End-to-End Time-Lapse Video Synthesis from a Single Outdoor Image

Seonghyeon Nam[1] Chongyang Ma[2] Menglei Chai[2]
William Brendel[2] Ning Xu[3] Seon Joo Kim[1]
[1]Yonsei University [2]Snap Inc. [3]Amazon Go

Abstract

Time-lapse videos usually contain visually appealing content but are often difficult and costly to create. In this paper, we present an end-to-end solution to synthesize a time-lapse video from a single outdoor image using deep neural networks. Our key idea is to train a conditional generative adversarial network based on existing datasets of time-lapse videos and image sequences. We propose a multi-frame joint conditional generation framework to effectively learn the correlation between the illumination change of an outdoor scene and the time of the day. We further present a multi-domain training scheme for robust training of our generative models from two datasets with different distributions and missing timestamp labels. Compared to alternative time-lapse video synthesis algorithms, our method uses the timestamp as the control variable and does not require a reference video to guide the synthesis of the final output. We conduct ablation studies to validate our algorithm and compare with state-of-the-art techniques both qualitatively and quantitatively.

1. Introduction

Time-lapse videos are typically created by using a fixed or slowly moving camera to capture an outdoor scene at a large frame interval. This unique kind of videos is visually appealing since it often presents drastic color tone changes and fast motions, which show the passage of time. But time-lapse videos usually require a sophisticated hardware setup and are time-consuming to capture and edit. Therefore, it is desirable and helpful to design and develop a system to facilitate the creation of time-lapse videos.

The appearance of an outdoor scene depends on many complicated factors including weather, season, time of day, and objects in the scene. As a result, most time-lapse videos present highly nonlinear changes in both the temporal and spatial domains, and it is difficult to derive an explicit model to synthesize realistic time-lapse videos while taking all the deciding factors into account accurately.

Figure 1: For each single outdoor image (the first column), we can predict continuous illuminance changes over time in an end-to-end manner (four columns on the right).

With various emerging social network services, a large amount of time-lapse video data that is captured at different locations around the world has become accessible on the Internet. Therefore, a natural idea for generating time-lapse videos is to automatically synthesize the animation output by learning from a large-scale video database. A data-driven hallucination algorithm [27] was proposed to synthesize a time-lapse video from an input image via a color transfer based on a reference video retrieved from a database. However, this framework needs to store the entire database of time-lapse videos for retrieval at runtime. Also, it may not always be possible to find a reference video that has components semantically similar to the input image for a visually plausible color transfer. Recent advances in computer vision and machine learning have shown that deep neural networks can be used to achieve photorealistic style transfer [22, 19, 18] and to synthesize high-fidelity video sequences [16, 34, 2]. Yet most existing deep video generation techniques require a reference video or a label map sequence to guide the synthesis of the output video.

In this work, we present an end-to-end data-driven time-lapse hallucination solution for a single image without the requirement of any semantic labels or reference videos at runtime. Given an outdoor image as the input, our method

can automatically predict how the same scene will look like at different times of a day and generate a time-lapse video with continuous and photorealistic illumination changes by using the timestamp as the control variable. See Figure 1 for some example results generated by our system.

Conventionally, video generation tasks have been modelled by spatiotemporal methods such as recurrent neural networks and volumetric convolutions [35, 32, 30, 42]. However, it is challenging to achieve our goal with these approaches since the raw footage of existing time-lapse datasets [12, 27] contains a number of unwanted camera motions, moving objects, or even corrupted frames, which aggravates the quality of output sequences. In this work, we cast our task as a conditional image-to-image translation task using the timestamp as the control variable, which enables our learning to be robust to such outliers through the structure preserving property [43, 11]. However, this alone cannot generate plausible time-lapse videos due to the independent modeling of different times. To effectively train the continuous change of illumination over time, we propose a multi-frame joint conditional generation framework (Section 3.1). For training, we leverage the AMOS dataset [12] and build a large collection of outdoor images with the corresponding timestamps of when the photos were taken.

One issue of using the AMOS dataset is that many footages in the dataset are visually uninteresting, because the dataset is collected from hundreds of thousands of surveillance cameras capturing outdoor scenes such as highways and landscapes. To further improve the visual quality of our synthesis output, we also leverage the time-lapse video database TLVDB [27], which is a small collection of time-lapse videos. The videos in the TLVDB dataset present rich illumination changes but do not have the ground-truth timestamp for each frame. To jointly learn from both the TLVDB dataset and the AMOS dataset, we propose a multi-domain training scheme (Section 3.2) based on image domain translation [43, 11]. It enables the TLVDB dataset to be trained with our conditional generation framework in a *semi-supervised* manner, which removes the necessity for timestamps in the TLVDB dataset. Our training scheme also effectively handles the difference of data distribution between the two datasets and makes the training process more stable compared to a naïve implementation.

We show a variety of time-lapse video synthesis results on diverse input images and compare our method with alternative approaches (Section 4). We also verify the design and implementation of our framework via extensive ablation studies and evaluations. In summary, our contributions are:

- We present the first solution for synthesizing a time-lapse video with continuous and photorealistic illumination changes from a single outdoor image without the requirement of any reference video at runtime.
- We propose a multi-frame joint conditional network to learn the distributions of color tones at different times of a day while bypassing the motions and outliers in the training data.
- We propose a multi-domain training scheme for stable semi-supervised learning from different datasets to further improve the visual quality of synthesis output.

2. Related Work

Image and video stylization. Image and video stylization has been an active research area over the past few years, especially with the recent advances in deep neural networks for robust and effective computation of visual features [7, 14, 43, 22, 19, 18]. A typical usage scenario of visual stylization algorithms is to transfer the style of the input from one *source* domain into another *target* domain while keeping the content, such as night to day, sketch to photo, label map to image, or vice versa [15, 38, 11, 4, 34, 2, 1]. In contrast to these prior methods, our technique aims to change the illumination of an input image in a *continuous* manner by using time of day as the control variable for a conditional generative model.

Animating still images. Creating animation from a single image has been a longstanding research problem in computer vision and computer graphics. Early work on this topic relies on either user interactions [5] or domain-specific knowledge [36, 13]. Most related to our approach, a data-driven hallucination method [27] was proposed to synthesize a time-lapse video from a single outdoor image by a color transfer algorithm based on a reference video. On the contrary, we only need to store a compact model for the synthesis and do not require any reference video at runtime. Therefore our method requires much less storage and can be more robust for input images that are significantly different from all the available reference videos.

