Time Flies: Animating a Still Image with Time-Lapse Video as Reference

Chia-Chi Cheng Hung-Yu Chen Wei-Chen Chiu
National Chiao Tung University, Taiwan
angelwmab.cs07g@nctu.edu.tw chen3381@purdue.edu walon@cs.nctu.edu.tw

Abstract

Time-lapse videos usually perform eye-catching appearances but are often hard to create. In this paper, we propose a self-supervised end-to-end model to generate the time-lapse video from a single image and a reference video. Our key idea is to extract both the style and the features of temporal variation from the reference video, and transfer them onto the input image. To ensure both the temporal consistency and realness of our resultant videos, we introduce several novel designs in our architecture, including class-wise NoiseAdaIN, flow loss, and the video discriminator. In comparison to the baselines of state-of-the-art style transfer approaches, our proposed method is not only efficient in computation but also able to create more realistic and temporally smooth time-lapse video of a still image, with its temporal variation consistent to the reference.

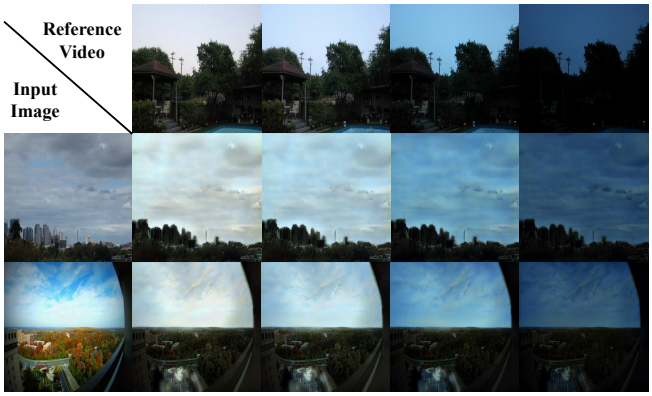

Figure 1. Given a still image (the leftmost column), we aim to generate its corresponding time-lapse animation which has the temporal changes consistent with the reference time-lapse video (as shown in the topmost row).

1. Introduction

Time-lapse videos provide a great way to capture the dynamic world and let us visualize the temporal change of the scenes. As the time-lapse videos are sped up to show longer footage (e.g. seasonal or daily changes) in a shorter period of time, they often present great variance in color appearance and rapid movement, which make them unique and popular nowadays. However, taking a time-lapse video is time-consuming and costly since it typically requires stable hardware devices or mounting equipment for cameras to prevent unwanted camera motions. In order to alleviate the difficulty of shooting time lapse videos, we propose an algorithm of time-lapse synthesis by animating a still image with its temporal change adopted from the existing time-lapse videos that are already available on the internet.

As examples shown in Figure 1, given an image in the leftmost column as our target, we attempt to transfer the continuous and photorealistic changes appearing in the reference video (as shown in the topmost row) into the target image and create the corresponding time-lapse video. Due to the nature of time-lapse videos for having complicated and nonlinear changes in both temporal and spatial domains, which we can usually observe, the resultant time-lapse video based on our synthesis should maintain both the temporal consistency and realness simultaneously.

There have been several works of time-lapse video generation proposed, yet with different problem settings from ours. For instance, the data-driven algorithm proposed by Shih *et al*. [24] relies on a database of time-lapse videos of various scenes to hallucinate a plausible image at a different time of day from an input image. Another deep-learning-based model from [19] learns the correlation between different timing of the day and their corresponding illumination change of an outdoor scene. While being given an outdoor image and various timestamps as inputs, the method can synthesize a time-lapse video with continuous changes in the illumination. Differ from these works, our proposed scenario does not require any database and is able to take arbitrary time-lapse video as reference, thus being more practical and not limited to have only daily or any particular temporal changes.

Without loss of generality, our work can also be seen as an extension of style transfer. Due to the popularity of deep learning techniques nowadays, we have recently witnessed quite some research efforts being devoted to transfer not only the artistic styles [6, 11, 16] but also the photorealistic

ones [15, 17, 32] into the target image, where the latter is more related to our settings to perform transfer across real-world images. Although those photorealistic style transfer algorithms could produce realistic results, they typically are computationally heavy [17, 32] and rely on post-processing steps to reduce the artifacts as well as maintain the realness of the generated output [15, 17]. Moreover, directly applying image-wise style transfer would lead to temporal inconsistency (e.g. flickering or artifacts on the boundary) in the resultant video. Even though there exist several video style transfer methods [1, 10, 21] to improve the temporal consistency, they are aiming to transfer artistic styles onto a video, in which their objective again is obviously different from ours of producing realistic time-lapse videos.

Given a time-lapse video as reference and a still image as our target to animate, our proposed framework of time-lapse generation resolves the aforementioned issues to have both realness and temporal consistency in the output video, by utilizing three main designs: (1) classwise AdaIN and NoiseAdaIN to perform fast style transfer from reference video to the input image and maintain different patterns of temporal variation across different object classes; (2) a new objective function, i.e. flow loss, to encourage the coherence in high-level feature space between the temporal changes in the reference video and those in our generated one; (3) a video discriminator based on adversarial learning to preserve both temporal consistency and realness of video frames simultaneously. Furthermore, another significant advantage of our proposed model is that its training is based on a self-supervised learning scheme. Therefore we do not require any particular dataset with groundtruth pairs between input images and the time-lapse videos.

We experimentally verify the efficacy of our proposed method in comparison to several baselines of style transfer, and demonstrate that our model is able to provide better qualitative results and obtain superior performance in terms of several evaluation metrics. Our contributions in this paper are threefold:

- We design an end-to-end time-lapse generative network which can synthesize a time-lapse video from an input image and a reference video, which is more general for the practical use.

- We introduce classwise NoiseAdaIn, flow loss, and video discriminator to improve both the temporal consistency and realness of the resultant time-lapse videos.

