Characterizing and Improving Stability in Neural Style Transfer

Agrim Gupta[1] Justin Johnson[1] Alexandre Alahi[1,2] Li Fei-Fei[1]
Stanford University[1] École Polytechnique Fédérate de Lausanne[2]

Abstract

Recent progress in style transfer on images has focused on improving the quality of stylized images and speed of methods. However, real-time methods are highly unstable resulting in visible flickering when applied to videos. In this work we characterize the instability of these methods by examining the solution set of the style transfer objective. We show that the trace of the Gram matrix representing style is inversely related to the stability of the method. Then, we present a recurrent convolutional network for real-time video style transfer which incorporates a temporal consistency loss and overcomes the instability of prior methods. Our networks can be applied at any resolution, do not require optical flow at test time, and produce high quality, temporally consistent stylized videos in real-time.

1. Introduction

Artistic style transfer of images aims to synthesize novel images combining the *content* of one image with the *style* of another. This longstanding problem [2, 9, 17, 18] has recently been revisited with deep neural networks [12]. Subsequent work has improved speed [22, 34], quality [35, 13, 36], and modeled multiple styles with a single model [8].

Recent methods for style transfer on images fall into two categories. Optimization-based approaches [12, 13] solve an optimization problem for each synthesized image; they give high-quality results but can take minutes to synthesize each image. Feedforward methods [22, 34, 8] train neural networks to approximate solutions to these optimization problems; after training they can be applied in real-time. However, all these methods are highly unstable resulting in visible flickering when applied to videos; see Figure 1. Ruder *et al*. [30] extends the optimization-based approach from images to videos. Their method produces high-quality stylized videos, but is too slow for real-time application.

In this paper, our goal is to perform feedforward style transfer of videos while matching the high-quality results produced by the optimization-based methods. Recent methods for style transfer use *Gram matrix* of features to represent image style: stylized images are synthesized by matching the Gram matrix of the style image. We find that the trace of the style image's Gram matrix is closely related to the pixel instability shown in Figure 1. Specifically, the solution set of the Gram matrix matching objective is a sphere with radius determined by the trace of the style image's Gram matrix. Due to the nonconvexity of this objective, minor changes in the content image can pull the synthesized image to different solutions of this Gram matrix matching objective function. If all the solutions are close together (small trace), then different solutions will still result in similar stylized images (no instability). However, if the solutions are far apart (large trace), then different solutions will result in very different stylized images (high instability).

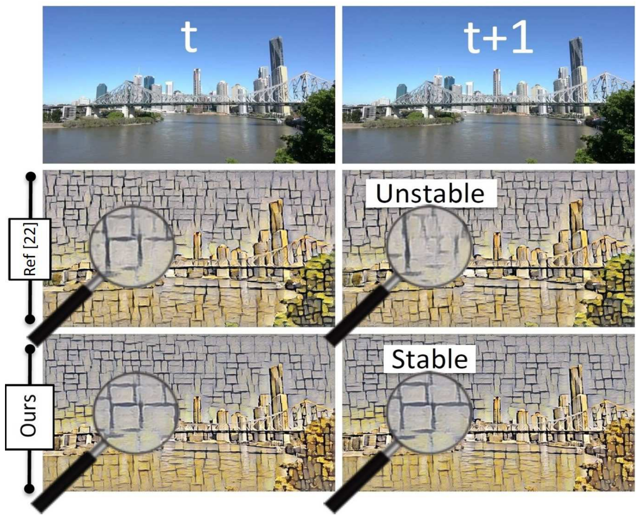


Figure 1. State-of-the-art real-time style transfer networks (e.g., [22]) are highly unstable. Consecutive video frames that are visually indistinguishable to humans (top) can result in perceptibly different stylized images (middle). In this work, we characterize such instability and propose a recurrent convolutional network that produces temporally consistent stylized video in real-time (bottom).

Based on this insight, we propose a method which greatly improves the stability of feedforward style transfer methods, synthesizing high-quality stylized video. Specifically, we use a recurrent convolutional network for video stylization, trained with a *temporal consistency loss* which encourages the network to find similar solutions to the Gram matrix matching objective at each time step.

Our contributions in this paper are twofold:

(i) First, we characterize the instability of recent style transfer methods by examining the solution set of the style transfer objective, showing an inverse relation between the trace of a style's Gram matrix and its stability. Our characterization applies to all neural style transfer methods based on Gram matrix matching.

(ii) Second, we propose a recurrent convolutional network for real-time video style transfer which overcomes the instability of prior methods. Inspired by [30], we incorporate a loss based on optical flow encouraging the network to produce temporally consistent results. Our method combines the speed of feedforward image stylization [22, 34, 8] with the quality and temporal stability of optimization-based video style transfer [30], giving a $1000\times$ speed improvement for stable video style transfer without sacrificing quality.

2. Related Work

Texture Synthesis. Texture synthesis is closely related to style transfer; the goal is to infer a generating process from an input texture to enable further production of samples of the same texture. Earlier attempts in computer vision to address the problem of texture synthesis can be divided into two distinct approaches: parametric and non-parametric. Parametric approaches compute global statistics in feature space and sample images from the texture ensemble directly [29, 42, 16, 6]. Non-parametric approaches estimate the local conditional probability density function and synthesize pixels incrementally. Methods based on this approach generate new textures either by re-sampling pixels [10, 38] or whole patches [25, 9] of the original texture.