More recently, deep neural networks such as generative adversarial networks (GANs) and variational autoencoders (VAEs) have been widely used for video synthesis and future frame prediction [37, 33, 32, 31, 30]. Due to the limited capability of neural networks, most of these techniques can only generate very short or fixed-length sequences with limited resolution, and/or have been focusing on specific target phenomenon, such as object transformation [42] and cloud motions [35]. Our approach is complementary to these prior methods, and we can animate a variety of high-resolution outdoor images by continuously changing the color tone to generate output videos of arbitrary length.

Learning from video dataset. Compared to traditional image datasets such as ImageNet [6] and COCO [21], large-scale video datasets (or image sequences from static

cameras) usually contain rich hidden information among the coherent frames within each sequence. On the other hand, these datasets present additional challenges to the learning algorithms since the amount of data is usually prohibitively large and less structured. The Archive of Many Outdoor Scenes (AMOS) dataset [12] contains millions of outdoor images captured with hundreds of webcams. In their work, the authors demonstrate the possibility of analyzing the dataset with automatic annotations, such as semantic labels, season changes, and weather conditions. It is also possible to extract illumination, material and geometry information from time-lapse videos as shown in previous methods [23, 29, 17]. Most recently, Li and Snavely [20] proposed to learn single-view intrinsic image decomposition from time-lapse videos in the wild without ground truth data. We draw inspirations from this line of research and propose to learn a generative model for the time-lapse video synthesis.

3. Our Method

Problem statement. To synthesize a time-lapse video from a single input image, we define our task as conditional image translation based on generative adversarial networks (GANs) [8, 24] by using the time of day as the conditional variable. Formally, let $\mathbf{I}$ be an input image and $t \in \mathbb{R}$ be the target timestamp variable in the range of $[0,\ 1)$ for a whole day. Our task can then be described as $\hat{\mathbf{I}}_t = \mathcal{G}(\mathbf{I}, t)$ where a generator $\mathcal{G}$ hallucinates the color tone of the input $\mathbf{I}$ to predict the output image $\hat{\mathbf{I}}_t$ at the time t. To generate a time-lapse video, we sample a finite set of timestamps $t \in \{t_0, t_1, t_2, \cdots, t_n\}$, then aggregate generated images to form a video $\mathbf{V} = \{\hat{\mathbf{I}}_0, \hat{\mathbf{I}}_1, \hat{\mathbf{I}}_2, \cdots, \hat{\mathbf{I}}_n\}$.

Note that our goal is to model continuous and nonlinear change of color tone over the time without considering dynamic motions such as moving objects. At test time, we synthesize output video in an end-to-end manner without the requirement of any reference video, scene classification, or semantic segmentation. In addition, our approach enables generating any number of frames at inference time by using real-valued t as the control variable.

Datasets. To learn an end-to-end model for the time-lapse video synthesis from a single input image, we leverage the AMOS dataset [12] and the time-lapse video database (TLVDB) [27]. The AMOS dataset is a large-scale dataset of outdoor image sequences captured from over $35,000$ webcams around the world. A typical sequence in the AMOS dataset has tens of frames for every 24 hours with timestamp labels of the time when they were captured. The TLVDB dataset contains 463 real time-lapse videos and most of them are about landmark scenes. Each of the videos in the TLVDB dataset has at least hundreds of frames without timestamp labels and the numbers of frames are different. Figure 2 shows several sample frames from the two datasets.

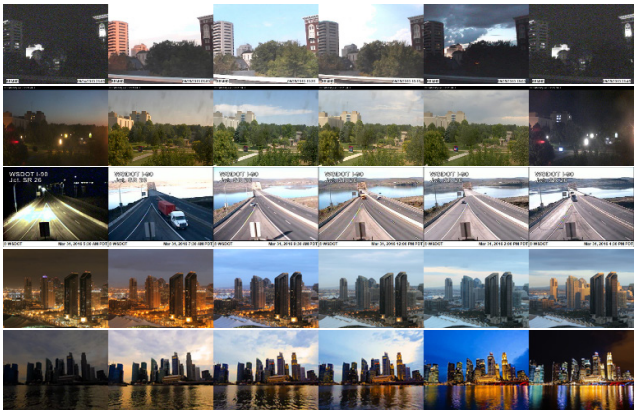

Figure 2: Sample sequences from the AMOS (top three rows) and the TLVDB (bottom two rows) datasets.

However, it is not easy to directly train a generative model using these two datasets since they contain many outliers, or even corrupted data. For instance, the images in some sequences are not aligned due to camera movement and contain abrupt scene changes, text overlays, fade in/out effects, etc. We only prune some obviously corrupted frames and sequences manually, as removing all the noisy data requires extensive human labor with heuristics.

3.1. Multi-Frame Joint Conditional Generation

We denote the AMOS dataset as $\mathcal{A}$. Each data $(\mathbf{I}_i, t_i)$ in $\mathcal{A}$ is a pair of an image $\mathbf{I}_i$ and its corresponding timestamp t_i. As a naïve approach, it is possible to adopt a conditional *image-to-image* translation framework using the timestamp t as the conditional variable. Specifically, from the AMOS dataset $\mathcal{A}$, we may train a generator $\mathcal{G}$ to synthesize an image $\hat{\mathbf{I}}_t = \mathcal{G}(\mathbf{I}, t)$ for a target timestamp t, while a discriminator $\mathcal{D}$ is trained to distinguish whether a pair of an image and a timestamp is real or fake.

However, we found that training such a naïve model using each frame in $\mathcal{A}$ independently tends to generate implausible color tone in the output sequence. This is because the illumination at certain time t is correlated with the illumination at different times of the same day. In addition, the illumination changes over multiple days can be different due to additional factors such as locations, season, and weather. To this end, we propose to train multiple frames from each sequence with a shared latent variable $\mathbf{z}$. We use this latent variable $\mathbf{z}$ as the temporal context of each sequence to learn the joint distribution of color tones at different times of the day.

