DeepLandscape: Adversarial Modeling of Landscape Videos

Elizaveta Logacheva[1], Roman Suvorov[1], Oleg Khomenko[1], Anton Mashikhin[1], and Victor Lempitsky[1,2]

[1] Samsung AI Center, Moscow
[2] Skolkovo Institute of Science and Technology, Moscow
elimohl@gmail.com

Abstract. We build a new model of landscape videos that can be trained on a mixture of static landscape images as well as landscape animations. Our architecture extends StyleGAN model by augmenting it with parts that allow to model dynamic changes in a scene. Once trained, our model can be used to generate realistic time-lapse landscape videos with moving objects and time-of-the-day changes. Furthermore, by fitting the learned models to a static landscape image, the latter can be reenacted in a realistic way. We propose simple but necessary modifications to StyleGAN inversion procedure, which lead to in-domain latent codes and allow to manipulate real images. Quantitative comparisons and user studies suggest that our model produces more compelling animations of given photographs than previously proposed methods. The results of our approach including comparisons with prior art can be seen in supplementary materials and on the project page https://saic-mdal.github.io/deep-landscape/.

1 Introduction

This work is motivated by the "bringing landscape images to life" application. We thus aim to build a system that for a given landscape photograph, generates its plausible animation with realistic movements and global lighting changes. To achieve our goal, we first build a generative model (Figure 1) of timelapse landscape videos, which can successfully capture complex aspects of this domain. These complexities include both static aspects such as abundance of spatial details, high variability of texture and geometry, as well as dynamic complexity including motions of clouds, waves, foliage, and global lighting changes. We build our approach upon the recent progress in the generative modeling of images, and specifically the StyleGAN model [1]. We show how to change the StyleGAN model to learn and to decompose different dynamic effects: global changes are controlled by the non-convolutional variables, strong local motions are controlled by "noise branch" inputs.

Similarly to the original StyleGAN model, ours requires a large amount of training data. While it is very hard to obtain a very large dataset of high-quality scenery timelapse videos, obtaining a large-scale dataset of scenery static images is much easier. We thus suggest how our generative model can be learned from two sources, namely (i) a large-scale dataset of static images, (ii) a smaller dataset of videos. Previous video GANs learn motion from sequences of consecutive video frames. We show that learning on randomly taken frames without an explicit motion model is possible. It allows to disentangle static appearance from the dynamic, as well as manifold of possible changes from a trajectory in it.

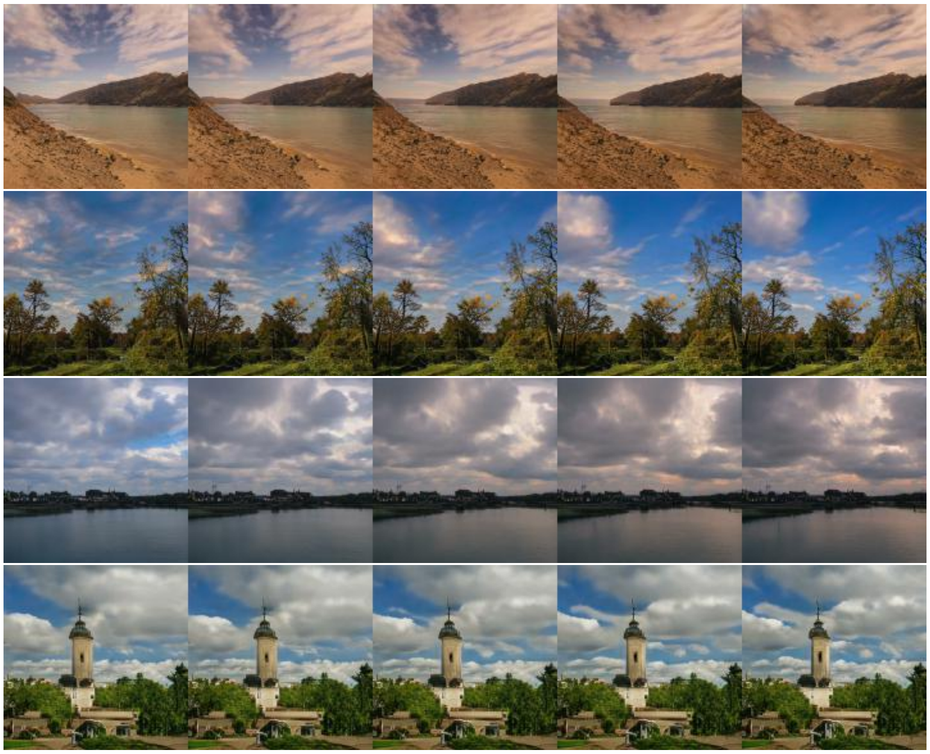


Fig. 1. Videos generated by the DeepLandscape model. Each row shows a separate video, obtained by sampling the static and dynamic components randomly, and then animating the dynamic components using homography warping. These videos are generated at 512 × 512 resolution (*zoom-in recommended*).

Once trained, our model can animate a given photograph. We first fit the latent variables of the model to the provided image, and then obtain the animation by changing the subset of variables corresponding to dynamic aspects appropriately. As our model has more latent parameters than a given static image, fitting them to a photograph is an ill-posed problem, and we develop a particular method for such fitting that results in plausible animations. While our model is trained to generate images at medium resolution (256×256 or 512×512), we show that we can postprocess the results with an appropriately trained super-resolution network to obtain videos at higher resolution (up to one megapixel).

In the experiments, we assess the realism of synthetic videos sampled from our generative model and its ablations. Furthermore, we evaluate our approach at our main task ("bringing landscape images to life"). For this task, both quantitative comparisons and, more importantly, user studies reveal a significant advantage of our system over the three recently proposed approaches [2–4].

2 Related work

Learning video representation and predicting future frames using deep neural networks is a very active area of research [5–8]. Most early works are focused on using deep neural networks (DNNs) with recurrent units (GRU or LSTM) and train them in supervised manner to obtain next frame using pixel-level prediction [8, 5]. At the same time, Generative Adversarial Nets (GANs) [9] have achieved very impressive results for image generation, and recently several methods extending them to video have been suggested. Some GAN-based models consider single image as an input (*image2video*) [10, 11], while others input sequences of frames (*video2video*, [7, 12–15]). In this work we focus only on the image2video setting. Training GANs for video-generation often performed with two discriminator networks: single image and temporal discriminators [12, 16, 14]. In this work we propose to use a simplified temporal discriminator, which only looks at unordered pairs of frames.