Parametric methods are based on Julesz [23] characterization of textures where two images are said to have same texture if they have similar statistical measurements over a feature space. Gatys *et al.* [11] build on the seminal work by Portilla and Simoncelli [29] by using feature space provided by high performing neural networks and using Gram matrix as the summary statistic. In [35] the authors tackle the problem of perceptual quality in feed forward based texture synthesis by proposing instance normalization and a new learning formulation that encourages generators to sample unbiasedly from the Julesz texture ensemble. Chen and Schmidt [5] build on the work done in texture transfer by proposing a novel"style swap" based approach where they create the activations of the output image by swapping patches of the content image with the closest matching style activation patches. The swapped activation is then passed through a inverse network to generate styled image. Their optimization formulation is more stable than [22, 34] making it particularly suitable for video applications. Though their approach has generalization power [12] and is stable, it can't be used for real time video application due to runtime being of the order of seconds.

Style Transfer. Gatys et al. [12] showed that high quality images can be generated by using feature representations from high-performing convolutional neural networks. Their optimization based approach produces perceptually pleasing results but is computationally expensive. Johnson *et al.* [22] and Ulyanov *et al.* [34] proposed feed-forward networks which were a thousand times faster than [12] and could produce stylized images in real-time. However, each style requires training of a separate feed-forward network and the perceptual quality of the images is inferior compared to the optimization based approach. Dumoulin *et al.* [8] proposed a conditional instance normalization layer to address this issue, allowing one network to learn multiple styles. This results in a simple and efficient model which can learn arbitrarily different styles with considerably fewer parameters without compromising on speed or perceptual quality as compared to [22, 34].

Optical Flow. Accurate estimation of optical flow is a well studied problem in computer vision with variety of real-world applications. Classical approach for optical flow estimation is by variational methods based on the work by Horn and Schunck [19]. Convolutional neural networks (CNNs) have been shown to perform as good as state of the art optical flow detection algorithms. FlowNet [20, 7] matches the performance of variational methods and introduces novel CNN architectures for optical flow estimation. A full review of optical flow estimation is beyond the scope of this paper and interested readers can refer to [1, 33, 3]

Style Transfer on Videos. Artistic stylization of images is traditionally studied under the label of non-photorealistic rendering. Litwinowicz [27] was one of the first to combine the idea of transferring brush strokes from impressionist painting to images and using optical flow to track pixel motion across video frames to produce temporally coherent output video sequences. Hays and Essa [14] build on this to add further optical and spatial constraints to overcome flickering and scintillation of brush strokes in [27]. Hertzmann [17] improves on the perceptual quality of images by presenting techniques for painting an image with multiple brush sizes, and for painting with long, curved brush strokes and later extend this work to videos in [18].

Recently Ruder *et al.* [30] extended the optimization based approach in [12] by introducing optical flow based constraints which enforce temporal consistency in adjacent frames. They also proposed a multi-pass algorithm to ensure long term consistency in videos. Their algorithm produces extremely good results in terms of temporal consistency and per frame perceptual quality but takes a few minutes to process one frame.

3. Stability in Style Transfer

3.1. Style Transfer on Images

We use the style transfer formulation from [12], which we briefly review. Style transfer is an image synthesis task where we receive as input a *content image* c and a *style image* s. The output image p minimizes the objective

$$\mathcal{L}(s,c,p) = \lambda_c\, \mathcal{L}_c(p,c) + \lambda_s\, \mathcal{L}_s(p,s), \tag{1}$$

where $\mathcal{L}_c$ and $\mathcal{L}_s$ are the *content reconstruction loss* and *style reconstruction loss* respectively; λ_c and λ_s are scalar hyperparameters governing their importance.

Content and style reconstruction losses are defined in terms of a convolutional neural network ϕ; we use the VGG-19 [32] network pretrained on ImageNet. Let $\phi_j(x)$ be the j^{th} layer network activations of shape $C_j \times H_j \times W_j$ for image x. Given a set of content layers $\mathcal{C}$ and style layers $\mathcal{S}$, the content and style reconstruction losses are defined as:

$$\mathcal{L}_c(p,c) = \sum_{j\in\mathcal{C}} \frac{1}{C_j H_j W_j} \|\phi_j(p) - \phi_j(c)\|_2^2, \tag{2}$$

$$\mathcal{L}_s(p,s) = \sum_{j\in\mathcal{S}} \frac{1}{C_j H_j W_j} \|G(\phi_j(p)) - G(\phi_j(s))\|_F^2, \tag{3}$$

where $G(\phi_j(x))$ is a $C_j \times C_j$ *Gram matrix* for layer j activations given by $G(\phi_j(x)) = \Phi_{jx}\Phi_{jx}^T$, where Φ_{jx} is a $C_j \times H_j W_j$ matrix whose columns are the C_j-dimensional features of $\phi_j(x)$.

Rather than forcing individual pixels of output image to match content and style images, the content and style reconstruction losses encourage the generated image to match the high-level features of the content image and the feature correlations of the style image.

3.2. Gram Matrix and Style Stability

As shown in Figure 1, imperceptible changes in the content image c can result in drastically different stylized images p. However, we observe that this instability is not uniform across all styles. Some style, such as *Composition XIV* (see Figure 1) are highly unstable, while others such as *The Great Wave* (see Figure 9) show less qualitative instability.

To understand how instability depends on the style image, we consider only style loss for a single layer. Then the style transfer network minimizes the objective (dropping the subscript j for notational convenience):

$$\begin{aligned} \min_{G(\phi(p))} \quad & \frac{1}{CHW}\|G(\phi(p)) - G(\phi(s))\|_F^2, \\ \min_{\Phi_p} \quad & \|\Phi_p\Phi_p^T - \Phi_s\Phi_s^T\|_F^2. \end{aligned} \tag{4}$$

As motivation, consider the simple case $C = H = W = 1$; then Equation 4 reduces to $(\Phi_p^2 - \Phi_s)^2$, which is a nonconvex function with minima at $\Phi_p = \pm\Phi_s$, shown in Figure 2

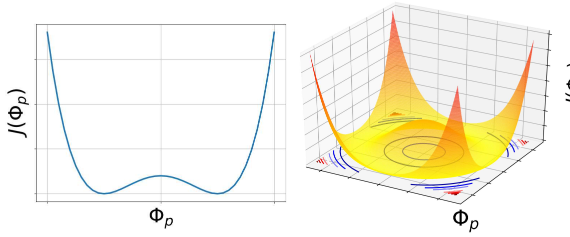


Figure 2. Visualization of the minimization objective in Equation 4 for 1-D and 2-D cases. For 1-D the two solutions differ by $2S$. Similarly, for 2-D case the solutions lie on a circle of radius S. This generalizes to the N-D case where the solutions lie on a N-D sphere centered at the origin with radius $\operatorname{tr}(G(\phi(p)))^{\frac{1}{2}}$.