Generator. Our generator $\mathcal{G}$ leverages a typical encoder-decoder architecture based on our proposed multi-frame joint generation scheme, as depicted in Figure 3(left). Let $\mathbf{S}_{\mathcal{A}}$ be a

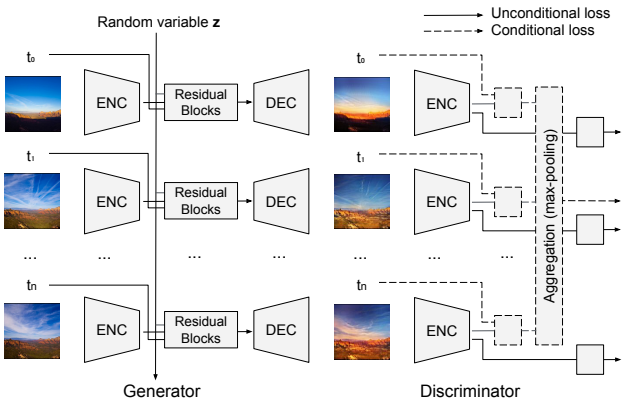

Figure 3: Illustration of our multi-frame joint conditional GAN method. For our discriminator, the encoded images are concatenated with the timestamps (dashed rectangles) before being aggregated to compute the conditional loss, while each image is directly used as an input to compute the unconditional loss (solid rectangles).

set of frames sampled from the same sequence in the AMOS dataset $\mathcal{A}$:

$$\mathbf{S}_{\mathcal{A}} = \{(\mathbf{I}_0, t_0), (\mathbf{I}_1, t_1), (\mathbf{I}_2, t_2), \cdots, (\mathbf{I}_n, t_n)\}. \quad (1)$$

An input image $\mathbf{I}_i$ is encoded by the encoder $\mathbf{E}_{\mathcal{G}}$. The shared latent variable $\mathbf{z}$ is sampled from the standard normal distribution $\mathcal{N}(0, 1)$ to represent the temporal context of the sequence from which $\mathbf{S}_{\mathcal{A}}$ is sampled. Then several residual blocks [9] take the encoded image $\mathbf{E}_{\mathcal{G}}(\mathbf{I}_i)$ together with the latent variable $\mathbf{z}$ and the timestamp t_i as the input and generate output features with a new color tone. Finally, the decoder $\mathbf{D}_{\mathcal{G}}$ in $\mathcal{G}$ decodes the feature from the residual blocks into an image $\hat{\mathbf{I}}_{t_i}$ as the reconstructed output of $\mathbf{I}_i$:

$$\begin{aligned} \hat{\mathbf{I}}_{t_i} &= \mathcal{G}(\mathbf{I}_i, t_i, \mathbf{z}) \\ &= \mathbf{D}_{\mathcal{G}}(\mathbf{E}_{\mathcal{G}}(\mathbf{I}_i), t_i, \mathbf{z}), \end{aligned} \quad (2)$$

where we omit the residual blocks for simplicity. The entire reconstructed output of $\mathbf{S}_{\mathcal{A}}$ consists of all the generated frames:

$$\hat{\mathbf{S}}_{\mathcal{A}} = \{\hat{\mathbf{I}}_{t_0}, \hat{\mathbf{I}}_{t_1}, \hat{\mathbf{I}}_{t_2}, \cdots, \hat{\mathbf{I}}_{t_n}\}. \quad (3)$$

During the training, we use different input images from the same sequence as depicted in Figure 3, which enables $\mathcal{G}$ to ignore moving factors. At the inference time, we use the same input image multiple times to get an output sequence.

Discriminator. Our discriminator $\mathcal{D}$ is divided into two parts, i.e., an *unconditional* discriminator $\mathcal{D}_u$ for each individual output image $\hat{\mathbf{I}}$ and a *conditional* discriminator $\mathcal{D}_c$ for the set of the reconstructed images from an input frame set $\mathbf{S}_{\mathcal{A}}$. The unconditional discriminator $\mathcal{D}_u$ is used to differentiate if each individual image is real or fake. The conditional discriminator $\mathcal{D}_c$ distinguishes if a generated frame set $\hat{\mathbf{S}}_{\mathcal{A}}$ is a real time-lapse sequence. In other words, $\mathcal{D}_c$ checks not only whether each individual frame $\hat{\mathbf{I}}_{t_i}$ matches the corresponding t_i, but also whether $\hat{\mathbf{S}}_{\mathcal{A}}$ presents realistic color tone changes over time.

We train both $\mathcal{D}_u$ and $\mathcal{D}_c$ based on the same image encoder $\mathbf{E}_{\mathcal{D}}$, as shown in Figure 3(right). For the conditional discriminator $\mathcal{D}_c$, the encoded image and the corresponding timestamp are concatenated for each frame and all the frames from the same sequence are aggregated to compute the discriminator score. Since the input of the conditional discriminator $\mathcal{D}_c$ is an unordered set of $\{(\hat{\mathbf{I}}_{t_i}, t_i)\}$ rather than an ordered sequence, the discriminator score should be permutation-invariant [39]. Therefore, we use *max-pooling* to aggregate encoded features of multiple frames.

Adversarial losses. The adversarial losses of our multi-frame joint conditional generation algorithm consist of an unconditional loss and a conditional loss. The unconditional adversarial loss l_u can be formally described as:

$$\begin{aligned} l_u = &\mathbb{E}_{\mathbf{I}\sim\mathcal{A}}[\log \mathcal{D}_u(\mathbf{I})] \\ &+ \mathbb{E}_{(\mathbf{I},t)\sim\mathcal{A},\mathbf{z}\sim\mathcal{N}}[1 - \log \mathcal{D}_u(\mathcal{G}(\mathbf{I}, t, \mathbf{z}))], \end{aligned} \quad (4)$$

where $\mathcal{N}$ is the standard normal distribution. Our conditional adversarial loss l_c is defined as below:

$$\begin{aligned} l_c = &\mathbb{E}_{\mathbf{S}_{\mathcal{A}}\sim\mathcal{A}}[\log \mathcal{D}_c(\mathbf{S}_{\mathcal{A}})] \\ &+ \mathbb{E}_{\tilde{\mathbf{S}}_{\mathcal{A}}\sim\mathcal{A}}[1 - \log \mathcal{D}_c(\tilde{\mathbf{S}}_{\mathcal{A}})] \\ &+ \mathbb{E}_{\mathbf{S}_{\mathcal{A}}\sim\mathcal{A},\mathbf{z}\sim\mathcal{N}}[1 - \log \mathcal{D}_c(\mathcal{G}(\mathbf{S}_{\mathcal{A}}, \mathbf{z}))]. \end{aligned} \quad (5)$$

