# Artistic style transfer for videos and spherical images

**Manuel Ruder · Alexey Dosovitskiy · Thomas Brox**

**Abstract** Manually re-drawing an image in a certain artistic style takes a professional artist a long time. Doing this for a video sequence single-handedly is beyond imagination. We present two computational approaches that transfer the style from one image (for example, a painting) to a whole video sequence. In our first approach, we adapt to videos the original image style transfer technique by Gatys et al. based on energy minimization. We introduce new ways of initialization and new loss functions to generate consistent and stable stylized video sequences even in cases with large motion and strong occlusion. Our second approach formulates video stylization as a learning problem. We propose a deep network architecture and training procedures that allow us to stylize arbitrary-length videos in a consistent and stable way, and nearly in real time. We show that the proposed methods clearly outperform simpler baselines both qualitatively and quantitatively. Finally, we propose a way to adapt these approaches also to 360 degree images and videos as they emerge with recent virtual reality hardware.

**Keywords** Style transfer · Deep networks · Artistic videos · Video stylization

## 1 Introduction

The seminal work of Gatys et al. [5] that transfers the artistic style of a painting to real world photographs has opened up an interesting application area for deep networks and has triggered lots of follow-up works. Their approach uses the statistics of high-level feature representations of the images from the hidden layers of an image classification network to separate and reassemble content and style. This is done by formulating an optimization problem that, starting with white noise, searches for a new image showing similar neural activations as the *content image* and similar feature correlations (expressed by a Gram matrix) as the *style image*. Based on this, Johnson et al. [12] demonstrated that style transfer, conceived as an image transformation problem, can be learned by a convolutional neural network, which drastically reduces the runtime to process an image.

This paper presents extensions of the works from Gatys et al. [5] and Johnson et al. [12] to video sequences. Given an artistic image, we transfer its particular style of painting to the entire video. Processing each frame of the video independently leads to flickering and false discontinuities, since the solution of the style transfer task is not stable. To regularize the transfer and to preserve smooth transition between individual frames of the video, we introduce a temporal constraint that penalizes deviations between two successive frames. The temporal constraint takes the optical flow from the original video into account: instead of penalizing the deviations from the previous frame, we penalize deviation along the point trajectories. Disoccluded regions as well as motion boundaries are excluded from the penalizer. This allows the process to rebuild disoccluded regions and distorted motion boundaries while preserving the appearance of the rest of the image; see Fig. 1.

Adding a loss that prefers temporal consistency between successive frames alone still leads to visible artifacts in the resulting videos. Thus, we present two extensions to our basic approach. The first one aims on improving the consistency over larger periods of time.

This study was partially supported by the Excellence Initiative of the German Federal and State Governments EXC 294

M. Ruder, A. Dosovitskiy, T. Brox
Department of Computer Science and BIOSS Centre for Biological Signalling Studies, University of Freiburg
E-mail: {rudera,dosovits,brox}@cs.uni-freiburg.de