Video generation/prediction works generally consider either videos with articulated objects/multiple moving objects [17, 18] or videos with weakly structured moving objects or dynamic textures such as clouds, grass, fire [14, 4]. Our work is more related to the latter case, namely: landscape photos and videos. Because of the domain specifics, we can model spatial motions in the video in the *latent* space using simple homography transformations, and let the generator to synthesize plausible deviations from this simplistic model. Our approach is thus opposed to methods that animate landscapes and textures by generating warping fields applied to the *raw pixels* of the input static image [2, 19, 11, 20, 21]. Animation in the latent space as well as the separation of latent space into static and dynamic components has been proposed and investigated in [22, 6, 23, 2, 24]. Our work modifies and extends these ideas to the StyleGAN [1] model.

As we need to find latent space embedding of static images in order to animate them, we follow a number of works on GAN inversion (inference). Here, we borrow ideas of using an encoder into the latent space followed by gradient descent [25], the latent space expansion for StyleGAN [26], and generator fine-tuning [27, 28]. On top of that, we have to make several important adjustments to the inference procedure specific to our architecture, and we show that without such adjustments the animation works poorly.

3 Method

3.1 Generative model of timelapse videos

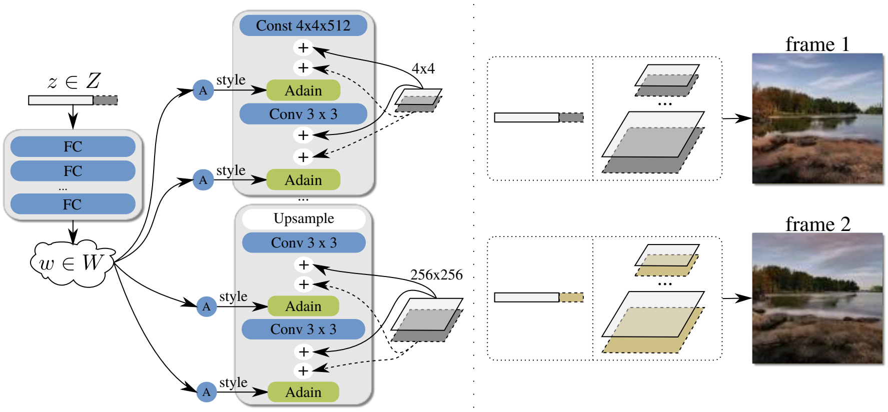


Fig. 2. Left – the generator used by our model (augmented StyleGAN generator). The main difference from StyleGAN is the second set of spatial input tensors (darkgray). Right – sampling procedure for our model. Two frames of the same video can be sampled by using same static latent variables (lightgray), and two different sets of dynamic latent variables (darkgray and yellow).

Model architecture. The architecture of our model is based on StyleGAN [1]. Our model outputs images of resolution 256×256 (or 512×512) and has four sets of latent variables:

- a vector $\mathbf{z}^{\mathrm{st}} \in \mathbb{R}^{D^{\mathrm{st}}}$, which encodes colors and the general scene layout;
- a vector $\mathbf{z}^{\mathrm{dyn}} \in \mathbb{R}^{D^{\mathrm{dyn}}}$, which encodes global lighting (e.g. time of day);
- a set $\mathcal{S}^{\mathrm{st}}$ of square matrices $S_1^{\mathrm{st}} \in \mathbb{R}^{4\times 4}$, ..., $S_N^{\mathrm{st}} \in \mathbb{R}^{2^{N+1}\times 2^{N+1}}$, which encode shapes and details of static objects at $N = 7$ different resolutions between 4×4 and 256×256 ($N = 8$ for 512×512);
- a set $\mathcal{S}^{\mathrm{dyn}}$ of square matrices $S_1^{\mathrm{dyn}} \in \mathbb{R}^{4\times 4}$, ..., $S_N^{\mathrm{dyn}} \in \mathbb{R}^{2^{N+1}\times 2^{N+1}}$, which encode shapes and details of dynamic objects at the corresponding resolutions.

Our generator has two components: the multilayer perceptron $\mathbf{M}$ and the convolutional generator $\mathbf{G}$. As in [1], the perceptron $\mathbf{M}$ takes the concatenated vector $\mathbf{z} = \left[\mathbf{z}^{\mathrm{st}}, \mathbf{z}^{\mathrm{dyn}}\right] \in \mathbb{R}^{512}$ and transforms it to the *style vector* $\mathbf{w} \in \mathbb{R}^{512}$. The convolutional generator $\mathbf{G}$ also follows [1] and has $N = 7$ (or 8) blocks. Within each block, a convolution is followed by two elementwise additions of two tensors obtained from S_n^{st} and S_n^{dyn} by a learnable per-channel scaling (whereas [1] has only one addition). Finally, the AdaIN [29] transform is applied using per-channel scales and biases obtained from $\mathbf{w}$ using learnable linear transform. Within each block, this sequence of steps is repeated twice followed by upsampling and convolution layers.

Below, we will refer to the set of input latent variables

$$\left\{\mathbf{z}^{\mathrm{st}}, \mathbf{z}^{\mathrm{dyn}}, S_1^{\mathrm{st}}, ..., S_N^{\mathrm{st}}, S_1^{\mathrm{dyn}}, ..., S_N^{\mathrm{dyn}}\right\}$$

as *original inputs* (or original latents). As in StyleGAN, the convolutional generator may use *separate* $\mathbf{w}$ vectors at each of the resolution (style mixing). We will then refer to the set of all style vectors as $\mathcal{W} = \{\mathbf{w}_1, ..., \mathbf{w}_N\}$. Finally, we will denote the set of all spatial random inputs of the generator as $\mathcal{S} = \{\mathcal{S}^{\mathrm{st}}, \mathcal{S}^{\mathrm{dyn}}\} = \left\{S_1^{\mathrm{st}}, ..., S_N^{\mathrm{st}}, S_1^{\mathrm{dyn}}, ..., S_N^{\mathrm{dyn}}\right\}$.