To effectively train the correlation between the input image $\mathbf{I}_i$ and its corresponding timestamp t_i, we introduce an additional term as shown in the second row of Eq. (5) for a set of negative pairs $\tilde{\mathbf{S}}_{\mathcal{A}}$, which we collect by randomly sampling mismatched pairs from $\mathbf{S}_{\mathcal{A}}$:

$$\tilde{\mathbf{S}}_{\mathcal{A}} = \{(\mathbf{I}_i, t_j) \mid i \neq j\}. \quad (6)$$

3.2. Multi-Domain Training

Our multi-frame joint conditional generation method effectively captures diverse illumination variation over time. However, the model trained based on the AMOS dataset alone tends to generate uninteresting outputs such as clipped and less saturated colors especially in the sky region, since most footages of the AMOS dataset were captured by surveillance cameras. To further improve the visual quality of synthesis output, we propose to additionally leverage the TLVDB dataset [27] and denote it as $\mathcal{B}$. Most videos in $\mathcal{B}$ are about landmark scenes captured using professional cameras and thus present much more interesting color tone distributions and changes over time.

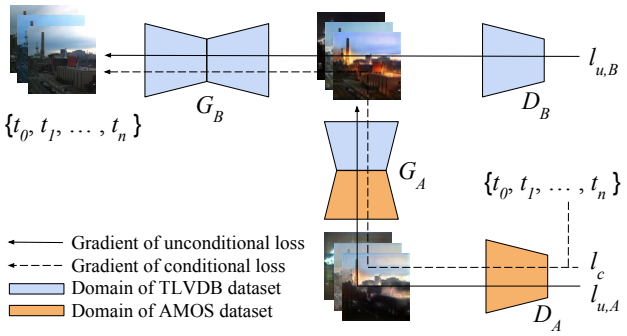

Figure 4: Illustration of our multi-domain training scheme.

However, the footages in the TLVDB dataset are videos without any ground-truth timestamp label for each frame. Therefore, it is infeasible to directly learn from this dataset using our conditional generation method described in Section 3.1. Furthermore, we have found that simply merging the AMOS dataset and the TVLDB dataset to train the unconditional image discriminator in Eq. (4) does not improve the results, due to the domain discrepancy of the two datasets. To handle the issues of missing timestamps and inconsistent data distributions, we propose a multi-domain training method.

Our key idea is to synthesize time-lapse sequences using the TLVDB dataset $\mathcal{B}$ and to learn continuous illumination changes over time from the AMOS dataset $\mathcal{A}$. Figure 4 shows the overview of our multi-domain training algorithm. Basically, we train a generator $\mathcal{G}_\mathcal{B}$ together with a discriminator $\mathcal{D}_\mathcal{B}$ based on $\mathcal{B}$ to synthesize time-lapse sequences. The synthesis results are then translated into the domain of $\mathcal{A}$ by using another generator $\mathcal{G}_\mathcal{A}$ as a proxy to get conditional training signals from the discriminator $\mathcal{D}_\mathcal{A}$ trained on $\mathcal{A}$. With the training signal from $\mathcal{D}_\mathcal{A}$, $\mathcal{G}_\mathcal{B}$ can be trained to synthesize images with the timestamp t being taken into account. To this end, we adopt our multi-frame conditional generation network (Section 3.1 and Figure 3) for $\mathcal{G}_\mathcal{B}$ and $\mathcal{D}_\mathcal{A}$, while using vanilla DCGAN [25] and U-Net [26] for $\mathcal{D}_\mathcal{B}$ and $\mathcal{G}_\mathcal{A}$, respectively.

Loss functions. For our multi-domain training scheme, the unconditional loss in Eq. (4) is reformulated as:

$$l_u = l_{u,\mathcal{A}} + l_{u,\mathcal{B}}, \tag{7}$$

where

$$\begin{aligned} l_{u,\mathcal{A}} &= \mathbb{E}_{\mathbf{I}\sim\mathcal{A}}[\log \mathcal{D}_\mathcal{A}(\mathbf{I})] \\ &\quad + \mathbb{E}_{\mathbf{I}\sim\mathcal{B},t\sim\mathcal{A},\mathbf{z}\sim\mathcal{N}}[1 - \log \mathcal{D}_\mathcal{A}(\mathcal{G}_\mathcal{A}(\mathcal{G}_\mathcal{B}(\mathbf{I},t,\mathbf{z})))], \end{aligned} \tag{8}$$

and

$$\begin{aligned} l_{u,\mathcal{B}} &= \mathbb{E}_{\mathbf{I}\sim\mathcal{B}}[\log \mathcal{D}_\mathcal{B}(\mathbf{I})] \\ &\quad + \mathbb{E}_{\mathbf{I}\sim\mathcal{B},t\sim\mathcal{A},\mathbf{z}\sim\mathcal{N}}[1 - \log \mathcal{D}_\mathcal{B}(\mathcal{G}_\mathcal{B}(\mathbf{I},t,\mathbf{z}))]. \end{aligned} \tag{9}$$