- We develop a self-supervised training scheme for learning time-lapse generation. Our model shows better qualitative and quantitative performance with respect to several baselines.

2. Related Works

Style Transfer. In recent years, style transfer has become a popular research topic together with the renaissance of deep learning [2, 5, 7, 20, 23, 29, 30]. Gatys *et al*. [6] are the first to demonstrate impressive style transfer results by matching the feature Gram matrices between the generated image and the style image based on an iterative optimization process, which however requires quite heavy computation. Some approaches [12, 14, 28] overcome the issue by replacing the optimization process with feed-forward neural networks, but are only able to transfer one fixed style in a single network. Several research works [3, 8, 11, 16, 22, 33] progress to release this limitation and perform universal style transfer (i.e. being able to take any image as the source of style) by matching feature statistics of the content image towards the ones from the style image. Although these methods perform well in transferring artistic styles, they are not suitable for photorealistic style transfer due to the severe distortion of structures. In order to generate photorealistic images upon style transfer, Luan *et al*. [18] introduce an additional photorealism regularization term to preserve the structure, but their algorithm requires heavy computation to solve regularized optimization problem. Other methods [15, 17] try to achieve photorealism by additional post-processing, which instead could blur the final outputs. An end-to-end model [32] proposed lately resolves the aforementioned issues by using wavelet pooling, which separates features into different frequency components for better preservation of the image details. However, even these style transfer algorithms perform well in generating individual frames of the time-lapse video, i.e. transferring the style of each reference video frame into the input image, the final output video could have the problem of temporal inconsistency. There are also several video style transfer approaches [1, 10, 21] proposed recently, which are able to preserve the temporal consistency by adding temporal constraints, but they focus mainly on the artistic style transfer instead of the photorealistic one that we would like to have in our task of time-lapse generation.

Animating Still Images. Creating animation from a single image has been a longstanding research problem in computer vision. Being the most related to our problem scenario of generating time-lapse video, Shih *et al*. [24] predict the appearance of a single outdoor image at different hours by using a color transfer algorithm with reference to the most similar videos in their database, which basically requires huge space of storage at runtime. Another conditional GAN based method [19] takes timestamp as condition, predicting how an outdoor scene looks like at different time of a day. However, this approach can only synthesize daily changes. Different from their objective, our model synthesizes time-lapse video of a still image via referring to

an input time-lapse video (which is taken under the static camera setting basically.) In other words, our method does not require timestamp condition, and the input reference video can be any time-lapse one hence being able to produce arbitrary temporal changes as shown in the reference. Moreover, in the past few years, lots of video synthesis or frame prediction methods [25, 27, 31, 35] based on generative adversarial networks (GANs) have been proposed. They, however, mainly focus on specific target actions or phenomenons thus are clearly different from our task in this paper.

3. Proposed Method

As motivated in the introduction, the objective of our proposed method is to synthesize a time-lapse video from a still image by taking another time-lapse video (captured with the static camera) as the reference of temporal variation. The architecture of our proposed model is illustrated in Figure 2, consisting of three components: generator $\mathcal{G}$, discriminator $\mathcal{D}$, and a feature extractor Φ based on a pre-trained VGG-19 network [26]. In the following, we will describe how we achieve the time-lapse generation in detail.

3.1. Time-Lapse Generation

Our generator $\mathcal{G}$ leverages a typical encoder-decoder architecture as depicted in the left of Figure 2, which is composed of three subnetworks: $E_{\mathcal{G}I}$, $E_{\mathcal{G}v}$, and $D_{\mathcal{G}}$. Let I be the target image that we would like to animate, generator $\mathcal{G}$ sequentially takes every 3 adjacent frames $V = \{F_t, F_{t+1}, F_{t+2}\}$ from the reference video to transfer their temporal changes into I for producing one video frame in the resultant time-lapse. The process of the transfer is based on the idea of stylization stemmed from the technique of adaptive instance normalization (i.e., AdaIN [11]). Basically the feature statistics obtained from V by $E_{\mathcal{G}v}$ are used to modulate the features $E_{\mathcal{G}I}(I)$ extracted from I, where the modulation takes place in the decoder $D_{\mathcal{G}}$ and finally we get the synthesized frame $\tilde{F}_t$ as the output of $D_{\mathcal{G}}$:

$$\begin{aligned}\tilde{F}_t &= \mathcal{G}(I, V) \\ &= D_{\mathcal{G}}(E_{\mathcal{G}I}(I), E_{\mathcal{G}v}(V)).\end{aligned} \tag{1}$$

The design of $E_{\mathcal{G}I}$ and $D_{\mathcal{G}}$ is similar to the one proposed in a recent work of end-to-end photorealistic style transfer model, i.e. WCT^2 [32]. Basically, the $E_{\mathcal{G}I}$ is built upon the ImageNet [4] pre-trained VGG-19 network [26] from `conv1_1` layer to `relu4_1` layer but replaces the max pooling layers with wavelet pooling, in which $E_{\mathcal{G}I}$ is fixed in the training stage. The decoder $D_{\mathcal{G}}$ has a mirror structure of $E_{\mathcal{G}I}$ with using wavelet unpooling layers instead of the typical upsampling ones. The wavelet pooling operations based on Haar kernels can decompose the input feature map into low-frequency components and high-frequency ones, where only the former proceeds to the next layer and get modulated in the decoder by using the statistics from $E_{\mathcal{G}v}(V)$ while the latter is directly skipped to the corresponding layers in the decoder. The motivation of adopting wavelet pooling is to preserve most of the fine structures and details of an image as they are mainly maintained in the high-frequency components, which leads to making the final output more photorealistic. For the encoder $E_{\mathcal{G}v}$ which aims to extract the style information and the dynamics of V, its architecture is almost identical to the VGG-19 network up to `relu2_1` layer, with simply modifying its first convolution layer `conv1_1` to support the input size of V (i.e. three video frames.)

