SIGGRAPH)*, 35(4):129, 2016.

[37] N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. In *Proc. ECCV*, pages 438–451. Springer, 2010.

[38] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. *arXiv preprint arXiv:1603.03417*, 2016.

[39] C.-M. Wang, Y.-H. Huang, and M.-L. Huang. An effective algorithm for image sequence color transfer. *Mathematical and Computer Modelling*, 44(7):608–627, 2006.

[40] J. Weickert, A. Bruhn, T. Brox, and N. Papenberg. A survey on variational optic flow methods for small displacements. In *Mathematical models for registration and applications to medical imaging*, pages 103–136. Springer, 2006.

[41] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. Deepflow: Large displacement optical flow with deep matching. In *Proc. CVPR*, pages 1385–1392, 2013.

[42] S.-H. Zhang, X.-Y. Li, S.-M. Hu, and R. R. Martin. Online video stream abstraction and stylization. *IEEE Transactions on Multimedia*, 13(6):1286–1294, 2011.

[43] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei. Deep feature flow for video recognition. *arXiv preprint arXiv:1611.07715*, 2016.