Algorithm 1 Our training algorithm

Set the learning rate η
Initialize network parameters $\theta_{\mathcal{G}_\mathcal{A}}, \theta_{\mathcal{D}_\mathcal{A}}, \theta_{\mathcal{G}_\mathcal{B}}, \theta_{\mathcal{D}_\mathcal{B}}$
for the number of iterations **do**
  Sample $\mathbf{S}_\mathcal{A}, \tilde{\mathbf{S}}_\mathcal{A} \sim \mathcal{A}, \mathbf{S}_\mathcal{B} \sim \mathcal{B}$
  (1) Update the discriminators $\mathcal{D}_\mathcal{A}$ and $\mathcal{D}_\mathcal{B}$
  Sample $\mathbf{z} \sim \mathcal{N}$
  Generate $\mathcal{G}_\mathcal{B}(\mathbf{S}_\mathcal{B}, \mathbf{z})$, $\mathcal{G}_\mathcal{A}(\mathcal{G}_\mathcal{B}(\mathbf{S}_\mathcal{B}, \mathbf{z}))$
  $\theta_{\mathcal{D}_\mathcal{A}} = \theta_{\mathcal{D}_\mathcal{A}} + \eta\nabla_{\theta_{\mathcal{D}_\mathcal{A}}}(l_{u,\mathcal{A}} + l_c)$
  $\theta_{\mathcal{D}_\mathcal{B}} = \theta_{\mathcal{D}_\mathcal{B}} + \eta\nabla_{\theta_{\mathcal{D}_\mathcal{B}}} l_{u,\mathcal{B}}$
  (2) Update the generator $\mathcal{G}_\mathcal{A}$
  Sample $\mathbf{z} \sim \mathcal{N}$
  Generate $\mathcal{G}_\mathcal{B}(\mathbf{S}_\mathcal{B}, \mathbf{z})$, $\mathcal{G}_\mathcal{A}(\mathcal{G}_\mathcal{B}(\mathbf{S}_\mathcal{B}, \mathbf{z}))$
  $\theta_{\mathcal{G}_\mathcal{A}} = \theta_{\mathcal{G}_\mathcal{A}} + \eta\nabla_{\theta_{\mathcal{G}_\mathcal{A}}}(l_{u,\mathcal{A}} + l_r)$
  (3) Update the generator $\mathcal{G}_\mathcal{B}$
  Sample $\mathbf{z} \sim \mathcal{N}$
  Generate $\mathcal{G}_\mathcal{B}(\mathbf{S}_\mathcal{B}, \mathbf{z})$, $\mathcal{G}_\mathcal{A}(\mathcal{G}_\mathcal{B}(\mathbf{S}_\mathcal{B}, \mathbf{z}))$
  $\theta_{\mathcal{G}_\mathcal{B}} = \theta_{\mathcal{G}_\mathcal{B}} + \eta\nabla_{\theta_{\mathcal{G}_\mathcal{B}}}(l_{u,\mathcal{B}} + l_c)$
end for

The conditional loss in Eq. (5) can be rewritten as:

$$\begin{aligned} l_c &= \mathbb{E}_{\mathbf{S}_\mathcal{A}\sim\mathcal{A}}[\log \mathcal{D}_\mathcal{A}(\mathbf{S}_\mathcal{A})] \\ &\quad + \mathbb{E}_{\tilde{\mathbf{S}}_\mathcal{A}\sim\mathcal{A}}[1 - \log \mathcal{D}_\mathcal{A}(\tilde{\mathbf{S}}_\mathcal{A})] \\ &\quad + \mathbb{E}_{\mathbf{S}_\mathcal{B}\sim\mathcal{B},\mathbf{z}\sim\mathcal{N}}[1 - \log \mathcal{D}_\mathcal{A}(\mathcal{G}_\mathcal{A}(\mathcal{G}_\mathcal{B}(\mathbf{S}_\mathcal{B}, \mathbf{z})))], \end{aligned} \tag{10}$$

where we omit the exact definition of $\mathbf{S}_\mathcal{B}$ for simplicity. We also add a reconstruction loss l_r for $\mathcal{G}_\mathcal{A}$ based on $L1$ norm to enforce the network to learn the mapping from a sample in one domain to a similar one in another domain:

$$l_r = \left\| \mathcal{G}_\mathcal{A}(\mathcal{G}_\mathcal{B}(\mathbf{I},t,\mathbf{z})) - \mathcal{G}_\mathcal{B}(\mathbf{I},t,\mathbf{z}) \right\|_1. \tag{11}$$

Training algorithm. Our network is trained by solving the following minimax optimization problem:

$$\mathcal{G}_\mathcal{A}^*, \mathcal{D}_\mathcal{A}^*, \mathcal{G}_\mathcal{B}^*, \mathcal{D}_\mathcal{B}^* = \min_{\mathcal{G}_\mathcal{A},\mathcal{G}_\mathcal{B}} \max_{\mathcal{D}_\mathcal{A},\mathcal{D}_\mathcal{B}} l_u + l_c + \lambda l_r, \tag{12}$$

where λ is the weight of the reconstruction loss l_r defined in Eq. (11). Note that l_c is only used for updating $\mathcal{G}_\mathcal{B}$ and we do not update $\mathcal{G}_\mathcal{A}$ with the gradient from l_c, since the purpose of using $\mathcal{G}_\mathcal{A}$ is to translate the domain without considering the timestamp condition. In addition, we update $\mathcal{G}_\mathcal{A}$ and $\mathcal{G}_\mathcal{B}$ alternately as they are dependent on each other. Our training procedure is described in Algorithm 1 step by step.

3.3. Guided Upsampling

Since our training data is very limited and contains lots of noise, it is difficult to train the network to directly output full-resolution results while completely preserving the local structure in the input image. Therefore, we first train our

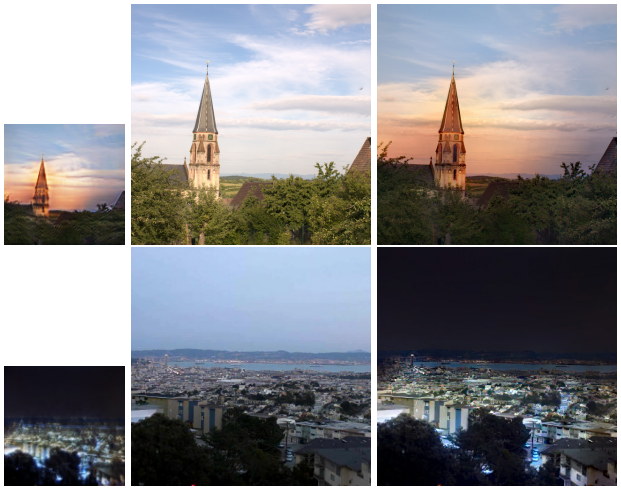

Figure 5: We apply guided upsampling in a post-processing step to get full-resolution output. From left to right: the output predicted by our network, the original input, and the upsampling result.

generative network and predict output $\hat{\mathbf{I}}$ at a lower resolution. Then we apply an automatic guided upsampling approach, following the local color transfer method in [10], as a post-processing step to obtain the full-resolution result.