(left). Similarly, when $C = H = 1, W = 2$, shown in Figure 2 (right), the minima lie on a circle of radius Φ_s. In both the cases, the minima lie at a distance Φ_s away from the origin. This observation holds in general:

Theorem 1. *Let γ be a sphere centered at the origin with radius $\operatorname{tr}(\Phi_s\Phi_s^T)^{\frac{1}{2}}$. Then, Φ_p minimizes the objective $J(\Phi_p) = \|\Phi_p\Phi_p^T - \Phi_s\Phi_s^T\|_F^2$ iff $\Phi_p \in \gamma$.*

Proof. Suppose that $\Phi_p = \Phi_s$; then $\Phi_p\Phi_p^T = \Phi_s\Phi_s^T$ and $J(\Phi_p) = 0$ so Φ_p minimizes the objective. Now let Φ_p be any minimizer of J; then $J(\Phi_p) = 0$, so $\Phi_p\Phi_p^T = \Phi_s\Phi_s^T$ and thus $\operatorname{tr}\Phi_p\Phi_p^T = \operatorname{tr}\Phi_s\Phi_s^T$ and so $\Phi_p \in \gamma$.

Now let $\Phi_p \in \gamma$. Since $J(\Phi_s) = 0$ we know $\Phi_s \in \gamma$; thus there exists an orthogonal rotation matrix U such that $\Phi_p = \Phi_s U$. Then we have $\Phi_p\Phi_p^T = \Phi_s UU^T\Phi_s^T = \Phi_s\Phi_s^T$, so $J(\Phi_p) = 0$ and thus Φ_p minimizes J. $\square$

This result suggests that styles for which the Gram matrix trace $\operatorname{tr}\Phi_s\Phi^T$ is large should exhibit more severe instability, since solutions to the style reconstruction loss will lie further apart in feature space as $\operatorname{tr}\Phi_s\Phi_s^T$ increases.

We empirically verify the relationship between $\operatorname{tr}\Phi_s\Phi_s^T$ and the stability of style transfer by collecting a small dataset of videos with a static camera and no motion; the only differences between frames are due to imperceptible lighting changes or sensor noise.

We then trained feedforward style transfer models on the COCO dataset [26] for twelve styles using the method of [22], and used each of these models to stylize each frame from our stable video dataset. Due to the static nature of the input videos, any difference in the stylized frames are due to the inherent instability of the style transfer models; we estimate the *instability* of each style as the average mean squared error between adjacent stylized frames. In Figure 3 we plot instability vs the trace of the Gram matrix for each style at the `relu1_1` and `relu2_1` layers of the VGG-16 loss network; these results show a clear correlation between the trace and instability of the styles.

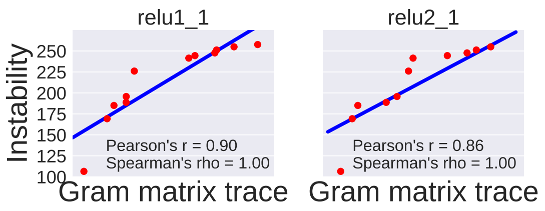


Figure 3. We train feedforward style transfer models for twelve styles, and define the *instability* of a style as the mean squared error between stylized adjacent frames over a dataset of videos with a static camera and no motion. We also compute the trace of the Gram matrix at two layers of the VGG-16 loss network for each style; styles with larger trace tend to be more unstable.

4. Method: Towards Stable Style Transfer

As shown above, feedforward networks for real-time style transfer can produce unstable stylized videos when the trace of the style's Gram matrix is large. We now present a feedforward style transfer method that overcomes this problem, matching the speed of [22] and the stability of [30].

4.1. Overall Architecture

Our method takes as input a sequence of content images $c_1, \ldots, c_T$ and a single style image s, and produces as output a sequence of stylized images $p_1, \ldots, p_T$. Each output image p_t should share content with c_t, share style with s, and be similar in appearance to p_{t-1}. At each time step, the output image p_t is synthesized by applying a learned *style transfer network* f_W to the previous stylized image p_{t-1} and the current content image c_t: $p_t = f_W(p_{t-1}, c_t)$.

Similar to [22, 34] we train one network f_W per style image s. The network is trained to minimize the sum of three loss terms at each time step:

$$\mathcal{L}(W, c_{1:T}, s) \tag{5}$$
$$= \sum_{t=1}^{T} (\lambda_c \mathcal{L}_c(p_t, c_t) + \lambda_s \mathcal{L}_s(p_t, s) + \lambda_t \mathcal{L}_t(p_{t-1}, p_t)),$$

where $\mathcal{L}_c$ and $\mathcal{L}_s$ are the content and style reconstruction losses from Section 3; $\mathcal{L}_t$ is *temporal consistency loss* which prevents the network output from drastically varying between time steps. The scalars λ_c, λ_s, and λ_t are hyperparameters weighting the importance of these terms. The network f_W is trained to minimize the combined loss in Equation 5 on a training dataset of video sequences $\{c_{1:T}\}$ via stochastic gradient descent.