There is a main modification we have in generator $\mathcal{G}$, i.e. **classwise AdaIN and NoiseAdaIN**, which significantly makes it different from the one of [32]. First, as we could observe from most of the time-lapse videos that the regions of different classes usually have different patterns of temporal variation, we thus explicitly utilize the semantic label map for performing the classwise modulation both in the training and testing stages. To be detailed, after applying the semantic segmentation on both V and I, for each region/segment in I of a specific class, its feature is modulated by the feature statistics obtained from the segments in V of the same class.

Second, instead of sequentially performing whitening & coloring transform (i.e. WCT [16]) at each scale of the encoder-decoder network as [32], we choose to use AdaIN [11] as our tool for modulation due to its simplicity and efficiency in terms of computation. In particular, we apply AdaIN twice in the decoder $D_{\mathcal{G}}$, on the layers symmetric to `relu2_1` and `relu1_1` of $E_{\mathcal{G}I}$ (denoted as `D_relu2_1` and `D_relu1_1` respectively). Please note that these two times of AdaIN are actually aimming to perform modulation on the higher-level and the lower-level features respectively.

In `D_relu2_1`, with denoting the feature map from previous layer as x and the feature map of `relu2_1` layer from $E_{\mathcal{G}V}$ as y (i.e. $E_{\mathcal{G}V}^{\texttt{relu2\_1}}(V)$), the classwise AdaIN is:

$$AdaIN(x_s, y_s) = \sigma(y_s)\left(\frac{x_s - \mu(x_s)}{\sigma(x_s)}\right) + \mu(y_s), \forall s \tag{2}$$

where the subscript s helps to indicate the regions of different object classes by referring to the corresponding semantic label maps, and μ, σ denote the mean and the standard deviation respectively. This operation is able to transfer the style from y to x by matching their statistics. Moreover, if there exists no counterpart in y for a certain class s in x, we simply use the statistics over the whole feature map, i.e. $\mu(y)$ and $\sigma(y)$, to perform the modulation on s.

For the second modulation in the `D_relu1_1` layer, since now it is dealing with the low-level features that are

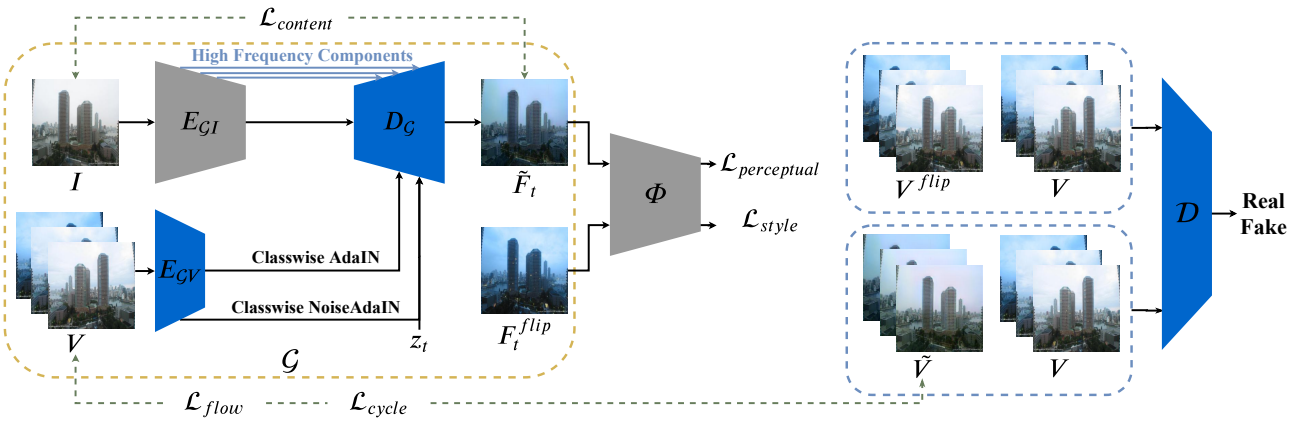

Figure 2. Architecture overview of our proposed method, together with the objectives used in the model learning. Our model contains a generator $\mathcal{G}$, a discriminator $\mathcal{D}$, and a feature extractor Φ based on a pretrained VGG-19 network [26], in which the generator $\mathcal{G}$ is composed of $E_{\mathcal{G}I}$, $E_{\mathcal{G}v}$, and $D_{\mathcal{G}}$. We shade each subnetwork in different colors, where the gray-shaded ones are pretrained and fixed, while the blue-shaded ones are learnable in our model training procedure. Our model takes an image I and a reference video clip V as input, and produces a frame $\tilde{F}_t$ which in result has the content of I and the style transferred from V. The learning of our model is based on the self-supervised scheme, where the input image I during the training stage is the mean image averaged over the whole input reference video after being horizontally-flipped. Please refer to the description in the Section 3 for more details.

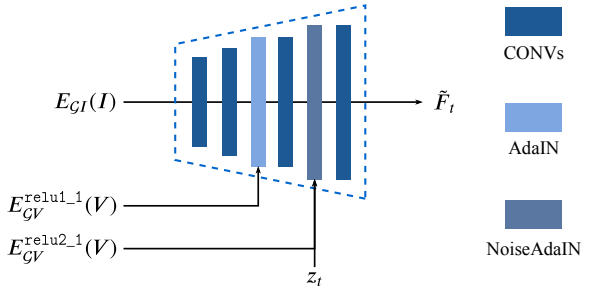

Figure 3. Details of the decoder $D_{\mathcal{G}}$. The decoder takes image features $E_{\mathcal{G}I}(I)$, video features $E_{\mathcal{G}V}^{\texttt{relu1\_1}}(V)$ and $E_{\mathcal{G}V}^{\texttt{relu2\_1}}(V)$ as inputs to perform the modulation twice: first on `D_relu2_1` by AdaIN, second on `D_relu1_1` by our proposed NoiseAdaIN with using the noise map z_t.