Learning the model. The model is trained from two sources of data, the dataset of static scenery images $\mathcal{I}$ and the dataset of timelapse scenery videos $\mathcal{V}$. It is relatively easy to collect a large static dataset, while with our best efforts we were able to collect a few hundreds of videos, that do not cover all the diversity of landscapes. Thus, both sources of data have to be utilized in order to build a good model. To do that, we train our generative model in an adversarial way with two different discriminators.

The *static discriminator* D_{st} has the same architecture and design choises as in StyleGAN. It observes images from $\mathcal{I}$ as real, while the fake samples are generated by our model. The *pairwise discriminator* D_{dyn} looks at pairs of images. It duplicates the architecture of D_{st} except first convolutional block that is applied separately to each frame. A real pair of images is obtained by sampling a video from $\mathcal{V}$, and then sampling two random frames (arbitrary far for each other) from it. A fake pair is obtained by sampling common static latents $\mathbf{z}^{\mathrm{st}}$ and $\mathcal{S}^{\mathrm{st}}$, and then individual dynamic latents $\mathbf{z}^{\mathrm{dyn},1}$, $\mathbf{z}^{\mathrm{dyn},2}$ and $\mathcal{S}^{\mathrm{dyn},1}$, $\mathcal{S}^{\mathrm{dyn},2}$. The two images are then obtained as $\mathbf{G}(\mathbf{M}(\mathbf{z}^{\mathrm{st}}, \mathbf{z}^{\mathrm{dyn},1}), \mathcal{S}^{\mathrm{st}}, \mathcal{S}^{\mathrm{dyn},1})$ and $\mathbf{G}(\mathbf{M}(\mathbf{z}^{\mathrm{st}}, \mathbf{z}^{\mathrm{dyn},1}), \mathcal{S}^{\mathrm{st}}, \mathcal{S}^{\mathrm{dyn},2})$. All samples are drawn from unit normal distributions.

The model is trained within standard GAN approach with non-saturating loss [9] with R1 regularization [30] as in the original StyleGAN paper. During each update of the generator, we either sample a batch of fake images to which the static discriminator is applied or a batch of image pairs to which the pairwise discriminator is applied. The proportions of the static discriminator and the pairwise discriminator are annealed from 0.5/0.5 to 0.9/0.1 respectively over each resolution transition phase and then kept fixed at 0.1. This helps the generator to learn disentangle static and dynamic latents early for each resolution and prevents the pairwise generator from overfitting to our relatively small video dataset.

During learning, we want the pairwise discriminator to focus on the inconsistencies within each pair, and leave visual quality to the static discriminator. Furthermore, since the pairwise discriminator only sees real frames sampled from a limited number of videos, it may prone overfit to this limited set and effectively stop contributing to the learning process (while the static discriminator, which observes more diverse set of scenes, keeps improving the diversity of the model). It turns out, both problems (focus on image quality rather than pairwise consistency, overfitting to limited diversity of videos) can be solved with a simple trick. We augment the fake set of frames with pairs of crops taken from same video frame, but from different locations. Since these crops have the same visual quality as the images in real frames, and since they come from the same videos as images within real pairs, the pairwise discriminator effectively stops paying attention to image quality, cannot simply overfit to the statistics of scenes in the video dataset, and has to focus on finding pairwise inconsistencies within fake pairs. We observed this *crop sampling* trick to improve the quality of our model significantly.

Config	I2S [26]	MO	E	EO	EOI	EOIF	EOIFS
Init $\mathcal{W}$	Mean	Mean	**E**	**E**	**E**	**E**	**E**
Init $\mathcal{S}$	Random	Zero	Random	Zero	Zero	Zero	Zero
Optimize $\mathcal{S}$		+		+	+	+	+
Optimize $\mathcal{W}$	+	+		+	+	+	+
L^{O}_{init}					+	+	+
Fine-Tune **G**						+	+
Segmentation							+
Reconstruction	-	+	-	+	±	+	+
Animation	-	-	+	-	+	+	+

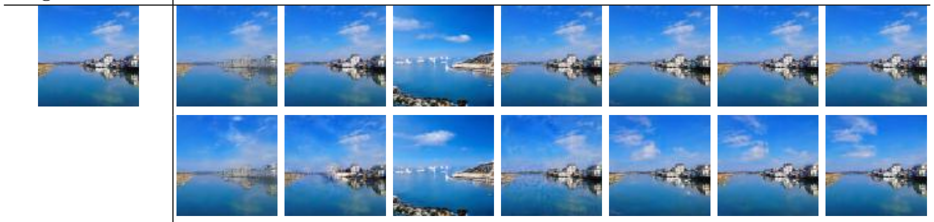


Fig. 3. The effect of different inference algorithms on the reconstruction quality and the ability to animate. **Left column**: original image. **First row**: reconstructions obtained with different inference algorithms. **Second row**: a frame from animation ($\mathcal{S}^{\text{dyn}}$ are shifted 50% left). Note that I2S [26] does not work well in our case, since our generator relies on $\mathcal{S}$ more than the original StyleGAN method. L^{O}_{init} is a regularization term applied to $\mathcal{W}$ during inference, which makes latents to stay in-domain and allows to manipulate real images. We quantify these effects in supplementary materials.

Sampling videos from the model. Our model does not attempt to learn full temporal dynamics of videos, and instead focuses on pairwise consistency of frames that are generated when the dynamic latent variables are resampled. In particular, the pairwise discriminator in our model does not sample real frames sequentially. The sampling procedure for fake pairs does not try to generate adjacent frames either. One of the reasons why we do not attempt to learn continuity, is because the training dataset contains videos of widely-varying temporal rates, making the notion of temporal adjacency for a pair of frames effectively meaningless.

Because of this our generation process is agnostic to a model of motion. The generator is forced to produce plausible frames regardless of $\mathcal{S}^{\mathrm{dyn}}$ and $\mathbf{z}^{\mathrm{dyn}}$ changes. In our experiments we found that a simple model of motion described below is enough to produce compelling videos. Specifically, to sample a video, we sample a single static vector $\mathbf{z}^{\mathrm{st}}$ from the unit normal distribution and then interpolate the dynamic latent vector between two unit normally-distributed samples $\mathbf{z}^{\mathrm{dyn},1}$ and $\mathbf{z}^{\mathrm{dyn},2}$. For the spatial maps, we again sample $\mathcal{S}^{\mathrm{st}}$ and $\mathcal{S}^{\mathrm{dyn},1}$ from a unit normal distribution and then warp the $\mathcal{S}^{\mathrm{dyn}}$ tensor continuously using a homography transform parameterized by displacements of two upper corners and two points at the horizon. The direction of the homogrpahy is sampled randomly, speed was chosen to match the average speed of clouds in our dataset. The homography is flipped vertically for positions below the horizon to mimic the reflection process. To obtain $\mathcal{S}^{\mathrm{dyn,i}}$, we make a composition of $i-1$ identical transforms and then apply it to $\mathcal{S}^{\mathrm{dyn},1}$. As we interpolate/warp the latent variables, we pass them through the trained model to obtain the smooth videos (Figure 1 and **Supplementary video**). Note that our models requires no image-specific user input.