4.2. Style Transfer Network

If our network is to produce temporally consistent outputs, then it cannot process frames independently; it must have the capacity to examine its own previous outputs to ensure consistency. Our networks therefore take as input both the current content image c_t and the stylized result from the previous frame $p_t = f_W(p_{t-1}, c_t)$. As shown in Figure 4, the output from the network at each time step is fed as input to the network at the next time step. The network f_W is therefore a *recurrent convolutional network*, and must be trained via backpropagation through time [31, 40].

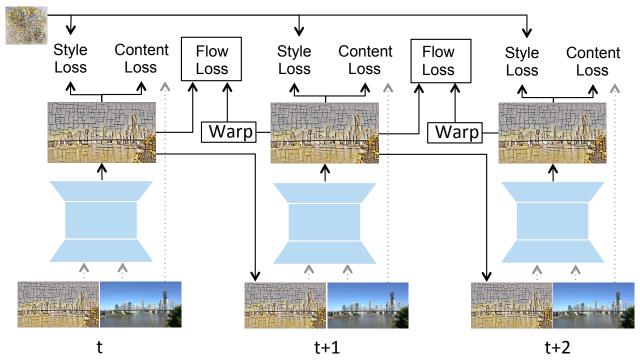


Figure 4. System overview. Our style transfer network takes as input the previous stylized image p_{t-1} and the current video frame c_t, and produces a stylized version of the frame. The output at each timestep is fed as input at the next time step, so our system is a *recurrent convolutional network*. At each time step we enforce style and content losses to ensure similarity with the input frame and style image; between each consecutive frame we enforce a *temporal consistency loss* which encourages temporally stable results.

The two inputs to f_W are concatenated along the channel dimension, after which the architecture of f_W follows [8]: it is a deep convolutional network with two layers of spatial downsampling followed by several residual blocks [15] and two layers of nearest-neighbor upsampling and convolution. All convolutional layers are followed by instance normalization [35] and ReLU nonlinearities.

4.3. Temporal Consistency Loss

By design our style transfer network can examine its own previous outputs, but this architectural change alone is not enough to ensure temporally consistent results. Therefore, similar to Ruder *et al.* [30] we augment the style and content losses with a *temporal consistency loss* $\mathcal{L}_t$ encouraging temporally stable results by penalizing the network when its outputs at adjacent time steps significantly vary.

The simplest temporal consistency loss would penalize per-pixel differences between output images: $\mathcal{L}_t(p_{t-1}, p_t) = \|p_{t-1} - p_t\|^2$. However, for producing high-quality stylized video sequences we do not want stylized video frames to be exactly the same between time steps; instead we want brush strokes, lines, and colors in each stylized frame to transfer to subsequent frames in a manner consistent with the motion in the input video.

To achieve this our temporal consistency loss utilizes *optical flow* to ensure that changes in output frames are consistent with changes in input frames. Concretely, let $\mathbf{w} = (u, v)$ be the (forward) optical flow field between input frames c_{t-1} and c_t. Perfect optical flow gives a pix-

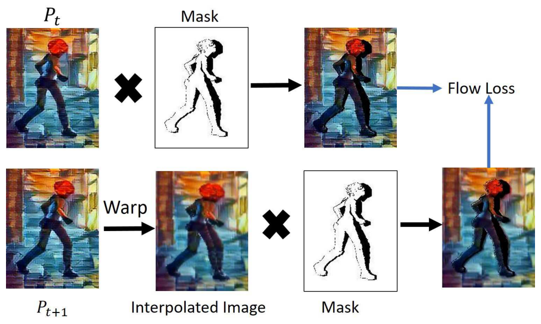


Figure 5. The temporal consistency loss $\mathcal{L}_t(p_{t-1}, p_t)$ warps p_t using optical flow, giving a warped frame $\tilde{p}_t$. The previous frame p_{t-1} and warped frame $\tilde{p}_t$ are multiplied by an *occlusion mask*; $\mathcal{L}_t(p_{t-1}, p_t)$ is then the Euclidean distance between p_{t-1} and $\tilde{p}_t$.

elwise correspondence between c_t and c_{t-1}; we want the corresponding pixels of p_t and p_{t-1} to match. Therefore the temporal consistency loss penalizes the difference

$$p_{t-1}(x, y) - p_t(x + u(x, y), y + v(x, y)) \tag{6}$$

for all pixel coordinates (x, y). This difference can be efficiently implemented by *warping* the output frame p_t using the optical flow to give a warped frame $\tilde{p}_t$, then computing the per-pixel differences between $\tilde{p}_t$ and p_{t-1}. The use of bilinear interpolation makes this warping differentiable [21].

Due to foreground object motion, some pixels in c_{t-1} may become occluded in c_t; likewise some pixels which are occluded in c_{t-1} may become disoccluded in c_t. As a result, enforcing the temporal consistency loss between *all* pixels of $\tilde{p}_t$ and p_{t-1} would result in artifacts at motion boundaries. We therefore use a ground-truth occlusion mask m to avoid enforcing the temporal consistency loss for occluded and disoccluded, giving our final temporal consistency loss:

$$\mathcal{L}_t(p_{t-1}, p_t) = \frac{1}{HW} \|m_t \odot p_{t-1} - m_t \odot \tilde{p}_t\|_F^2, \tag{7}$$

where $m(h, w) \in [0, 1]$ is 0 in regions of occlusion and motion boundaries, $\odot$ is elementwise multiplication, and H, W are the height and width of the input frame. This loss function is summarized in Figure 5.

Computing this loss requires optical flow and occlusion masks; however since this loss is only applied during training, our method does not require computing optical flow or occlusion masks at test time.

