StyleCineGAN: Landscape Cinemagraph Generation using a Pre-trained StyleGAN

Supplementary Material

A. Implementation Details

In this section, we provide details about the implementation of our method. Specifically, we explain each component of our method in detail, and provide training details about our encoder network, motion generator, and mask predictor.

A.a. Detailed explanations of each component

Mask Prediction We first predict the mask using an ensemble of 10 MLP classifiers [12], and refine it at inference to further improve the quality. To accomplish this, we compute contour areas in the mask using a contour detection algorithm [5]. Next, each area is considered to be holes and are removed if the ratio of its area with respect to the total area is less than 3%. This process effectively removes the noise from the predicted segmentation mask S as illustrated in Fig 1 (c).

Multi-Scale Deep Feature Warping We warp multi-scale features D_t^i, each of which corresponds to a different resolution. Given that the computed displacement fields $F_{0\to t}$ and $F_{N\to t}$ are at a fixed resolution of 512×512, it is necessary to resize them to match the dimensions of the respective deep features. This resizing operation is carried out using bilinear interpolation. Because the displacement fields represent pixel-level shifts, the values must be adjusted relative to the size of the features they are applied to. For instance, a 2-pixel shift at 256×256 resolution is equivalent to a movement twice the size of a 2-pixel shift at 512×512 resolution. To resolve this scaling discrepancy, we multiply scaling factors C_u and C_v to the u and v components of the displacement fields, respectively. The scaling factors are defined as $C_u = \frac{W}{512}$ and $C_v = \frac{H}{512}$, where W and H represent the width and height of the deep feature D_t^i.

Cinemagraph Generation While the multi-scale features warped with modified joint splatting cover most of the pixels, there can be some pixels with missing values. We apply a median filter to D_t^i to fill these small regions. Our final composited features D_t^i with missing values filled can be expressed as follows:

$$D_t^i(x') = (1 - H) \odot D_t^i(x') + H \odot \text{Median}(D_t^i(x')), \quad (1)$$

where H is a binary hole mask. The mask has a value of 1 for missing pixels and 0 for filled pixels. Median represents a median filter with a kernel of size 7×7. Large holes with more than 3% of the total area sometimes occur, and in this case a simple image inpainting method [7] is applied to D_t^i.

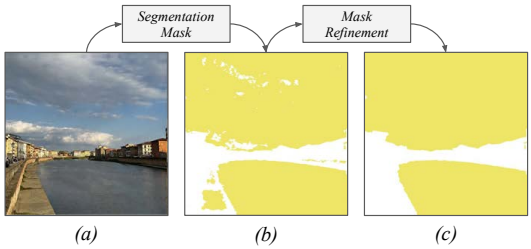

Figure 1. Mask prediction results. (a) Given an input image, (b) a mask is predicted by the mask predictor and (c) further refined by hole filling.

A.b. Training

Encoder Network To encode an input image into both latent codes and the deep features of StyleGAN, we train an encoder network using the architectures proposed in Yao et al. [10]. Images from the LHQ [4] dataset, 84,466 for training and 5,534 for validation, were used to train this encoder network. We trained the network for 12 epochs using the following loss function:

$$\mathcal{L}_{enc} = \mathcal{L}_2 + \lambda_{lpips}\mathcal{L}_{lpips} + \lambda_{reg}\mathcal{L}_{reg}, \quad (2)$$

where $\mathcal{L}_2$, $\mathcal{L}_{lpips}$, and $\mathcal{L}_{reg}$ are the reconstruction loss, perceptual loss, and feature regularization loss, respectively. λ_{lpips} and λ_{reg} are the weights for each loss term. We set $\lambda_{lpips} = 0.2$ and $\lambda_{reg} = 0.01$. We used the ADAM [3] optimizer with an initial learning rate of 10^{-4}, which resulted in a computation time of about 19 hours on an NVIDIA RTX A5000 GPU with a batch size of 1.