3.2 Animating Real Scenery Images with Our Model

Inference. To animate a given scenery image I, we find (infer) a set of latent variables that produce such image within the generator. Following [26], we look for extended latents $\mathcal{W}$ and $\mathcal{S}$, so that $\mathbf{G}(\mathcal{W},\mathcal{S}) \approx I$, but our procedure is different from theirs. After that, we apply the same procedure as described above to animate the given image.

The latent space of our generator is highly redundant, and to obtain good animation, we have to ensure that the latent variables come roughly from the same distribution as during the training of the model (most important, $\mathcal{W}$ should belong to the output manifold of $\mathbf{M}$). Without such prior, the latent variables that generate good reconstruction might still result in implausible animation (or lack of it). We therefore perform inference using the following three-step procedure:

1. **Step 1**: predicting a set of style vectors $\mathcal{W}'$ using a feedforward *encoder* network $\mathbf{E}$ [25]. The encoder has ResNet-152 [31] architecture and is trained on 200000 synthetic images with mean absolute error loss. $\mathcal{W}$ is predicted by two-layer perceptron with ReLU from the concatenation of features from several levels of ResNet, aggregated by global average pooling.
2. **Step 2**: starting from $\mathcal{W}'$ and zero $\mathcal{S}$, we optimize all latents to improve reconstruction error. In addition, we penalize the deviation of $\mathcal{W}$ from the predicted $\mathcal{W}'$ (with coefficient 0.01) and the deviation of $\mathcal{S}$ from zero (by reducing learning rate). We optimize for up to 500 steps with Adam [32] and

large initial learning rate (0.1), which is halved each time the loss does not improve for 20 iterations. A variant of our method that we evaluate separately, uses a binary segmentation mask obtained by ADE20k-pretrained [33] segmentation network[3]. The mask identifies dynamic (sky+water) and remaining (static) parts of the scene. In this variant, $\mathcal{S}^{\text{st}}$ (respectively $\mathcal{S}^{\text{dyn}}$) are kept at zero for dynamic (respectively, static) parts of the image.

3. **Step 3**: freezing latents and fine-tuning the weights of $\mathbf{G}$ to further drive down the reconstruction error [27, 28]. The step is needed since even after optimization, the gap between the reconstruction and the input image remains. During this fine-tuning, we minimize the combination of the per-pixel mean absolute error and the perceptual loss [34], with much larger (10×) weight for the latter. We do 500 steps with ADAM and $lr = 0.001$.

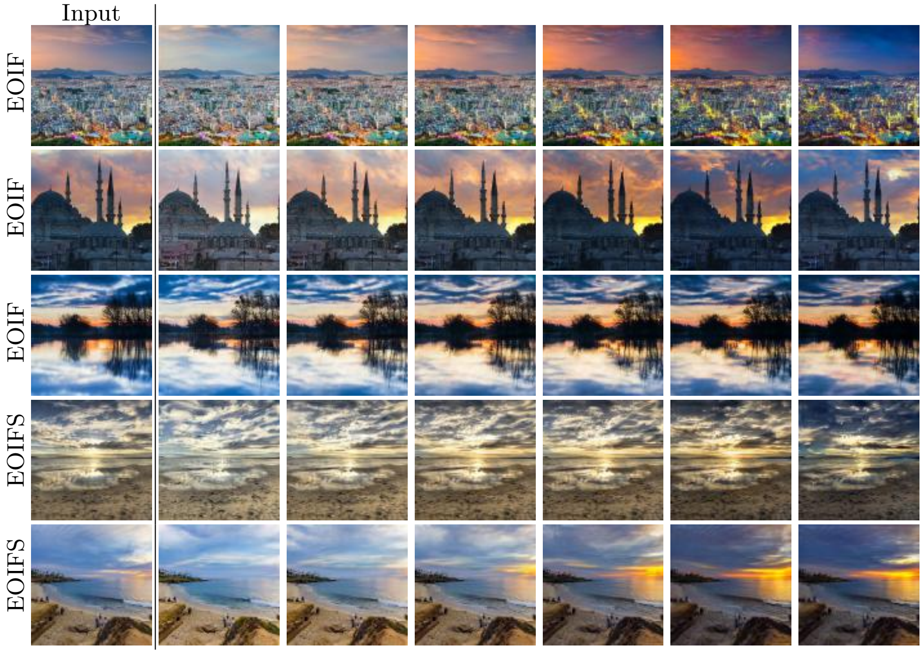


Fig. 4. Examples of real images animated with our model. Each row shows a sequence of frames from a single video. Each frame is 256 × 256 (please zoom in for details). Clouds, reflections and waves move and change their shape naturally; time of day also changes. More examples are available in the **Supplementary video**.

Please refer to Figure 3 and *Supplementary Materials* for examples of qualitative effects of fine tuning. We also evaluate our inference pipeline quantitatively (see Section 4).

[3] CSAIL-Vision: https://github.com/CSAILVision/semantic-segmentation-pytorch

Lighting manipulation. During training of the model, $\mathbf{M}$ is used to map $\mathbf{z}$ to $\mathbf{w}$. We resample $\mathbf{z}^{\mathrm{dyn}}$ in order to take into account variations of lighting, weather changes, etc. and to have $\mathbf{z}^{\mathrm{st}}$ describe only static attributes (land, buildings, horizon shape, etc.). To change lighting in a real image, one has to change $\mathbf{z}^{\mathrm{dyn}}$ and then use MLP to obtain new styles $\mathcal{W}$. Our inference procedure, however, outputs $\mathcal{W}$ and we have found it very difficult to invert $\mathbf{M}$ and obtain $\mathbf{z} = \mathbf{M}^{-1}(\mathbf{w})$.