4.4. Implementation Details

Following [22, 8] we first train an image style transfer network on the COCO dataset [26] using only the content and style losses $\mathcal{L}_c$ and $\mathcal{L}_s$. We then finetune the model using all three losses on the Sintel Dataset [4, 41]; Sintel consists of rendered images and thus provides pixel-perfect optical flow and occlusion masks. Finetuning rather than training from scratch allows for a more controlled comparison with previous methods for feedforward image stylization. During training we resize all video frames to 512×218 and use random horizontal and vertical flips for data augmentation; we train for 10 epochs with BPTT for 4 time steps using Adam [24] with learning rate 1×10^{-3}.

Style	Real-Time Baseline [22]	Optim Baseline [30]	Ours
The Wave	24.3 / 0.47	**25.5** / 0.48	24.8 / **0.54**
Metzinger	23.6 / 0.31	**24.4** / 0.37	24.2 / **0.42**
Composition XIV	23.8 / 0.31	24.0 / 0.38	**24.2** / **0.42**
Mosaic	23.7 / 0.31	**24.4** / 0.37	24.0 / **0.39**
Rain Princess	23.8 / 0.41	**25.2** / 0.45	24.4 / **0.49**

Table 1. We evaluate the stability of each method by finding corresponding 100×100 background patches between adjacent video frames from the DAVIS dataset, then computing PSNR / SSIM between these patches in the stylized versions of the frames. We report mean PSNR / SSIM between stylized corresponding patches across all frames from all videos of the DAVIS dataset. Our method is generally more stable than the Real-Time baseline, and has comparable stability to the Optim baseline.

5. Experiments

Our experiments show that our method results in stylized video sequences with comparable image quality and stability as optimization-based methods [30], while retaining the speed advantage of feedforward methods [22, 34].

5.1. Baselines

We compare our method with two state-of-the-art approaches for image and video stylization.

Real-Time Baseline [22, 8]. We train feedforward networks for image stylization and apply the resulting network to each video frame independently. This method allows for high-quality image stylization in real-time, but leads to severe temporal instability when applied to videos.

Optim Baseline [30]. This method explicitly minimizes an objective function to synthesize stylized frames, leading to very stable, high-quality results, but requiring minutes of processing per video frame. We use the single-pass algorithm from [30]; their multipass algorithm improves long-term temporal consistency at the cost increased runtime. We use the open-source code released by the authors of [30].

5.2. Datasets

Sintel. The Sintel Dataset [4, 41] consists of 35 video sequences from an animated movie, divided into a training set of 23 videos and a test set of 12 videos; each sequence has between 20 and 50 frames. We use Sintel for training since it provides pixel-perfect optical flow and occlusion masks; we show qualitative results on the Sintel test set.

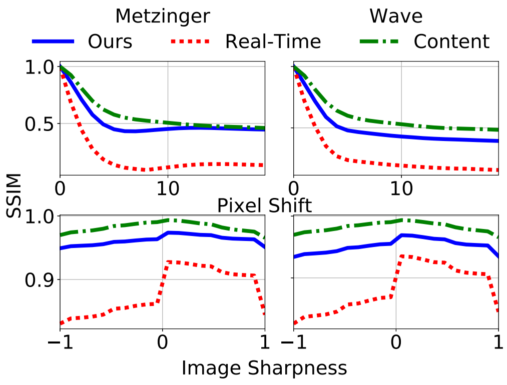


Figure 6. We measure the stability of our method with respect to controlled image distortions by taking an image patch, distorting it, then computing SSIM between the original and distorted patch (Content); we then stylize both the original and distorted patches using our method and the Real-Time baseline, and compute SSIM between the stylized original and stylized distorted patches. Varying the magnitude of the distortion measures the stability of the method with respect to that distortion. We consider shifting the patch by 0 to 19 pixels in the input frame (top) and applying blur and sharpening kernels to the patch with varying strength (bottom); repeating the experiment for two styles (Metzinger, left and Wave, right). All results are averaged over random 256 × 256 background patches from 100 random frames from DAVIS videos. Our method is much more robust to these controlled distortions.

DAVIS. The DAVIS dataset [28] comprises 50 real-world video sequences with an average of 69 frames per sequence. These videos include effects such as occlusion, motion-blur, appearance change, and camera motion, making it a challenging benchmark for video stylization. Each video frame is annotated with a ground-truth foreground/background segmentation mask. We use this dataset for qualitatively and quantitatively evaluating all methods.

5.3. Quantitative Evaluation

Controlled Distortions. If a style transfer method is stable, then small changes to its input should result in modest changes to its output. Before moving to unconstrained videos, we can first measure the stability of our method in a more controlled setting using artificial image distortions.

Given an image patch c, we apply a distortion d to give a distorted patch $d(c)$. We measure SSIM between c and $d(c)$, then apply a trained style transfer network f_W to both c and $d(c)$, and measure SSIM between $f_W(c)$ and $f_W(d(c))$; we then repeat the process, varying the magnitude of the distortion d. If the network f_W is robust to the distortion, then as the magnitude of d increases the image similarity between $f_W(c)$ and $f_W(d(c))$ should decay at the same rate as the similarity between c and $d(c)$.

In Figure 6 we show the results of this experiment for two types of distortion: translation and blurring / sharpening, comparing our method against the Real-Time baseline on two styles. In all cases, as distortion magnitude increases our method shows a decay in image similarity comparable to that between c and $d(c)$; in contrast the image similarity for the baseline decays sharply. This shows that compared to the baseline, our method is significantly more robust to controlled distortions.

Image-Size	Real-Time Baseline [22]	Optim Baseline [30]	Ours	Speedup vs [30]
256 × 256	0.024	22.14	0.024	922x
512 × 512	0.044	59.64	0.044	1355x
1024 × 1024	0.141	199.6	0.141	1415x

Table 2. Speed (in seconds) for our method vs the two baseline methods for processing a single video frame for varying resolutions. For the Optim baseline we use the default of 1000 iterations. Our method generates stylized video matching the speed of real-time baseline method and the temporal consistency of the Optim baseline. All methods are benchmarked on a Titan X Pascal GPU.