Motion Generator We trained an image-to-image translation network [8] as our motion generator. The motion generator was trained for 35 epochs, using the default parameters of Holynski et al. [2]. For sky motion, a set of 1,060 image-motion pairs from 430 unique videos of Sky Time-Lapse [9] were used for training. For fluid motion, a set of 4,895 image-motion pairs from 979 unique videos of the Eulerian [2] dataset are used. We resized all videos to a size of 512×512. To generate ground-truth motion fields, we use a pre-trained optical flow estimator [6]. The estimator calculates the average optical flow between consecutive frames within a 2-second window. The motion generator was trained with the ADAM [3] optimizer with an initial learning rate of 2×10^{-4}, which resulted in a computation time of about 6 hours on an NVIDIA GeForce RTX 3090 GPU with a batch size of 2.

Table 1. Human perceptual study results for assessing the occurrence of tearing artifacts. The best scores are bolded.

Method	Tearing Artifacts ↓
Ours - w/o DFW	4.37 ± 0.83
Ours - Full	$\mathbf{1.97 \pm 0.99}$

Mask Predictor We trained a MLP classifier as our mask predictor using the architectures proposed in Zhang et al. [12]. An ensemble of 10 MLP classifiers was trained using 32 input features paired with human-annotated segmentation masks. To construct the input feature D^*, we first project the annotated images into both D^{10} and w^+. We then feed D^{10} and w^+ to a pre-trained StyleGAN to extract the deep features D^i where $i \in \{10, 11, ..., 18\}$. These deep features are all resized to 512×512 and concatenated in the channel dimension. The constructed input feature is $D^* \in \mathbb{R}^{512\times512\times1472}$. We trained each classifier for 3 epochs with the ADAM [3] optimizer with an initial learning rate of 10^{-3}. Training all 10 MLP classifiers required about 15 hours of computation time on an NVIDIA Tesla V100 GPU with a batch size of 2.

B. Effectiveness of Deep Feature Warping

When warping is performed in image space, tearing artifacts are likely to occur for large pixel flows. To evaluate the effectiveness of our method for removing the tearing artifacts, in addition to qualitative and quantitative evaluations, we conducted a human perceptual study. The user study involved 19 participants who were presented with cinemagraph results and asked to score the occurrence of tearing artifacts. The ratings were made on a 1-to-5 scale, with "strongly disagree" being 1 and "strongly agree" being 5. The scores reported in Table 1 reveal that the application of DFW indeed removes the tearing artifacts, proving its effectiveness.

C. Selection of Deep Feature Index

Our method utilizes the deep features of a pre-trained StyleGAN for both the GAN inversion and cinemagraph generation process. Various deep features D^i where $i \in [1, 2, ..., 18]$ can be obtained from StyleGAN, thus we conducted a series of experiments to determine the proper index for our task.

C.a. Reconstruction and Warping Index

For each GAN inversion and DFW, a different feature index i can be used. Thus we explored various combinations of feature indices to assess which one led to the best perceptual quality. For GAN inversion, we employed feature indices $i \in [5, 6, ..., 13]$, while for DFW, we utilized feature indices $j \in [i, i+1, ..., 16]$. To assess the perceptual quality of the generated cinemagraphs, we measured LPIPS [11] between the generated frame $\hat{I}_n$ and the corresponding ground truth frame I_n using 224 test videos from the Sky Time-Lapse dataset [9]. Table 2 reveals the quantitative evaluation results on perceptual quality. The results show that the perceptual quality tends to improve as higher feature indices are employed for reconstruction. The highest perceptual quality is achieved when the latest feature index $j \in [6, 8, ..., 16]$ within the same StyleGAN block as the reconstruction index i is used for warping. Specifically, pairs (i, j) such as $(5, 6), (6, 6), (7, 8), (8, 8), ..., (15, 16)$, and $(16, 16)$ produce the highest perceptual quality for each reconstruction index i.

Table 2. Quantitative evaluation of perceptual quality according to reconstruction and warping indices. The best scores are bolded.