To tackle this problem, we train a separate neural network, $\mathbf{A}$, to approximate local dynamics of $\mathbf{M}$. Let $\mathbf{w}_a = \mathbf{M}(\mathbf{z}_a^{\mathrm{st}}, \mathbf{z}_a^{\mathrm{dyn}})$ and $\mathbf{w}_b = \mathbf{M}(\mathbf{z}_b^{\mathrm{st}}, \mathbf{z}_b^{\mathrm{dyn}})$, we optimize $\mathbf{A}$ as follows: $\mathbf{A}(\mathbf{w}_a, \mathbf{z}_b^{\mathrm{dyn}}, c) \approx \mathbf{M}(\mathbf{z}_a^{\mathrm{st}}, \mathbf{z}_a^{\mathrm{dyn}}\sqrt{1-c} + \mathbf{z}_b^{\mathrm{dyn}}\sqrt{c})$, where $c \sim Uniform(0,1)$ is coefficient of interpolation between $\mathbf{w}_a$ and $\mathbf{w}_b$. Thus, $c = 0$ corresponds to $\mathbf{z}_a^{\mathrm{dyn}}$, so $\mathbf{A}(\mathbf{w}_a, \mathbf{z}_b^{\mathrm{dyn}}, 0) \approx \mathbf{w}_a$; $c = 1$ corresponds to $\mathbf{z}_b^{\mathrm{dyn}}$, so $\mathbf{A}(\mathbf{w}_a, \mathbf{z}_b^{\mathrm{dyn}}, 1) \approx \mathbf{w}_b$.

We implement this by the combination of L1-loss $L_{Abs}^{\mathbf{A}} = |\mathbf{w}_b - \mathbf{A}(\cdot)|$ and relative direction loss $L_{Rel}^{\mathbf{A}} = 1 - \cos(\mathbf{w}_b - \mathbf{w}_a, \mathbf{A}(\cdot) - \mathbf{w}_a)$. The total optimization criterion is $L^{\mathbf{A}} = L_{Abs}^{\mathbf{A}} + 0.1 L_{Rel}^{\mathbf{A}}$. We train $\mathbf{A}$ with ADAM [32] until convergence. At test time, the network $\mathbf{A}$ allows us to sample a random target $\mathbf{z}_b^{\mathrm{dyn}}$ and update $\mathcal{W}$ towards it by increasing the interpolation coefficient c as the animation progresses. Please refer to Figure 4 and **Supplementary Video** for examples of animations with our full pipeline.

Super Resolution (SR). As our models are trained at medium resolution (e.g. 256×256), we aim to bring fine details from the given image that we need to animate through a separate super-resolution procedure. The main idea of our super resolution approach is to borrow as much as possible from the original high-res image (which is downsampled for animation via $\mathbf{G}$). To achieve that, we super-resolve the animation and blend it with the original image using a standard image superresolution approach. We use ESRGANx4 [35] trained on a dedicated dataset that is created as follows. To obtain the (hi-res, low-res) pair, we take a frame I from our video dataset as a hi-res image, we downsample it and run the first two steps of inference and obtain an (imperfect) low-res image. Thus, the network is trained on a more complex task than superresolution.

After obtaining the super-resolved video, we transfer dynamic parts (sky and water) from it to the final result. The static parts are obtained by running the guided filter [36] on the super-resolved frames while using the input high-res image as a guide. Such procedure effectively transfers high-res details from the input, while retaining the lighting change induced by lighting manipulation (Figure 5).

4 Experiments

We evaluate our method both quantitatively and qualitatively (via user study) on synthetic and real images separately. Evaluation on synthetic images (*generation*) aims on quantifying impact of major design choices of $\mathbf{G}$ itself (without encoding and super-resolution). Evaluation on real images (*animation*) aims on comparison with previous single-image animation methods, including Animating Landscape (*AL*) [2], SinGAN (*SG*) [3] and Two-Stream Networks (*TS*) [4]. The Animating Landscape system is based on learnable warping and is trained on more than a thousand time-lapse videos from [37, 14]. The SinGAN method creates a hierarchical model of image content based on the input model alone. It therefore has an advantage of not needing an external dataset, though, as a downside, it requires considerable time to fit a new image. Two-Stream Networks [4] create animated textures given a static texture image and a short clip (an example of motion) via optimization of video tensor. We also tried a to include two more baselines, i.e. linear dynamic systems [38] and Seg2Vid [10], but with former we got very poor quality and the latter failed to converge on our data, so we did not proceed with full comparison. We also tried to train and finetune *AL* on our video dataset (which is significantly smaller than that from *AL* paper), with little success (see supp.mat.).

We estimate quality through three different aspects: *individual image quality*; *static consistency*; *animation plausibility*. Individual image quality is estimated via Fréchet Inception Distance [39], masked SSIM [40] and LPIPS [41]. Static consistency evaluation aims on quantifying how good objects that must not move (e.g. buildings, mountains etc.) are preserved over time. For that purpose we calculate SSIM and LPIPS between first frame and each generated video frame (only for static parts). Perfect image quality and static consistency can be achieved by not animating anything at all. Thus, we evaluate animation plausibility via user study and Fréchet Video Distance [42].

To generate videos using our method, we use a manually constructed set of homographies. Data-driven estimation of homographies is out scope of this work, so we have prepared 12 homographies, one for each clock position (e.g. the "12h" move clouds up and towards the observer, the "3h" moves straight to the right, etc.). Normally, these homographies resemble the average speed of clouds in our training dataset. We increase this speed for synthetic experiments to make differences between variants of our method more obvious; we slow down animation for experiments with real images in order to approximately align our speed with that of the competitors (AL, SG and TS).

Datasets. Our model was trained using both videos and single images available in the Internet under Creative Commons License. For evaluation we use 69 landscape FullHD time-lapse videos published on YouTube between Dec. 28 2019 and Jan. 29 2020. For FID computation, we have collected 2400 pictures from Flickr[4].