Video Stability. We aim to show that our method can stylize real-world videos, matching the stability of the optim baseline. The instability of the real-time baseline typically manifests most strongly in background image regions with relatively little motion. To quantitatively measure this phenomenon, we use the foreground/background masks from the DAVIS dataset.

For each video sequence in the DAVIS dataset, we find corresponding 100 × 100 pixel patches in adjacent frames that do not intersect the foreground object as follows: We first choose a random background patch at time t, then find the patch at time $t + 1$ which is within 20 pixels of the first patch and maximizes the PSNR between the two patches. We then compute PSNR and SSIM [37] between the stylized versions of these corresponding patches, and report the average PSNR and SSIM across all methods, videos[1], and across 5 styles; results are shown in Table 1.

This experiment quantifies the instability of the real-time baseline, and shows that our method produces stylized videos with stability comparable to the Optim baseline.

Speed. Table 2 compares the runtime of our method with the baselines for several image sizes; for the Optim baseline we exclude the time for computing optical flow.

At test-time our method produces temporally consistent stylized video frames without explicitly computing optical flow. It matches the speed of the real-time baseline; both are roughly three orders of magnitude faster than the Optim baseline. Our method can process images of size 512 × 512 at 20 FPS, making it feasible to run in real-time.

[1]Running the Optim baseline for all videos and styles would take approximately 25 days on a GPU, which is computationally infeasible. Therefore, for this method we select three random videos for each style.

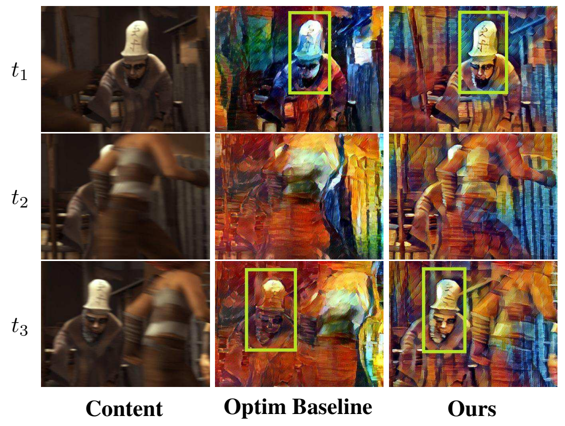


Figure 7. Video sequence where our method performs better than the Optim baseline. We consider three consecutive video frames. The man gets occluded in t_2 and on dis-occlusion in t_3 the style of the man is different from t_1 for Optim baseline. In case of our method the styles remains the same.

5.4. Qualitative Evaluation

Short-Term Consistency. Figure 9 shows patches from adjacent frames of stylized videos from the DAVIS dataset for different styles. The real-time baseline method is unable to produce the same stylization for background regions across consecutive video frames; these sudden changes between adjacent frames manifest as a severe "flickering" effect in video. In contrast, the Optim baseline and our method both result in consistent stylization of patches across adjacent frames, eliminating this flickering.

Figure 9 also showcases the dependence between Gram matrix trace and style instability. Both *Portrait de Metzinger* and *Rain Princess* style images have very high Gram matrix traces, causing the instability in the real-time baseline. The trace of the Gram matrix for *The Great Wave* is much smaller, and correspondingly the real-time baseline shows less instability on this style.

Long-Term Stability. One challenging problem in video stylization is *long-term stability*: When an object is occluded and then reappears, it should have the same style. Although we do not explicitly enforce long-term consistency in our loss, we find that our method nevertheless sometimes shows better long-term consistency than the Optim baseline; see Figure 7. In this example the man is visible at t_1, is occluded by the girl at t_2, and reappears at t_3. In our method the man has similar appearance at t_1 and t_3, but the baseline "smears" the red color of the girl onto the man in t_3. The authors of [30] also propose a multi-pass version of their algorithm which explicitly addresses this issue but it requires at least 10 passes over the video.

Dependency on Optical Flow. The Optim baseline requires forward and backward optical flow at test time to generate occlusion masks; failures in optical flow estimation can result in poor stylization.