much closer to the final output, the typical AdaIN operation would apply the same mean and standard deviation for all the pixels in the region of the same class thus potentially causing unrealistic and monotonic appearance in each region. In order to deal with this issue, we propose the **NoiseAdaIN** technique which basically attempts to add slight noise during the procedure of AdaIN for enriching the spatial variation within each region. Given the feature map x before `D_relu1_1` and the feature map y from `relu1_1` of $E_{\mathcal{G}V}$ (i.e. $E_{\mathcal{G}V}^{\texttt{relu1\_1}}(V)$), our NoiseAdaIN is realized by:

$$NoiseAdaIN(x, y) = \sigma(y)\left(\frac{x - \mu(x)}{\sigma(x)}\right) + \mu(y)z. \quad (3)$$

where z is a noise map with its size equal to $x \in \mathbb{R}^{H\times W\times D}$. Please note that the NoiseAdaIN operation is also applied in a classwise manner, here we skip the subscript s in Eq. 3 for simplicity.

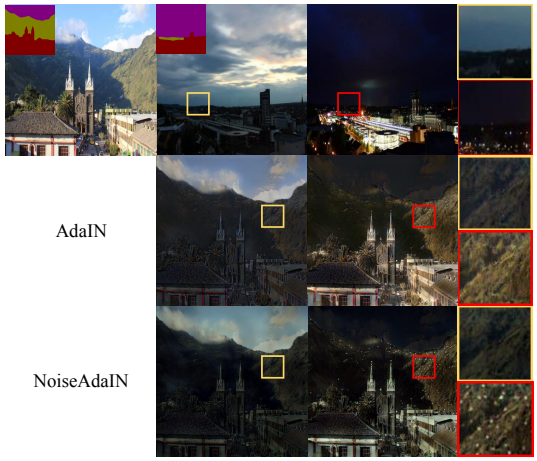

Figure 4. Comparison between AdaIN and NoiseAdaIN. The first row sequentially shows the input image and two frames in the reference video, with two regions in the reference frames highlighted on the rightmost column. The corresponding semantic label maps of input image and reference frames are shown on the top-left corner of them. The second and third rows show the results produced by using typical AdaIN and our proposed NoiseAdaIN in the modulation respectively.

Furthermore, since we would sequentially take every three frames from the reference video together with I as our input to generate time-lapse frames $\{\tilde{F}_t\}_{t=1\cdots N}$, where N is the total number of frames in the reference video, the noise map z used in each time step t should not only be spatially but also temporally smooth for avoiding the drastic variation in the local region and the flickering between generated frames. To achieve this, we first create a global noise map $Z \in \mathbb{R}^{H\times W\times(DN)}$ with each element sampled from $\mathcal{N}(1, \sigma_z)$, smooth it with a Gaussian filter of size

$3 \times 3 \times 3$, where the temporal and spatial smoothness of the noise maps are ensured, and split the smoothed Z into N tensors $\{z_1, z_2, \cdots, z_N\}$ with each at size of $H \times W \times D$. For generating $\tilde{F}_t$, we then take its corresponding z_t for the use of applying NoiseAdaIN. σ_z here decides the influence level of z, which is set to 0.5 across all our experiments. In Figure 4 we provide a comparison of applying AdaIN or NoiseAdaIN in the `D_relu1_1` layer. As we can see, there are lighted lamps in the mountain area in the reference video (first row), our proposed NoiseAdaIN (third row) is able to simulate the lighting phenomenon while AdaIN (second row) instead homogeneously brightens the entire mountain area due to its application of the same statistics on the whole region.

3.2. Training Objectives

As our problem scenario is to take a still image and a reference video to produce a time-lapse video, there is no dataset under such setting that we can directly play with. Moreover, it is also almost impossible to collect a dataset with proper groundtruth (e.g. there is no way we can find a real-world time-lapse video of the landscape in Tokyo which has the seasonal or daily changes as in Paris.)

For addressing the problem of having no proper dataset, here we propose a **self-supervised learning scheme**. Basically, we use the mean image averaged over all the frames in the reference video, i.e. $\frac{1}{N}\sum_{t=1}^{N} F_t$, as the source of our input image I in the training time. Furthermore, we explicitly flip I horizontally, as shown in Figure 2, in order to prevent our model from learning the spatial correlation between I and the reference video clip V since now they are from the same video. Based on this setting, the ideal result of the generated output $\tilde{F}_t$ is exactly the flipped F_t from V, where their difference can then be used as the objective to drive our model training. We now detail all the objective functions in the following.

Perceptual loss. As mentioned above, since I is of the same scene as the flipped reference video in the training time, the generated frame $\tilde{F}_t$ must be similar to the flipped F_t, which is denoted as F_t^{flip}. We adopt the perceptual loss $\mathcal{L}_{perceptual}$ proposed in [12] to penalize the Euclidean distance between the deep features extracted from F_t^{flip} and the ones from $\tilde{F}_t$:

$$\mathcal{L}_{perceptual} = \sum_{l} \| \Phi^l(F_t^{flip}) - \Phi^l(\tilde{F}_t) \|_2, \tag{4}$$

where $\Phi^l(\cdot)$ denotes the feature representation obtained from the l-th layer of a pretrained VGG network, basically `relu1_1`, `relu2_1`, `relu3_1`, and `relu4_1` layers are used in our implementation.

Content loss. Moreover, in order to preserve the structure of I in the generated $\tilde{F}_t$, we design the content loss $\mathcal{L}_{content}$ to encourage their content similarity on both high-frequency and low-frequency components (please note that our $E_{\mathcal{G}I}$ has the wavelet pooling operation to decompose the features into different frequency components, as described in Section 3.1). Furthermore, we also minimize the difference between the whitened I and the whitened $\tilde{F}_t$ for alleviating the influence of different color appearance.

$$\begin{aligned} \mathcal{L}_{content} =& \alpha \| E_{\mathcal{G}I}(I) - E_{\mathcal{G}I}(\tilde{F}_t) \|_2 \\ &+ (1-\alpha) \| h(I) - h(\tilde{F}_t) \|_2, \end{aligned} \tag{5}$$

where $h(\cdot)$ denotes the whitening function and α helps to balance those two terms in $\mathcal{L}_{content}$, in which α is first set to 1 and exponentially decays as training goes on.