Basically, we model the per-pixel linear transformation between the final result $\bar{\mathbf{I}}$ and the input image $\mathbf{I}$ at a pixel location p as a scaling factor $s(p)$ with a bias $b(p)$:

$$\bar{\mathbf{I}}(p) = \mathbf{T}(p)(\mathbf{I}(p)) = s(p) \times \mathbf{I}(p) + b(p). \tag{13}$$

The key idea of guided upsampling is to use the raw network output $\hat{\mathbf{I}}$ as the guidance to compute the transformation $\mathbf{T} = \{s, b\}$, while using color information between neighboring pixels in the input image $\mathbf{I}$ as regularization. Specifically, we formulate the task as the following least-squares problem:

$$\begin{aligned} \mathcal{E} &= \mathcal{E}_d + \mu\, \mathcal{E}_s, \\ \mathcal{E}_d &= \sum_p \left\| \bar{\mathbf{I}}(p) - \hat{\mathbf{I}}(p) \right\|^2, \\ \mathcal{E}_s &= \sum_p \sum_{q \in \mathbf{N}(p)} w\big(\mathbf{I}(p), \mathbf{I}(q)\big) \left\| \mathbf{T}(p) - \mathbf{T}(q) \right\|^2, \end{aligned} \tag{14}$$

where $\mathbf{N}(p)$ is the one-ring neighborhood of p and $w\big(\mathbf{I}(p), \mathbf{I}(q)\big)$ measures the inverse color distance between two neighboring pixels p and q in the original image $\mathbf{I}$. The data term $\mathcal{E}_d$ in Eq. (14) preserves the color from $\hat{\mathbf{I}}$ and the smoothness term $\mathcal{E}_s$ enforces local smoothness of the linear transformation $\mathbf{T}$ between neighboring pixels of similar colors. A global constant weight μ is used to balance the two energy terms. We compute the least-squares optimization for each color channel independently and then upsample $\mathbf{T}$ bilinearly to the full resolution before apply to the original image. See Figure 5 for two example results.

4. Experiments

4.1. Experimental Setup

Dataset. We use both the AMOS [12] and the TLVDB [27] datasets to train our network. For the AMOS dataset, we only select sequences with geolocation information and adjust all timestamp labels to local time accordingly. In addition, we remove some obviously corrupted data such as zero-byte images, grayscale images, etc. All in all, we collected $40,537$ sequences containing $1,310,375$ images from $1,633$ cameras. We split the collected AMOS dataset into a training set and a test set, which contain sequences from $1,533$ and 100 cameras, respectively. For the TLVDB dataset, we use 463 videos that have $1,065,427$ images without preprocessing. We randomly select 30 videos as the test set and use the remaining videos for training.

Implementation details. We implement our method using PyTorch. We train our model with $60,000$ iterations using Adam optimizer with the momentum set to 0.5. The batch size is set to be 4 and the learning rate is 0.0002. We use 16 frames for each example in a batch to train our multi-frame joint conditional GAN and set λ to be 0.5 based on visual quality. For data augmentation, we first resize images to 136×136 and then apply random affine transformation including rotation, scale, and shear followed by random horizontal flipping. Finally, the images are randomly cropped to patches of resolution 128×128. For the encoder of $\mathcal{G}_{\mathcal{B}}$, we adopt a pre-trained VGG-16 network [28] while all other components are trained from scratch.[1]

Baselines. We compare our method with two existing color transfer methods of Li *et al*. [19] and Shih *et al*. [27] using the source code with the default parameters provided by the authors. As both methods need reference videos to guide the output synthesis, we additionally implement the reference video retrieval method proposed in [27], which is to compare the global feature of an input image with reference video frames to find the most similar video. We use the output of a fully-connected layer in a ResNet-50 model [9] as the global feature. The model is pre-trained for a scene classification task [40]. We additionally use a pre-trained scene parsing network [41] to produce semantic segmentation masks used in Li *et al*.'s method [19].

4.2. Experimental Results

Quantitative results. Since both baseline methods require a reference video as input while ours does not, it is difficult to conduct a completely fair comparison side-by-side. To evaluate our method quantitatively, we performed a human evaluation following the experiment in [27].

[1] Please refer to the supplementary materials for more details.

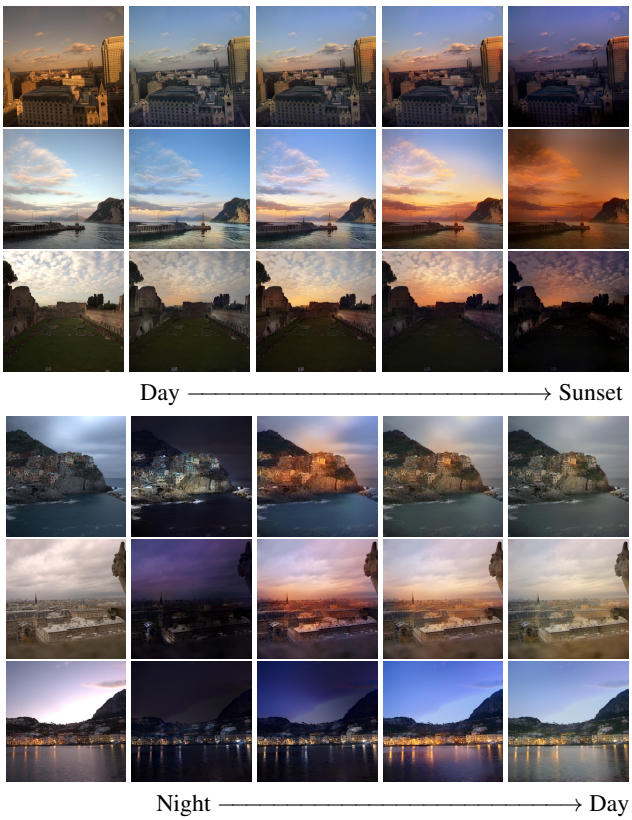

Figure 6: Our prediction results of the input images at different times of the day. The input images are shown on the left in each row.