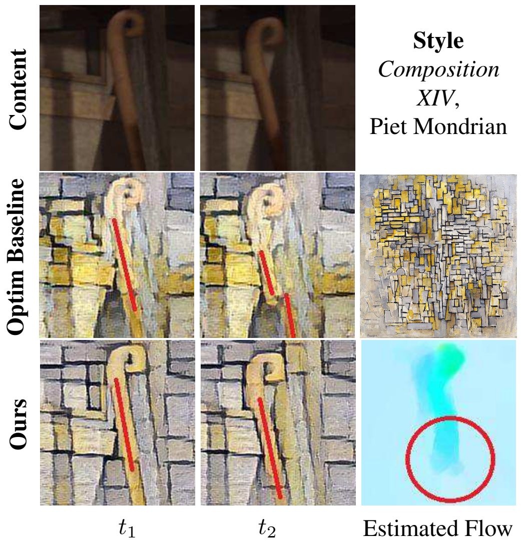


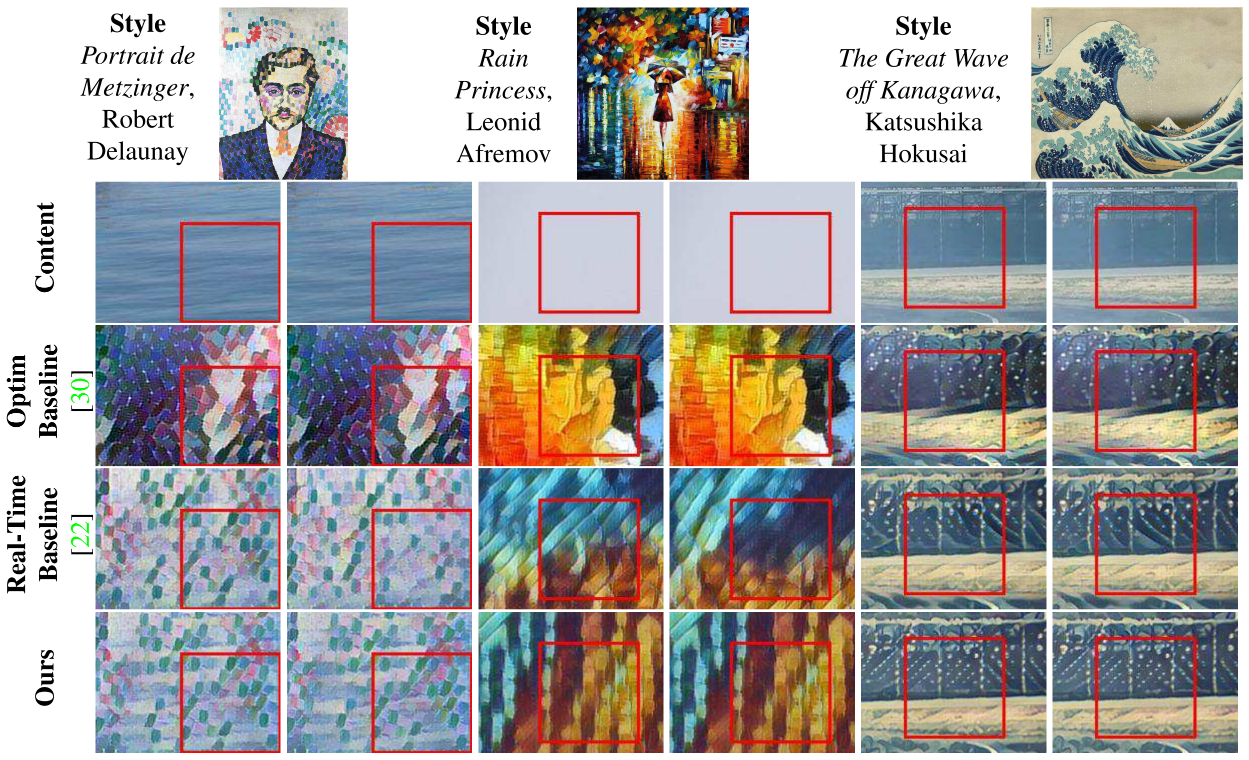
Figure 8. Optim baseline is susceptible to errors in optical flow estimation. The estimated optical flow does not capture the motion of the bottom portion of the stick (Bottom-left: Dark blue represents more motion). Hence, at time t_2 the stylized image produced by Optim baseline shows the stick to be "broken" at the top. Our method does not require explicit optical flow at test time and does suffer from this failure mode (red lines added for emphasis).

Figure 8 shows a crop of two frames from the *Market_1* video from the Sintel test set. Figure 8 (bottom-right) shows the optical field as estimated by state-of-the-art optical flow algorithm [39]. The estimated optical flow incorrectly estimates that only the top portion of the stick is moving. This error propagates to the stylized image: unlike the original pair of images where the whole stick moves, only the top portion of the stick to moves forward in the output produced by the Optim baseline. Our method does not require ground truth optical flow during test time and consequently does not suffer from this failure mode.

User Study. We performed a user study on Amazon Mechanical Turk to compare the subjective quality of our method and the two baselines. In each trial a worker is shown a video from the DAVIS dataset [28], a style image, and stylized output videos from two methods. In each trial the worker answers three questions: *"Which video flickers more?"*, *"Which video better matches the style?"*, and *"Overall, which video do you prefer?"*. For each question, the worker can either choose a video or select *"About the same"*. Results are shown in Table 3.

Taken as a whole, this user study shows that our method results in videos with significantly less qualitative flickering than the Real-Time baseline, with temporal stability almost on par with the slower Optim baseline. Our method is perceived to match the style image about as well as other

[3]Again, running the Optim baseline for all 50 videos is infeasible.



Style
Portrait de Metzinger, Robert Delaunay
Style
Rain Princess, Leonid Afremov
Style
The Great Wave off Kanagawa, Katsushika Hokusai
Content
Optim Baseline [30]
Real-Time Baseline [22]
Ours

Method	PSNR / SSIM	PSNR / SSIM	PSNR / SSIM
[30]	23.67 / 0.2078	**23.68** / 0.9136	**32.53 / 0.9728**
[22]	23.47 / 0.1856	23.13 / 0.1622	26.00 / 0.7432
Ours	**24.14 / 0.5234**	23.44 / **0.9240**	27.76 / 0.9194

Figure 9. Examples of consecutive pairs of stylized video frame output. Our method produces stylistically similar frame sequences like Optim baseline. We report the PSNR/SSIM values for each example crop shown. Our method is significantly better in producing temporally consistent frames than real-time baseline for highly unstable styles like *Rain Princess* and *Metzinger*.

	Ours vs RT [22]		Ours vs Optim [30]	
Question	Ours	RT	Ours	Optim
More Flicker	**26**	193	6	**4**
Style Match	70	**114**	**8**	4
Overall Prefer	**133**	80	7	**8**

Table 3. Summary of user study results. We use 50 videos per style to evaluate our method against the Real-Time (RT) baseline [22] and 3 videos[3] to evaluate against the Optim baseline [30]. Each pair of videos is evaluated by five workers on Amazon Mechanical Turk. We report the number of videos where the majority of workers preferred one method over another for a particular question. Values in bold are better.

methods, and users prefer the results from our method significantly more often than the Real-Time baseline. We refer the reader to Supplementary material for further details.