Generation. In order to perform thorough ablation study in reasonable time, we perform all evaluations in this section at 128 × 128 resolution. To estimate static consistency, we sample 1200 pairs of images from $\mathbf{G}$, mask out sky and water according to segmentation mask and calculate LPIPS and SSIM between two images in a pair. In each pair the images are generated from the same $\mathbf{z}^{\text{st}}, \mathcal{S}^{\text{st}}$ and different $\mathbf{z}^{\text{dyn}}, \mathcal{S}^{\text{dyn}}$. For the user study we sample 100 videos 200 frames long at 30 FPS. In order to compare different ablations, the assessors

[4] https://flickr.com

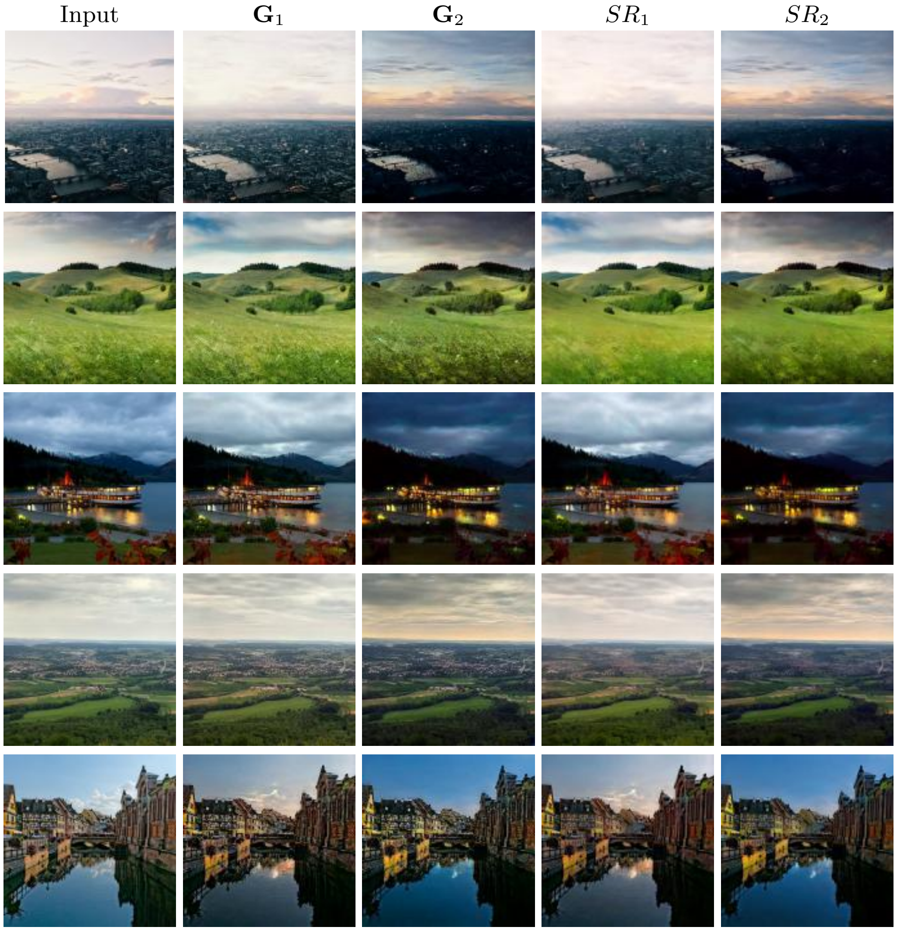


Fig. 5. Examples of super-resolution (SR) applied to the output of our generator ($\mathbf{G}$) given input image (Input). The inputs and SR are at 1024×1024 resolution, while the low-res images are at 256×256 resolution. Zoom-in recommended.

Setup	FID↓	SSIM↑	LPIPS↓	ΔR
Original StyleGAN	**48.40**	0.809	**0.049**	
+ frame discriminator	55.92	0.846	0.064	0.13
+ separate $\mathcal{S}^{\mathrm{st}}$ and $\mathcal{S}^{\mathrm{dyn}}$	55.15	0.854	0.073	0.01
+ separate $\mathbf{z}^{\mathrm{st}}$ and $\mathbf{z}^{\mathrm{dyn}}$	54.38	0.879	0.065	0.03
+ crop sampling	56.13	**0.884**	0.062	**0.06**

Fig. 6. Results of the ablation study of our model for the task of new video generation. The column ΔR in the table are obtained from the side-by-side user study. ΔR shows the increase in frequency when assessors prefer this variant to that in previous row (+0.23 against original StyleGAN).

were asked to select the most realistic video from a pair shown side-by-side. Each assessor is limited to evaluate no more than three pages with four tasks on each and has five minutes to complete each page. In our user study we showed each pair to five assessors. The ablation study results (Figure 6) reveal that the original StyleGAN generates the most high-fidelity images, but fails to preserve details of static objects. LPIPS is more tolerant to motion until the "texture type" changes dramatically. Thus, despite LPIPS and FID achieving the best values for the original StyleGAN, it actually does not preserve static objects (see **Supplementary Video**). Our modifications allow to keep a similar level of the FID value, but gradually improve static consistency and animation plausibility.

Real image animation. Experiments in this section are performed at 256×256 resolution. To calculate quantitative and qualitative metrics, we took the first frames I_0 of the test videos, encoded and animated them with our method. Denote the n-th frames of real and generated videos as I_n and $\widehat{I}_n$ respectively. With our method, for each input image we generate five variants with homographies randomly sampled from the predefined set. For AL we generate five videos for each input image with randomly sampled motion, as described in the original paper. For all quantitative evaluations we do not apply style transfer in AL and $\mathcal{W}$ manipulation in our method. We evaluate two variants of AL: with (AL) and without first-to-last interpolation ($\mathrm{AL}_{\mathrm{noint}}$), which stabilizes image quality, but makes long movements impossible. We use the official implementation[5] of Animating Landscape [2] provided by authors. We use pretrained AL model; we also evaluate finetuned AL model and found that most metrics degraded, while the training loss continued to improve. This can be attributed to the fact that the video dataset used in AL is bigger than ours; both include the public part of data from [14]. Both datasets are just youtube landscape videos and seem to be equally close to the validation (we are not aware of any biases). Also, our dataset contains videos with very different motion speed, and neither text of AL nor its code contains details regarding video speed equalization. All images are animated in original resolution cropped to 1:1 aspect ratio via center crop, then bilinearly downsampled to 256×256 resolution.