Style loss. In addition to the content loss, here we also adopt the style loss $\mathcal{L}_{style}$ as in the original AdaIN work [11] to encourage the similarity in terms of feature statistics between $\tilde{F}_t$ and F_t^{flip}, which is written as:

$$\begin{aligned} \mathcal{L}_{style} =& \frac{1}{N_s} \sum_{s} \sum_{l} \| \mu(\Phi_s^l(F_t^{flip})) - \mu(\Phi_s^l(\tilde{F}_t)) \|_2 \\ &+ \frac{1}{N_s} \sum_{s} \sum_{l} \| \sigma(\Phi_s^l(F_t^{flip})) - \sigma(\Phi_s^l(\tilde{F}_t)) \|_2, \end{aligned} \tag{6}$$

where s represents different semantic classes in the scene and N_s denotes the total number of classes.

Flow loss. We assume that for a region of a specific class within the reference time-lapse video (captured with the static camera), its temporal variation would likely construct a path in the high-level feature space (i.e. pretrained VGG features in our method), and we denote the mean difference between VGG features obtained from the same region of two adjacent video frames as *flow*, which can be written as:

$$\mathcal{F}_s^l(F_t, F_{t+1}) = \mu(\Phi_s^l(F_{t+1}) - \Phi_s^l(F_t)). \tag{7}$$

As our model aims to transfer the pattern of temporal variation from V to I for generating $\tilde{F}$, ideally the flow of the generated video should be identical to the one of the reference video, thus the difference in flow between them ought to be penalized for encouraging the temporal consistency in the generated time-lapse. The flow loss $\mathcal{L}_{flow}$ is hence defined as:

$$\begin{aligned} \mathcal{L}_{flow} =& \frac{1}{N_s} \sum_{s} \sum_{l} | \mathcal{F}_s^l(F_t, F_{t+1}) - \mathcal{F}_s^l(\tilde{F}_t, \tilde{F}_{t+1}) | \\ &+ \frac{1}{N_s} \sum_{s} \sum_{l} | \mathcal{F}_s^l(F_1, F_t) - \mathcal{F}_s^l(\tilde{F}_1, \tilde{F}_t) |. \end{aligned} \tag{8}$$

Note here we explicitly include the flow between the first frame and the frame at time t as the second term in $\mathcal{L}_{flow}$ for keeping the long-term temporal consistency as well.

Cycle Consistency loss. We step further to design a cycle consistency loss $\mathcal{L}_{cycle}$ which helps $E_{\mathcal{G}v}$ and $D_{\mathcal{G}}$ to

precisely extract temporal variation from V and generate frames with consistent temporal changes. Basically, the adjacent generated frames $\tilde{V} = \{\tilde{F}_t, \tilde{F}_{t+1}, \tilde{F}_{t+2}\}$ should have the similar classwise temporal changes as V regardless of their difference in the content. In order to make comparison between $\tilde{V}$ and V independent of the content, the cycle consistency loss $\mathcal{L}_{cycle}$ is based on the Gram matrix of the feature maps extracted by $E_{\mathcal{G}v}$, in which it is defined as:

$$\mathcal{L}_{cycle} = \frac{1}{N_s}\sum_{s}\|G_s(E_{\mathcal{G}V}(\tilde{V})) - G_s(E_{\mathcal{G}V}(V))\|_2, \quad (9)$$

where the function G_s outputs the Gram matrix of the feature maps obtained by $E_{\mathcal{G}v}$ for a certain class s.

Video Adversarial loss. Inspired by the adversarial learning scheme [9], we design a discriminator to further improve both the temporal smoothness and the realness of the generated video. As illustrated in the right of Figure 2, the input of $\mathcal{D}$ is either a reference video clip V concatenated with horizontally flipped video V^{flip} or V concatenated with generated video clip $\tilde{V}$. The adversarial objective of $\mathcal{D}$ is formulated as:

$$\begin{aligned}\mathcal{L}_{adv} =& \mathbb{E}_{(V,V^{flip})} log\mathcal{D}(V, V^{flip}) \\ &+ \mathbb{E}_{(V,\tilde{V})} log(1 - \mathcal{D}(V, \tilde{V})),\end{aligned} \quad (10)$$

where $\mathcal{D}$ distinguishes between real videos and generated ones. Since the inputs are video clips, $\mathcal{D}$ can simultaneously judge the realness of frames and the fluency of the video.

3.3. Total Loss and Implementation Details

The overall objective of our model training is the sum of the aforementioned loss terms:

$$\begin{aligned}\mathcal{L}(\theta_{\mathcal{G}}, \theta_{\mathcal{D}}) =& \mathcal{L}_{adv} + \lambda_1\mathcal{L}_{perceptual} + \lambda_2\mathcal{L}_{content} \\ &+ \lambda_3\mathcal{L}_{style} + \lambda_4\mathcal{L}_{flow} + \lambda_5\mathcal{L}_{cycle},\end{aligned} \quad (11)$$

where $\theta_{\mathcal{G}}$ and $\theta_{\mathcal{D}}$ are the network parameters of generator $\mathcal{G}$ and discriminator $\mathcal{D}$. Note here again that in the generator $\mathcal{G}$, only $E_{\mathcal{G}V}$ and $D_{\mathcal{G}}$ are learnable while $E_{\mathcal{G}I}$ is pretrained and fixed. To control the balance of each loss function, the hyperparameters λ_1 to λ_5 are respectively set to 0.1, 0.05, 10, 100, and 0.05.