Warp Index \ Recon Index	5	6	7	8	9	10	11	12	13
5	.01335	-	-	-	-	-	-	-	-
6	**.01327**	**.01324**	-	-	-	-	-	-	-
7	.02033	.02388	.01284	-	-	-	-	-	-
8	.02042	.02388	**.01279**	**.01265**	-	-	-	-	-
9	.01882	.02199	.02105	.02475	.01258	-	-	-	-
10	.01877	.02199	.02107	.02478	**.01251**	**.01246**	-	-	-
11	.02049	.02418	.02363	.02761	.02612	.02717	.01240	-	-
12	.02054	.02420	.02362	.02761	.02608	.02729	**.01238**	**.01224**	-
13	.02098	.02324	.02319	.02609	.02535	.02869	.02083	.02759	.01248
14	.02094	.02323	.02319	.02603	.02535	.02676	.02075	.02759	**.01235**
15	.02277	.02432	.02429	.02708	.02729	.02859	.02348	.02835	.02533
16	.02322	.02460	.02453	.02731	.02736	.02861	.02360	.02840	.02530

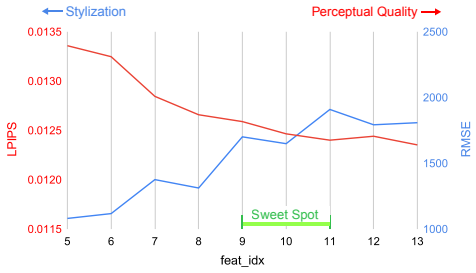

Figure 2. Trade-off between perceptual quality and stylization according to feature index i.

C.b. Perceptual Quality and Stylization Ability

We searched for the sweet spot in a trade-off between perceptual quality and stylization ability. We used 224 test videos from the Sky Time-Lapse dataset [9], utilizing the feature index pairs (i, j) with the highest perceptual quality described in Sec. C.a. To assess perceptual quality, we computed LPIPS between the generated frame $\hat{I}_n$ and its corresponding ground-truth frame I_n. To evaluate stylization ability, we computed RMSE between the Gram matrix [1] extracted from the target style image $G(w_t^+)$ and the generated frame $G_{warp}(D^{10}, w_s^+, F_{0\to t}, F_{N\to t})$. Target style latent w_t^+ was obtained from 16 color images from the RYB color model. We optimized w_t^+ while fixing StyleGAN, using the ADAM [3] optimizer with an initial learning rate of 0.1. Figure 2 shows the trade-off between perceptual quality and stylization ability: when we increase feature index i, the generated cinemagraphs exhibit higher perceptual quality but demonstrate less stylization ability. We selected feature index $i = 10$ considering this trade-off.

References

[1] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1, page 262–270, Cambridge, MA, USA, 2015. MIT Press. 2

[2] Aleksander Holynski, Brian L. Curless, Steven M. Seitz, and Richard Szeliski. Animating pictures with eulerian motion fields. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 5810–5819, 2021. 1

[3] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 1, 2

[4] Ivan Skorokhodov, Grigorii Sotnikov, and Mohamed Elhoseiny. Aligning latent and image spaces to connect the unconnectable. arXiv preprint arXiv:2104.06954, 2021. 1

[5] Satoshi Suzuki and KeiichiA be. Topological structural analysis of digitized binary images by border following. Computer Vision, Graphics, and Image Processing, 30(1): 32–46, 1985. 1

[6] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II, page 402–419, Berlin, Heidelberg, 2020. Springer-Verlag. 1

[7] Alexandru Telea. An image inpainting technique based on the fast marching method. Journal of Graphics Tools, 9(1): 23–34, 2004. 1

[8] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018. 1

[9] 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 The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018. 1, 2

[10] Xu Yao, Alasdair Newson, Yann Gousseau, and Pierre Hellier. A style-based gan encoder for high fidelity reconstruction of images and videos. European conference on computer vision, 2022. 1

[11] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In CVPR, 2018. 2

[12] Yuxuan Zhang, Huan Ling, Jun Gao, Kangxue Yin, Jean-Francois Lafleche, Adela Barriuso, Antonio Torralba, and Sanja Fidler. Datasetgan: Efficient labeled data factory with minimal human effort. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 10145–10155, 2021. 1, 2