Specifically, we select 24 images from 30 test images in the TLVDB test set and generated 24 time-lapse sequences using the two baseline approaches [19, 27] and our method. Then, we randomly select two or three images from each output sequence. Eventually, we collect 71 images for each method. We additionally selected the same number of frames from the original videos of the test images to consider real images as another baseline. We conducted a user study on Amazon Mechanical Turk by asking 10 users if each image was real or fake. We restricted the users to those who had high approval rates greater than 98% to control quality.

As the result, 60.6% of our results were perceived as real images by the users. In contrast, the corresponding numbers for real images, Li *et al.*'s method [19], and Shih *et al.*'s method [27] are 67.5%, 34.1%, and 44.9%, respectively. Our percentage is also higher than the reported value in [27], which is 55.2%. We attribute the lower performance of two baseline methods to the failure of color transfer when the retrieved reference video does not perfectly match the input image. In contrast, our results were mostly preferred by the users without using any reference video.

Figure 7: Comparisons with existing methods. The same input image is shown on the left. From top to bottom, we show frames from a retrieved reference video, the results from [19], [27], and our method, respectively.

Qualitative results. Figure 6 shows our results based on a variety of outdoor images from the MIT-Adobe 5K dataset [3]. Our method can robustly handle input images with different semantic compositions and effectively synthesize illumination changes over time. Figure 7 shows qualitative comparisons between our results and those from the two baseline methods [27, 19]. The input image is repeatedly shown in the first column in Figure 7. The first row shows a set of frames from a retrieved reference video. Starting from the second row, we show the results generated by [19], [27], and our method, respectively. In many cases, both baseline methods produce unrealistic images, which is mainly because the scene in the reference video does not perfectly match the input image. For both baseline methods, the color tone changes are driven by the retrieved reference video and may present noticeable visual artifacts if the reference video has semantic composition significantly different from the input image. In contrast, our method can effectively generate plausible color tone changes over time without a reference video. Also the color tone changes in our results are more visually pleasing due to the use of both AMOS and TLVDB datasets.

Computation time. At inference time, we only need $\mathcal{G}_{\mathcal{B}}$ to synthesize time-lapse videos. The inference of $\mathcal{G}_{\mathcal{B}}$ takes about 0.02 seconds with a GPU and 0.8 seconds with a CPU. The guided upsampling step takes about 0.1 seconds on a CPU for an original input image of resolution 512×512. In contrast, Shih *et al.*'s method [27] takes 58 seconds for a 700-pixels width image on CPU, and Li *et al.*'s method [19] takes 6.3 seconds for 768×384 resolution on GPU. Therefore,

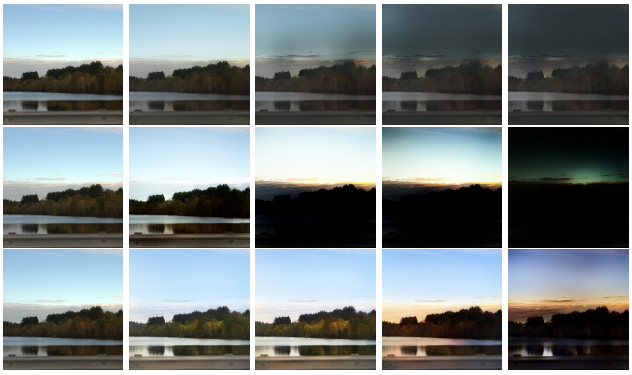

Figure 8: Ablation study of our algorithm. The same input is shown on the left. From top to bottom, we show results from (A) a vanilla cGAN, (B) our multi-frame joint conditional GAN only, and (C) our full algorithm, respectively.

our method is much faster than existing methods and is more suitable for deployment on mobile devices.

4.3. Discussion

Ablation studies. We conduct ablation studies to verify important components of our proposed method. In Figure 8, we show some qualitative results to compare with our own baselines: (A) a vanilla cGAN, (B) our multi-frame joint conditional GAN without the multi-domain training, and (C) our full algorithm with both the multi-frame joint conditional GAN and the multi-domain training. Method (A) only changes the overall brightness without considering color tone changes, especially at a transition time such as the sunrise and the sunset. This issue is because there can be various illumination changes at a specific time due to locations, season, and weather changes, which confuses the generator. Thus, the generator is likely to change brightness only as the easiest way to fool the discriminator. Method (B) effectively captures illumination changes over time by considering the context of the entire sequence. In many cases, however, it produces clipped pixels and less saturated color tone, because the AMOS dataset consists of footages captured by surveillance cameras which are visually less interesting. Our full algorithm (C) overcomes this limitation by jointly learning from both the AMOS dataset and the TLVDB dataset.

Evaluation of multi-domain training. Figure 9 shows some examples after translating the output of $\mathcal{G}_{\mathcal{B}}$ into the domain of AMOS dataset $\mathcal{A}$ using $\mathcal{G}_{\mathcal{A}}$ for the conditional training. Directly training $\mathcal{G}_{\mathcal{B}}$ using $\mathcal{D}_{\mathcal{A}}$ fails due to the domain discrepancy of the two datasets such as different color tones and scene compositions. As shown in the figure, $\mathcal{G}_{\mathcal{A}}$ can effectively change the output of $\mathcal{G}_{\mathcal{B}}$ to fool $\mathcal{D}_{\mathcal{A}}$ and get conditional training signals from it.

Figure 9: The effect of using $\mathcal{G}_{\mathcal{A}}$. The input image is shown on the left. Two image pairs of before (the 2nd and 4th columns) and after $\mathcal{G}_{\mathcal{A}}$ (the 3rd and 5th columns) are shown to verify the multi-domain training.

5. Conclusions

In this paper, we presented a novel framework for the time-lapse video synthesis from a single outdoor image. Given an input image, our conditional generative adversarial network can predict the illumination changes over time by using the timestamp as the control variable. Compared to other methods, we do not require semantic segmentation or a reference video to guide the generation of the output video.

Figure 10: Failure cases.

Our method still has some limitations. As shown in Figure 10, our method fails to hallucinate daytime images from a nighttime input where most parts of the input are very dark. In some cases, our method fails to generate artificial lighting in regions such as building windows. In addition, our method only changes the color tones of a given input image without introducing any motions such as moving objects. It would be interesting to combine our approach with frame prediction or motion synthesis techniques [35] to generate time-lapse videos with both interesting motions and illumination changes. We also plan to extend our approach to support additional semantic controls such as sunrise and sunset times in the prediction results [15]. Finally, we would like to investigate using our synthesis framework with an implicit control variable for general video synthesis tasks.