We implement our model based on the PyTorch framework. We train our model for 200 epochs with the batch-size set to 1. For $D_{\mathcal{G}}$ and the discriminator $\mathcal{D}$, they are trained from scratch with Adam optimizer and learning rate of 10^{-4}. And for $E_{\mathcal{G}V}$, its parameters from `conv1_2` to `relu2_1` layers are initialized by the ones from from ImageNet pre-trained VGG-19 network, which get finetuned in our training process by using SGD optimizer with learning rate 10^{-4}, momentum 0.9, and weight decay 10^{-4}. The semantic label maps of the training set are annotated by ourselves, while the ones used in the testing time are generated by the semantic segmentation model released in [34]. All our source code, models, and the dataset will be publicly available upon paper acceptance.

4. Experiments

Datasets. We adopt the Webcam Clip Art Dataset [13] for our model training and use the dataset of [24] as the test set. The Webcam Clip Art Dataset contains roughly $580,000$ images taken from 54 webcam sequences, totaling around $10,500$ days. However, it is time consuming to train with such long sequences. To overcome this issue, in each epoch, we randomly select 64 adjacent frames from each of the sequences as the reference videos. For the test database from [24], it contains 450 time-lapse videos, covering a wide range of landscapes and cityscapes, including city skyline, lake, and mountain view. We select 30 videos (in total $10,350$ images) with significant changes in color appearance as our reference videos in the testing time.

Baselines. Since there is no prior work of the same problem setting as ours, we compare our results with several existing style transfer methods, including the universal approaches (i.e. WCT [16] and AdaIN [11]) and the photorealistic ones (i.e. WCT^2 [32] and Li *et al*. [15]). We use the source code provided by the authors with the default parameters, and perform the transfer process in a frame-by-frame manner. Please note that both the WCT^2 [32] and Li *et al*. [15] have demonstrated the ability of their proposed algorithms in performing style transfer on videos, therefore we believe it is reasonable to have them also as competitive baselines of the photorealistic video style transfer, while other related works in video style transfer only focus on artistic styles. The semantic label maps are provided to all these baselines in order to have fair comparison.

4.1. Qualitative Results

Figure 5 shows the results from WCT [16], AdaIN [11], WCT^2 [32], Li *et al.* [15], and ours. WCT not only seriously distorts the content but also create videos with flickering artifacts. AdaIN can better preserve temporal consistency, but it still can not well maintain the structure and details. WCT^2 well keeps the details of the input image. However, there are two significant disadvantages for WCT^2: First, it causes obvious boundaries between different regions, e.g. the boundary between the tree and the buildings in the first set as well as the border between the ground and the sky in the second set; Second, it sometimes creates spotty artifacts like the second set shows. Method proposed by Li *et al*. learns a covariance-based transformation matrix to transfer the target style. However, their results often preserve too much color information of the content image. For example in the first set, the color appearance of the sea and the buildings near to it should be darker as other style transfer methods show. Furthermore, Li *et al*. rely on post-processing step to

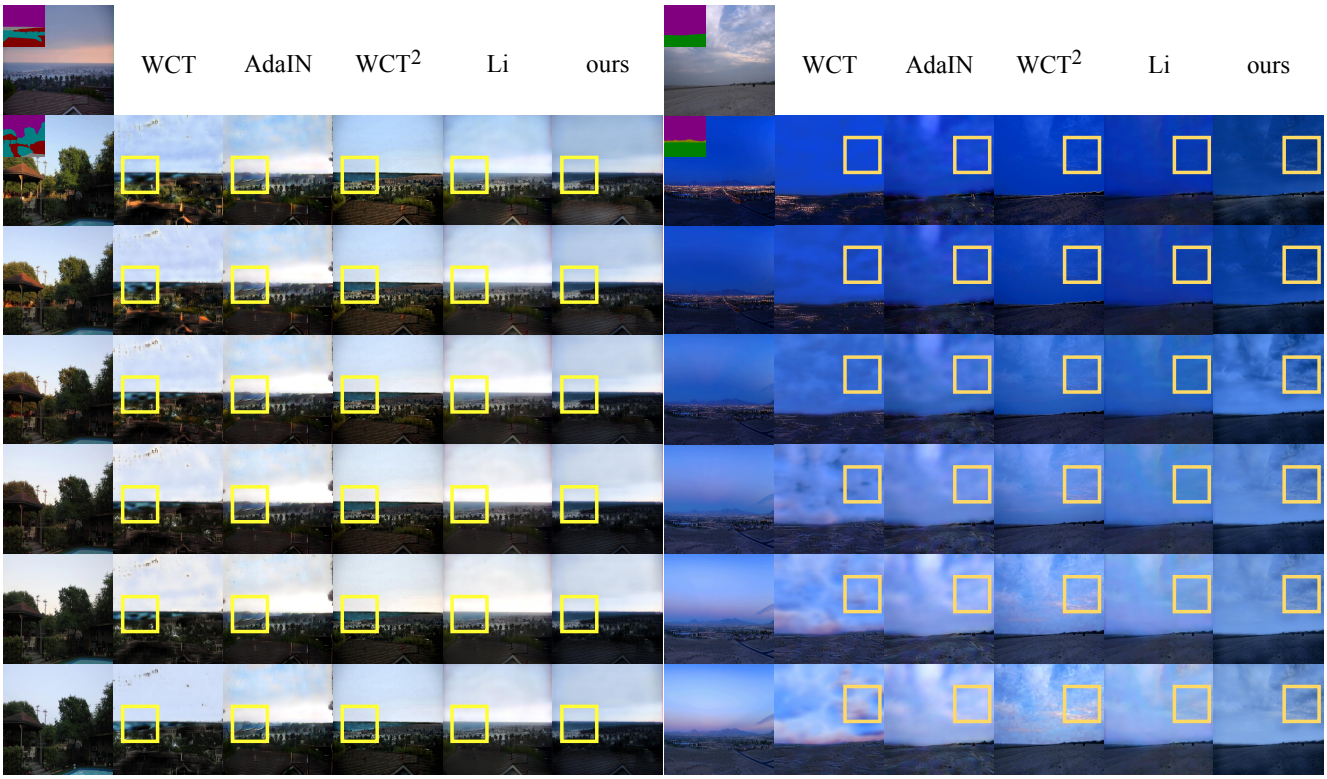

Figure 5. Two sets of example results for time-lapse video synthesis. For each set, input image, reference video frames, and their corresponding segmentation maps (on the top-left corner) are shown on the first column. The results produced by WCT [16], AdaIN [11], WCT^2 [32], Li *et al*. [15], and our proposed model are sequentially provided from the second column.