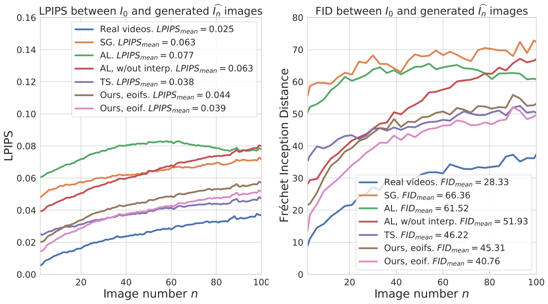
For SG [3] we used the official implementation[6] and default parameters. We have not noticed significant difference between multiple SG runs both in terms of quantitative metrics and visual diversity. Hence we decided not to generate similar videos many times and sampled only one video for each input image.

For TS [4] we used the official implementation[7]. TS can animate only the whole image, so (1) we used semantic segmentation to extract sky; (2) transferred motion to the extracted image fragment from a random video from the validation set; (3) blended static part of the original image with the generated clip. TS is only capable of producing 12 frames due to GPU memory limitations, so we interpolated frames in order to obtain the necessary video length.

[5] https://github.com/endo-yuki-t/Animating-Landscape

[6] https://github.com/tamarott/SinGAN

[7] https://github.com/ryersonvisionlab/two-stream-dyntex-synth



Name	FVD	LPIPS	SSIM	FID
SG	210	0.063	0.93	66.7
AL	275	0.077	0.91	61.9
AL_{noint}	162	0.063	0.92	52.4
TS	420	**0.039**	**0.96**	46.6
$\text{Ours}_{\text{eoifs}}$	161	0.044	0.94	45.8
$\text{Ours}_{\text{eoif}}$	**149**	**0.039**	0.95	**41.2**

Fig. 7. Quantitative comparison of image quality, static consistency and motion plausibility. **Left and middle**: LPIPS↓ and FID↓ between I_0 and $\widehat{I}_n$, which mostly measure image quality and static consistency. The legend contains metrics averaged over time. As can be seen, pixel-level transformations (e.g. using predicted flows in AL) lead to faster deterioration of generated images over time, compared to our approach, especially for later frames ($n \gtrsim 50$). **Right**: FVD↓, LPIPS↓, SSIM↑ and FID↓ between I_n and $\widehat{I}_n$ averaged over time, which measure not only image quality, but also animation plausibility.

We evaluate image quality by measuring FID between the set of all first frames of real videos I_0 and the set of n-th frames of generated videos $\widehat{I}_n$. Thus, we can see how fast these two distributions diverge. Too fast divergence in terms of FID may indicate image quality degradation in time. We evaluate static consistency by measuring LPIPS between I_0 and $\widehat{I}_n$ with moving parts masked out according to semantic segmentation. We always predict semantic segmentation only for I_0. Higher LPIPS may indicate that static areas are tampered during animation (i.e. they are erroneously moving). We also follow the adopted practice to quantitatively measure motion similarity using Fréchet Video Distance (FVD) [42] between real and generated videos, which is averaged over motion directions. Different motion directions are obtained via sampling different homography (Ours), motion code (AL) and horizontal flipping, choosing random reference video (TS). As revealed in Figure 7, our method preserves static details better and the speed of image quality degradation with time is slower than that of AL_{noint}.

The user study is carried out using the same real and generated videos as the ones used in quantitative evaluation. We decided to conduct two sets of user studies involving real image animation: side-by-side comparisons and real/fake questions. In the side-by-side setting, assessors are asked to select the more realistic variant of animation (from two) given the real image shown in the middle. Both videos in a pair are obtained from the same real image using different methods. In real/fake setting, assessors see only a single video and guess whether it is real or not. Each assessor was shown at most 12 questions, 5 different assessors per one question. During the study we noticed that the video speed affects user preference (slower ones are more favorable). Since we cannot

Method	**short**		**long**	
	EOIF	**EOIFS**	**EOIF**	**EOIFS**
SG	0.40	0.44	0.26	0.29
AL (no int)	0.46	0.47	0.37	0.38
AL (+ style)	0.18	0.18	0.11	0.10
TS	0.11	0.12	0.12	0.14
Real	0.41	0.44	0.44	0.45
Ours (EOIF)	–	0.52	–	0.52
Ours (EOIFS)	0.48	–	0.48	–

	FR
AL (+ style)	0.25
SG	0.38
Ours (Synth.)	0.42
AL (no int)	0.54
TS	0.20
Real	0.59
Ours (EOIFS)	0.62
Ours (EOIF)	**0.63**

Fig. 8. Left: Ratio of wins row-over-column for side-by-side settings for short (100 frames) and long (200 frames) videos. **Right**: fooling ratio for the real/fake protocol. Note that advantage of our method becomes more evident in long videos.

control animation speed in our baselines fairly, we decided to conduct two sets of user studies: (A) with motion speed aligned to that of competitors and (B) aligned to that of real videos. Here we present only results of A setting (see supp.mat. for B setting). To sum up, the user study reveals the advantage of our method over three baselines (AL, SG, TS), especially in longer videos.

Please refer to **Supplementary Materials** for more details on methods and experiments, including quantitative ablation study of inference procedure.

5 Discussion

We have presented a new generative model for landscape animations derived from StyleGAN, and have shown that it can be trained from the mixture of static images and timelapse videos, benefiting from both sources. We have investigated how the resulting model can be used to bring to life (reenact) static landscape images, and have shown that this can be done more successfully than with previously proposed methods. Extensive results of our method are shown in the supplementary video.

The supplementary video also shows failure modes. Being heavily reliant on machine learning, our approach fails when reenacting static images atypical for its training dataset. Furthermore, as our video dataset is relatively small and focuses on slower motions (clouds), we have found that method often fails to animate waves and grass sufficiently strongly or realistically. Enlarging the image dataset and, in particular, the video dataset seems to be the most straightforward way to address these shortcomings.