Acknowledgements. This work was supported by National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (NRF-2016R1A2B4014610) and Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (2014-0-00059). Seonghyeon Nam was partially supported by Global Ph.D. Fellowship Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF2015H1A2A1033924).

References

[1] Asha Anoosheh, Torsten Sattler, Radu Timofte, Marc Pollefeys, and Luc Van Gool. Night-to-day image translation for retrieval-based localization. *arXiv preprint arXiv:1809.09767*, 2018.

[2] Aayush Bansal, Shugao Ma, Deva Ramanan, and Yaser Sheikh. Recycle-gan: Unsupervised video retargeting. In *The European Conference on Computer Vision (ECCV)*, pages 119–135, 2018.

[3] Vladimir Bychkovsky, Sylvain Paris, Eric Chan, and Frédo Durand. Learning photographic global tonal adjustment with a database of input/output image pairs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 97–104, 2011.

[4] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1520–1529, 2017.

[5] Yung-Yu Chuang, Dan B Goldman, Ke Colin Zheng, Brian Curless, David H. Salesin, and Richard Szeliski. Animating pictures with stochastic motion textures. *ACM Trans. Graph.*, 24(3):853–860, 2005.

[6] 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)*, pages 248–255, 2009.

[7] 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)*, pages 2414–2423, 2016.

[8] 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*, pages 2672–2680, 2014.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[10] Mingming He, Jing Liao, Lu Yuan, and Pedro V Sander. Neural color transfer between images. *arXiv preprint arXiv:1710.00756*, 2017.

[11] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976, 2017.

[12] Nathan Jacobs, Nathaniel Roman, and Robert Pless. Consistent temporal variations in many outdoor scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–6, 2007.

[13] Wei-Cih Jhou and Wen-Huang Cheng. Animating still landscape photographs through cloud motion creation. *IEEE Transactions on Multimedia*, 18(1):4–13, 2016.

[14] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *The European Conference on Computer Vision (ECCV)*, pages 694–711, 2016.

[15] Levent Karacan, Zeynep Akata, Aykut Erdem, and Erkut Erdem. Learning to generate images of outdoor scenes from attributes and semantic layouts. *arXiv preprint arXiv:1612.00215*, 2016.

[16] Hyeongwoo Kim, Pablo Carrido, Ayush Tewari, Weipeng Xu, Justus Thies, Matthias Niessner, Patrick Pérez, Christian Richardt, Michael Zollhöfer, and Christian Theobalt. Deep video portraits. *ACM Trans. Graph.*, 37(4):163:1–163:14, 2018.

[17] Pierre-Yves Laffont and Jean-Charles Bazin. Intrinsic decomposition of image sequences from local temporal variations. In *IEEE International Conference on Computer Vision (ICCV)*, pages 433–441, 2015.

[18] Hsin-Ying Lee, Hung-Yu Tseng, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Diverse image-to-image translation via disentangled representations. In *The European Conference on Computer Vision (ECCV)*, pages 35–51, 2018.

[19] Yijun Li, Ming-Yu Liu, Xueting Li, Ming-Hsuan Yang, and Jan Kautz. A closed-form solution to photorealistic image stylization. In *The European Conference on Computer Vision (ECCV)*, pages 453–468, 2018.

[20] Zhengqi Li and Noah Snavely. Learning intrinsic image decomposition from watching the world. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9039–9048, 2018.

[21] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *The European Conference on Computer Vision (ECCV)*, pages 740–755, 2014.

[22] Fujun Luan, Sylvain Paris, Eli Shechtman, and Kavita Bala. Deep photo style transfer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6997–7005, 2017.

[23] Wojciech Matusik, Matthew Loper, and Hanspeter Pfister. Progressively-refined reflectance functions from natural illumination. In *Rendering Techniques*, pages 299–308, 2004.

[24] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[25] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[26] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241, 2015.

[27] Yichang Shih, Sylvain Paris, Frédo Durand, and William T. Freeman. Data-driven hallucination of different times of day from a single outdoor photo. *ACM Trans. Graph.*, 32(6):200:1–200:11, 2013.

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

[29] Kalyan Sunkavalli, Wojciech Matusik, Hanspeter Pfister, and Szymon Rusinkiewicz. Factored time-lapse video. *ACM Transactions on Graphics*, 26(3):101:1–101:10, 2007.

[30] 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)*, pages 1526–1535, 2018.

[31] Ruben Villegas, Jimei Yang, Seunghoon Hong, Xunyu Lin, and Honglak Lee. Decomposing motion and content for natural video sequence prediction. In *International Conference on Learning Representations (ICLR)*, 2017.

[32] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *Advances In Neural Information Processing Systems*, pages 613–621, 2016.

[33] Jacob Walker, Carl Doersch, Abhinav Gupta, and Martial Hebert. An uncertain future: Forecasting from static images using variational autoencoders. In *The European Conference on Computer Vision (ECCV)*, pages 835–851, 2016.

[34] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. In *Advances in Neural Information Processing Systems*, pages 1152–1164, 2018.

[35] 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)*, pages 2364–2373, 2018.

[36] Xuemiao Xu, Liang Wan, Xiaopei Liu, Tien-Tsin Wong, Liansheng Wang, and Chi-Sing Leung. Animating animal motion from still. *ACM Trans. Graph.*, 27(5):117:1–117:8, 2008.

[37] Tianfan Xue, Jiajun Wu, Katherine Bouman, and Bill Freeman. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2016.

[38] Zili Yi, Hao (Richard) Zhang, Ping Tan, and Minglun Gong. DualGAN: Unsupervised Dual Learning for Image-to-Image Translation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2868–2876, 2017.

[39] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. Deep Sets. In *Advances in Neural Information Processing Systems*, pages 3391–3401, 2017.

[40] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(6):1452–1464, 2018.

[41] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[42] Yipin Zhou and Tamara L Berg. Learning temporal transformations from time-lapse videos. In *The European Conference on Computer Vision (ECCV)*, pages 262–277, 2016.

[43] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2242–2251, 2017.