Figure 6. Example comparison of our model with respect to baselines on the robustness toward the reference frame with defects, i.e. the horizontal stripes in the sky region. Our result is less affected while the baselines have visible artifacts.

generate harmonious and photorealistic results but it leads to blurry artifacts, as observable on the sky region in the second set. In contrast, our model successfully learns the features of temporal variation from the adjacent frames of reference video and synthesizes realistic time-lapse frames without utilizing any post-processing operation to preserve the details of the input image. Besides, our method is able to work on arbitrary input image and reference videos. More examples will be provided in the supplementary material.

Moreover, since our model extracts features from multiple adjacent frames of the reference video, it is robust to deal with the reference frame with defects. As shown in Figure 6, the reference frame F_t suddenly corrupts due to horizontal stripes in the sky (please zoom in for better visualization), which leads to visible artifacts on the results from baselines. On the contrary, our result is almost unaffected.

4.2. Quantitative Results

Photorealistic stylization. To measure the photorealism, we adopt two metrics for spatial recovery and stylization effectiveness. In the evaluation of spatial recovery, we calculate the structural similarity (SSIM) between input image and generated frames. As for stylization, we report the difference in VGG Gram matrices between the outputs of each model and the corresponding reference frame. Figure 7 shows SSIM (X-axis) versus style loss (Y-axis). The ideal photorealistic result should simultaneously have the similar structure as the input image and match the style of reference video frames, in which it corresponds to the top-right corner of Figure 7. As we can see, artistic style transfer

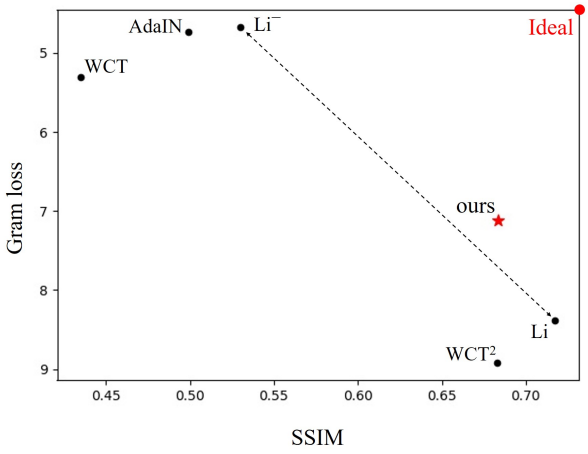

Figure 7. SSIM (higher is better) versus Gram loss (lower is better). Li$^-$ represents [15] without post-processing step. Dotted line depicts the gap between before and after post-processing step in the approach of Li *et al*. Our result is marked as red star.

	WCT	AdaIN	WCT^2	Li	ours
photorealism			✓	✓	✓
E_{tmp}	1.999	0.189	0.093	0.100	0.113
σ_{tmp}	1.641	0.199	0.234	0.235	0.227

Table 1. Evaluation of temporal consistency on different methods.

methods (i.e. WCT and AdaIN) get higher scores in stylization but are not able to well preserve the structure. WCT^2 successfully maintains the input content to get higher SSIM but has less consistency to the style of reference frames which leads to higher Gram loss. By comparing the gap between before and after post-processing step, we can obviously observe that the baseline of Li *et al*. reaches high image quality mainly by the post-processing step instead of the stylization network. However, as we can see in Figure 5, post-processing step could overly smooth the final results. On the contrary, our proposed method outperforms baselines without any post-processing step.

Temporal consistency. We compare temporal consistencies of different methods via E_{tmp} [10], which evaluates temporal error by averaged pixel-wise Euclidean difference in color between consecutive frames over a video sequence:

$$E_{tmp} = \beta \sqrt{\frac{1}{(N-1) \times D} \sum_{t=1}^{N-1} (\tilde{F}_t - \tilde{F}_{t+1})^2}, \quad (12)$$

where N denotes the total number of frames, D is the number of pixels in a frame, and β is a constant, which is set to 10^3 here. To further compare the flow of the generated video and the one of reference video, we evaluate the difference in standard deviation between the reference flow and the generated flow in pixel level:

$$\sigma_{tmp} = \frac{\beta}{(N-1)} \sum_{t=1}^{N-1} |\sigma(F_{t+1} - F_t) - \sigma(\tilde{F}_{t+1} - \tilde{F}_t)|. \quad (13)$$

	WCT	AdaIN	WCT^2	Li	ours
photorealism			✓	✓	✓
64×64	0.59	27.66	0.41	3.82	60.87
128×128	0.43	26.13	0.40	3.59	31.77
256×256	0.28	19.80	0.36	2.73	11.19
512×512	0.11	11.19	0.33	1.82	2.23
1024×1024	0.03	4.35	0.26	0.73	0.58

Table 2. Runtime performance. The result is measured in FPS on a single 2080 Ti GPU. Our approach is faster than other photorealism methods in most of the cases.

Table 1 shows the evaluation of temporal consistency in terms of E_{tmp} and σ_{tmp} (both lower the better). As we can see, artistic methods are more prone to having flicker videos with flickering artifacts (thus higher in E_{tmp}) while all the photorealistic methods can generate temporally smooth videos. When we further compare σ_{tmp} across photorealistic methods, it shows that our model has the best results due to our novel design of temporal objectives, e.g. $\mathcal{L}_{flow}$.