References

1. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 4401–4410

2. Endo, Y., Kanamori, Y., Kuriyama, S.: Animating landscape: Self-supervised learning of decoupled motion and appearance for single-image video synthesis. ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH Asia 2019) **38**(6) (2019) 175:1–175:19
3. Shaham, T.R., Dekel, T., Michaeli, T.: Singan: Learning a generative model from a single natural image. In: Proceedings of the IEEE International Conference on Computer Vision. (2019) 4570–4580
4. Tesfaldet, M., Brubaker, M.A., Derpanis, K.G.: Two-stream convolutional networks for dynamic texture synthesis. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (2017) 6703–6712
5. Srivastava, N., Mansimov, E., Salakhudinov, R.: Unsupervised learning of video representations using lstms. In: International conference on machine learning. (2015) 843–852
6. Villegas, R., Yang, J., Hong, S., Lin, X., Lee, H.: Decomposing motion and content for natural video sequence prediction. arXiv preprint arXiv:1706.08033 (2017)
7. Mathieu, M., Couprie, C., LeCun, Y.: Deep multi-scale video prediction beyond mean square error. CoRR **abs/1511.05440** (2015)
8. Finn, C., Goodfellow, I., Levine, S.: Unsupervised learning for physical interaction through video prediction. In: Advances in neural information processing systems. (2016) 64–72
9. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems. (2014) 2672–2680
10. Pan, J., Wang, C., Jia, X., Shao, J., Sheng, L., Yan, J., Wang, X.: Video generation from single semantic label map. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 3733–3742
11. Li, Y., Fang, C., Yang, J., Wang, Z., Lu, X., Yang, M.H.: Flow-grounded spatial-temporal video prediction from still images. In: ECCV. (2018)
12. Wang, T.C., Liu, M.Y., Zhu, J.Y., Liu, G., Tao, A., Kautz, J., Catanzaro, B.: Video-to-video synthesis. In: Advances in Neural Information Processing Systems (NeurIPS). (2018)
13. Aigner, S., Körner, M.: Futuregan: Anticipating the future frames of video sequences using spatio-temporal 3d convolutions in progressively growing autoencoder gans. arXiv preprint arXiv:1810.01325 (2018)
14. Xiong, W., Luo, W., Ma, L., Liu, W., Luo, J.: Learning to generate time-lapse videos using multi-stage dynamic generative adversarial networks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (June 2018)
15. Li, Y., Roblek, D., Tagliasacchi, M.: From here to there: Video inbetweening using direct 3d convolutions. ArXiv **abs/1905.10240** (2019)
16. Clark, A., Donahue, J., Simonyan, K.: Efficient video generation on complex datasets. ArXiv **abs/1907.06571** (2019)
17. Soomro, K., Zamir, A.R., Shah, M.: Ucf101: A dataset of 101 human actions classes from videos in the wild. (2012)
18. Carreira, J., Noland, E., Banki-Horvath, A., Hillier, C., Zisserman, A.: A short note about kinetics-600. ArXiv **abs/1808.01340** (2018)
19. Chen, B., Wang, W., Wang, J.: Video imagination from a single image with transformation generation. In: ACM Multimedia. (2017)
20. Van Amersfoort, J., Kannan, A., Ranzato, M., Szlam, A., Tran, D., Chintala, S.: Transformation-based models of video sequences. arXiv preprint arXiv:1701.08435 (2017)

21. Chuang, Y.Y., Goldman, D.B., Zheng, K.C., Curless, B., Salesin, D.H., Szeliski, R.: Animating pictures with stochastic motion textures. In: ACM SIGGRAPH 2005 Papers. SIGGRAPH '05, Association for Computing Machinery (2005) 853–860
22. Tulyakov, S., Liu, M.Y., Yang, X., Kautz, J.: Mocogan: Decomposing motion and content for video generation. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (2017) 1526–1535
23. Vondrick, C., Pirsiavash, H., Torralba, A.: Generating videos with scene dynamics. In: Advances in neural information processing systems. (2016) 613–621
24. Denton, E.L., et al.: Unsupervised learning of disentangled representations from video. In: Advances in neural information processing systems. (2017) 4414–4423
25. Zhu, J., Krähenbühl, P., Shechtman, E., Efros, A.A.: Generative visual manipulation on the natural image manifold. In Leibe, B., Matas, J., Sebe, N., Welling, M., eds.: Proc. ECCV. Volume 9909 of Lecture Notes in Computer Science., Springer (2016) 597–613
26. Abdal, R., Qin, Y., Wonka, P.: Image2stylegan: How to embed images into the stylegan latent space? In: Proceedings of the IEEE International Conference on Computer Vision. (2019) 4432–4441
27. Bau, D., Strobelt, H., Peebles, W.S., Wulff, J., Zhou, B., Zhu, J., Torralba, A.: Semantic photo manipulation with a generative image prior. ACM Trans. Graph. **38**(4) (2019) 59:1–59:11
28. Zakharov, E., Shysheya, A., Burkov, E., Lempitsky, V.: Few-shot adversarial learning of realistic neural talking head models. In: Proceedings of the IEEE International Conference on Computer Vision. (2019) 9459–9468
29. Huang, X., Belongie, S.: Arbitrary style transfer in real-time with adaptive instance normalization. In: Proceedings of the IEEE International Conference on Computer Vision. (2017) 1501–1510
30. Lars Mescheder, Andreas Geiger, S.N.: On the convergence properties of GAN training. CoRR **abs/1801.04406** (2018)
31. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 770–778
32. Kingma, D.P., Ba, J.L.: Adam: A method for stochastic optimization
33. Zhou, B., Zhao, H., Puig, X., Xiao, T., Fidler, S., Barriuso, A., Torralba, A.: Semantic understanding of scenes through the ade20k dataset. International Journal on Computer Vision (2018)
34. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: Computer Vision – ECCV 2016, Springer International Publishing (2016) 694–711
35. Wang, X., Yu, K., Wu, S., Gu, J., Liu, Y., Dong, C., Qiao, Y., Change Loy, C.: Esrgan: Enhanced super-resolution generative adversarial networks. In: Proceedings of the European Conference on Computer Vision (ECCV). (2018) 0–0
36. Wu, H., Zheng, S., Zhang, J., Huang, K.: Fast end-to-end trainable guided filter. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2018) 1838–1847
37. Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: Proceedings of the IEEE international conference on computer vision. (2017) 2223–2232
38. Yuan, L., Wen, F., Liu, C., Shum, H.Y.: Synthesizing dynamic texture with closed-loop linear dynamic system. In Pajdla, T., Matas, J., eds.: Computer Vision - ECCV 2004, Springer Berlin Heidelberg (2004) 603–616

39. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. In: NIPS. (2017)
40. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing **13**(4) (2004) 600–612
41. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2018) 586–595
42. Unterthiner, T., van Steenkiste, S., Kurach, K., Marinier, R., Michalski, M., Gelly, S.: Towards accurate generative models of video: A new metric & challenges. arXiv preprint arXiv:1812.01717 (2018)