Failure Cases. Our method sometimes results in object transparency when one object occludes another; for an example see Supplementary Material. However, this effect typically occurs for the first frame or two of object occlusion, and is thus not very perceptible when viewing videos.

Our system sometimes suffers from shower-door artifacts; we hypothesize that the network learns a motion bias from the Sintel dataset, since most frames exhibit global camera motion. We have found that careful tuning of λ_t and pretraining on the synthetic Flying Chairs dataset [7] (which has static backgrounds) help reduce this effect.

6. Conclusion

We studied the stability of the recent style transfer methods based on neural networks. We characterized the instability of style transfer methods based on Gram matrix matching by examining the solution set of the style transfer objective, arguing that instability is exacerbated when the trace of the Gram matrix of the style image is large.

We then proposed a recurrent convolutional network for real-time video style transfer which overcomes the instability of prior methods. As future work, we want to investigate methods which can encode long-term consistency while maintaining real-time performance, ensuring stylistic consistency of objects which get occluded and then reappear over a sequence of consecutive video frames.

Acknowledgments. We thank Jayanth Koushik for helpful comments and discussions. This work is partially supported by an ONR MURI grant.

References

[1] J. Aggarwal and N. Nandhakumar. On the computation of motion from sequences of images-a review. *Proceedings of the IEEE*, 76(8):917–935, 1988. 2

[2] M. Ashikhmin. Synthesizing natural textures. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 217–226. ACM, 2001. 1

[3] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *IJCV*, 1994. 2

[4] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *ECCV*, 2012. 5

[5] T. Q. Chen and M. Schmidt. Fast patch-based style transfer of arbitrary style. *CoRR*, abs/1612.04337, 2016. 2

[6] J. S. De Bonet. Multiresolution sampling procedure for analysis and synthesis of texture images. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 361–368. ACM Press/Addison-Wesley Publishing Co., 1997. 2

[7] A. Dosovitskiy, P. Fischer, , E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. FlowNet: Learning optical flow with convolutional networks. In *ICCV*, 2015. 2, 8

[8] V. Dumoulin, J. Shlens, and M. Kudlur. A learned representation for artistic style. In *ICLR*, 2017. 1, 2, 4, 5

[9] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 341–346. ACM, 2001. 1, 2

[10] A. A. Efros and T. K. Leung. Texture synthesis by nonparametric sampling. In *ICCV*, 1999. 2

[11] L. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In *NIPS*, 2015. 2

[12] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016. 1, 2, 3

[13] L. A. Gatys, A. S. Ecker, M. Bethge, A. Hertzmann, and E. Shechtman. Controlling perceptual factors in neural style transfer. In *CVPR*, 2017. 1

[14] J. Hays and I. Essa. Image and video based painterly animation. In *Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering*, pages 113–120. ACM, 2004. 2

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 4

[16] D. J. Heeger and J. R. Bergen. Pyramid-based texture analysis/synthesis. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 229–238. ACM, 1995. 2

[17] A. Hertzmann. Painterly rendering with curved brush strokes of multiple sizes. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 453–460. ACM, 1998. 1, 2

[18] A. Hertzmann and K. Perlin. Painterly rendering for video and interaction. In *Proceedings of the 1st international symposium on Non-photorealistic animation and rendering*, pages 7–12. ACM, 2000. 1, 2

[19] B. K. Horn and B. G. Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981. 2

[20] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. In *CVPR*, 2017. 2

[21] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *NIPS*, 2015. 5

[22] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016. 1, 2, 3, 4, 5, 6, 8

[23] B. Julesz. Visual pattern discrimination. *IRE transactions on Information Theory*, 8(2):84–92, 1962. 2

[24] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 5

[25] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick. Graphcut textures: image and video synthesis using graph cuts. In *ACM Transactions on Graphics (ToG)*, volume 22, pages 277–286. ACM, 2003. 2

[26] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 3, 5

[27] P. Litwinowicz. Processing images and video for an impressionist effect. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 407–414. ACM Press/Addison-Wesley Publishing Co., 1997. 2

[28] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. V. Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *CVPR*, 2016. 6, 7

[29] J. Portilla and E. P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *IJCV*, 2000. 2

[30] M. Ruder, A. Dosovitskiy, and T. Brox. Artistic style transfer for videos. In *German Conference on Pattern Recognition*, pages 26–36. Springer, 2016. 1, 2, 4, 5, 6, 7, 8

[31] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Technical report, DTIC Document, 1985. 4

[32] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 3

[33] D. Sun, S. Roth, and M. J. Black. A quantitative analysis of current practices in optical flow estimation and the principles behind them. *IJCV*, 2014. 2

[34] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *ICML*, 2016. 1, 2, 4, 5

[35] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *CVPR*, 2017. 1, 2, 4

[36] X. Wang, G. Oxholm, D. Zhang, and Y.-F. Wang. Multimodal transfer: A hierarchical deep convolutional neural network for fast artistic style transfer. In *CVPR*, 2017. 1

[37] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 6

[38] L.-Y. Wei and M. Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 479–488. ACM Press/Addison-Wesley Publishing Co., 2000. 2

[39] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. Deepflow: Large displacement optical flow with deep matching. In *ICCV*, 2013. 7

[40] P. J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990. 4

[41] J. Wulff, D. J. Butler, G. B. Stanley, and M. J. Black. Lessons and insights from creating a synthetic optical flow benchmark. In A. Fusiello et al. (Eds.), editor, *ECCV Workshop on Unsolved Problems in Optical Flow and Stereo Estimation*, Part II, LNCS 7584, pages 168–177. Springer-Verlag, Oct. 2012. 5

[42] S. C. Zhu, X. W. Liu, and Y. N. Wu. Exploring texture ensembles by efficient markov chain monte carlo-toward a" trichromacy" theory of texture. *TPAMI*, 2000. 2