Runtime comparison. Table 2 shows the runtime comparison of baselines and ours. The reported runtime for each model is an average of 10 video generation, 640 frames in total, on a single 2080 Ti GPU. The results are measured in FPS. As the table shows, our method can reach real-time generation in low resolution, achieving maximum about 148 times faster than other methods. In 256×256 resolution, AdaIN is able to complete real-time transfer. However, the generated image is not exquisite enough. Among photorealism methods, our algorithm remarkably outperforms others and reaches the same order in FPS as AdaIN. In high resolution generation, existing methods are unable to achieve real-time effectiveness, while our method can still reach the same order as the state-of-the-art photorealistic algorithm.

5. Conclusion

In this paper, we propose an end-to-end time-lapse synthesis network with taking a single image and a reference video as inputs. The overall model training is driven by our self-supervised learning scheme. The experimental results show that our model is both efficient and effective, which transfers various temporal changes from the reference video into the generated time-lapse video. In comparison to baselines of state-of-the-art style transfer methods, our model can simultaneously keep the details of the input image and obtain better temporal consistency. Moreover, as currently the moving objects are not taken into consideration for our model, we would step forward to address that in the future work.

Acknowledgement This project is supported by MOST-109-2636-E-009-018, MOST-109-2634-F-009-020, and MOST-109-2634-F-009-015. We are grateful to the National Center for High performance Computing for computer time and facilities.

References

[1] Dongdong Chen, Jing Liao, Lu Yuan, Nenghai Yu, and Gang Hua. Coherent online video style transfer. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[2] Dongdong Chen, Lu Yuan, Jing Liao, Nenghai Yu, and Gang Hua. Stylebank: An explicit representation for neural image style transfer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[3] Tian Qi Chen and Mark Schmidt. Fast patch-based style transfer of arbitrary style. In *NIPS Workshop in Constructive Machine Learning*, 2016.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[5] Oriel Frigo, Neus Sabater, Julie Delon, and Pierre Hellier. Split and match: Example-based adaptive patch sampling for unsupervised style transfer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[6] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[7] Leon A. Gatys, Alexander S. Ecker, Matthias Bethge, Aaron Hertzmann, and Eli Shechtman. Controlling perceptual factors in neural style transfer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[8] Golnaz Ghiasi, Honglak Lee, Manjunath Kudlur, Vincent Dumoulin, and Jonathon Shlens. Exploring the structure of a real-time, arbitrary neural artistic stylization network. *ArXiv:1705.06830*, 2017.

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[10] Haozhi Huang, Hao Wang, Wenhan Luo, Lin Ma, Wenhao Jiang, Xiaolong Zhu, Zhifeng Li, and Wei Liu. Real-time neural style transfer for videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[11] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[12] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision (ECCV)*, 2016.

[13] Jean-François Lalonde, Alexei A. Efros, and Srinivasa G. Narasimhan. Webcam clip art: Appearance and illuminant transfer from time-lapse sequences. *ACM Transactions on Graphics (TOG)*, 2009.

[14] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *European Conference on Computer Vision (ECCV)*, 2016.

[15] Xueting Li, Sifei Liu, Jan Kautz, and Ming-Hsuan Yang. Learning linear transformations for fast arbitrary style transfer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[16] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal style transfer via feature transforms. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[17] Yijun Li, Ming-Yu Liu, Xueting Li, Ming-Hsuan Yang, and Jan Kautz. A closed-form solution to photorealistic image stylization. In *European Conference on Computer Vision (ECCV)*, 2018.

[18] Fujun Luan, Sylvain Paris, Eli Shechtman, and Kavita Bala. Deep photo style transfer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[19] Seonghyeon Nam, Chongyang Ma, Menglei Chai, William Brendel, Ning Xu, and Seon Joo Kim. End-to-end time-lapse video synthesis from a single outdoor image. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[20] Eric Risser, Pierre Wilmot, and Connelly Barnes. Stable and controllable neural texture synthesis and style transfer using histogram losses. *ArXiv:1701.08893*, 2017.

[21] Manuel Ruder, Alexey Dosovitskiy, and Thomas Brox. Artistic style transfer for videos. In *German Conference on Pattern Recognition (GCPR)*, 2016.

[22] Lu Sheng, Ziyi Lin, Jing Shao, and Xiaogang Wang. Avatar-net: Multi-scale zero-shot style transfer by feature decoration. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[23] YiChang Shih, Sylvain Paris, Connelly Barnes, William T. Freeman, and Frédo Durand. Style transfer for headshot portraits. *ACM Transactions on Graphics (TOG)*, 2014.

[24] Yichang Shih, Sylvain Paris, Frédo Durand, and William T Freeman. Data-driven hallucination of different times of day from a single outdoor photo. *ACM Transactions on Graphics (TOG)*, 2013.

[25] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. Animating arbitrary objects via deep motion transfer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[26] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *ArXiv:1409.1556*, 2014.

[27] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[28] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor S Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *International Conference on Machine Learning (ICML)*, 2016.

[29] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[30] Xin Wang, Geoffrey Oxholm, Da Zhang, and Yuan-Fang Wang. Multimodal transfer: A hierarchical deep convolutional neural network for fast artistic style transfer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[31] Wei Xiong, Wenhan Luo, Lin Ma, Wei Liu, and Jiebo Luo. Learning to generate time-lapse videos using multi-stage dynamic generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[32] Jaejun Yoo, Youngjung Uh, Sanghyuk Chun, Byeongkyu Kang, and Jung-Woo Ha. Photorealistic style transfer via wavelet transforms. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[33] Hang Zhang and Kristin Dana. Multi-style generative network for real-time transfer. In *European Conference on Computer Vision Workshops*, 2018.

[34] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision (IJCV)*, 2018.

[35] Yipin Zhou and Tamara L Berg. Learning temporal transformations from time-lapse videos. In *European Conference on Computer Vision (ECCV)*, 2016.