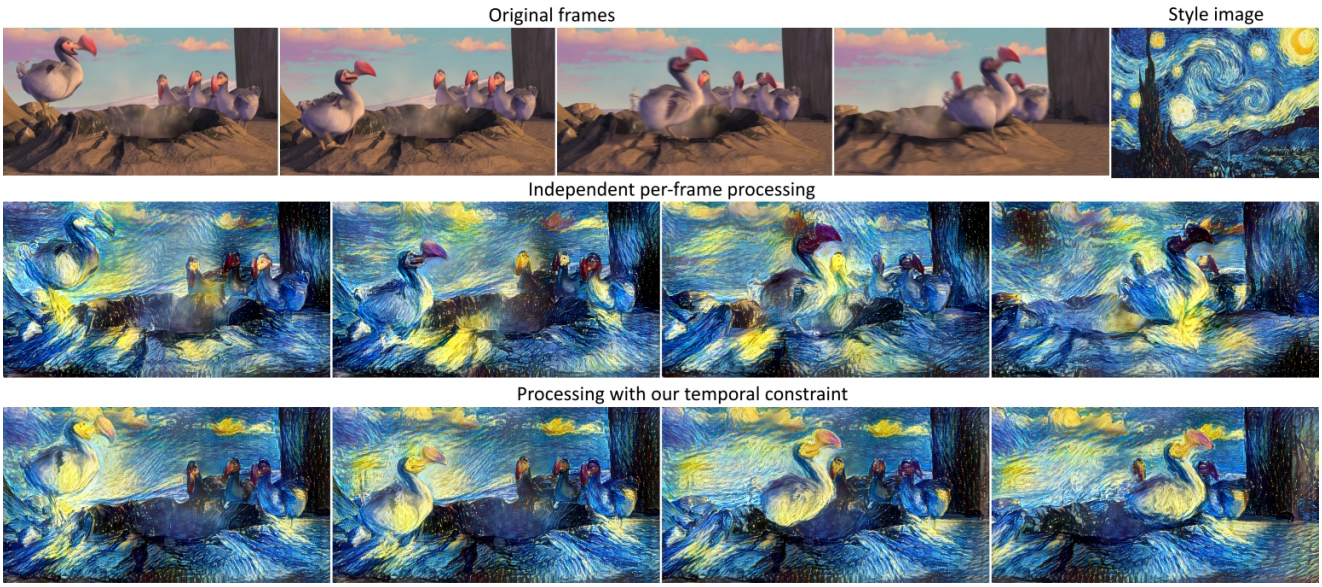


**Fig. 1** Scene from *Ice Age* (2002) processed in the style of *The Starry Night*. Comparing independent per-frame processing to our temporally consistent approach, the latter is clearly preferable. Best observed in the supplemental video, see `https://youtu.be/SKql5wkWz8E`.

When a region occluded in some frame and disoccluded later gets rebuilt during the process, most likely this region will be given a different appearance than before the occlusion. We solve this issue by taking long-term motion estimates into account. Secondly, the style transfer tends to create artifacts at the image boundaries. Therefore, we developed a multi-pass algorithm processing the video in alternating temporal directions using both forward and backward flow.

Apart from the optimization-based approach, which yields high-quality results but is very slow to compute, we also present a network-based approach to video style transfer in the spirit of Johnson et al. [12]. We present a network to stylize subsequent frames of a video given the previously stylized frame as additional input, warped according to the optical flow. As with the optimization-based approach, we incorporate a temporal constraint into the loss and train a network which can be applied recursively to obtain a stable stylized video.

We compare both approaches in terms of speed and consistency. We show quantitative results on the Sintel MPI dataset and qualitative results on several movie shots. We were able to successfully eliminate most of the temporal artifacts and can create smooth and coherent stylized videos both using the optimization-based approach and image transformation networks. Additionally, we provide a detailed ablation study showing the importance of different components of both approaches.

Finally, we apply the style transfer to spherical images projected onto a cube map. The goal is to stylize a cube map such that adjacent edges remain consistent to each other. Technically this is related to video style transfer. We show that style transfer to videos and spherical images can be combined to process spherical videos. Although the approach is still a little too slow for real-time processing, it demonstrates the potential of artistic virtual reality applications.

## 2 Related work

**Image style transfer.** In an initial study, Gatys et al. [4] showed that hidden feature maps learned by a deep neural network can be used for texture synthesis by using feature correlations of an image classification network as a descriptor for the texture.

Later, Gatys et al. [5] extended their approach to synthesizing images which have the texture (or, more generally, the style) of one image while showing the content of another image. The style is represented by feature correlations and the content of an image is described by high-level feature maps of the deep neural network.

The approach of Gatys et al. received much attention inside and outside the computer vision community and triggered a whole line of research on deep-learning-based style transfer. Li et al. [13] proposed a different way to represent the style with the neural network to improve visual quality where the content and style image show the same semantic content. Nikulin et al. [19] tried the style transfer algorithm on features from other networks and proposed several variations in the

way the style of the image is represented to achieve different goals like illumination or season transfer. Luan et al. presented an approach for photo-realistic style transfer, where both the style image and the content are photographs [17].

**Fast image style transfer.** The processing time can be reduced by orders of magnitude by training a deep neural network to learn a mapping from an image to its stylized version. Different approaches have been developed: Li et al. [14] used supervised training and adversarial training. Johnson et al. [12] presented an unsupervised training approach where the loss is computed by using neural network features and statistics as introduced by Gatys et al. [5]. Ulyanov et al. [25] proposed an unsupervised approach, too, but used a multi-scale network architecture. Unsupervised approaches allow the use of larger training datasets, like Microsoft COCO, while Li et al. precomputed a small number of stylized images and generated more training samples by data augmentation techniques. Ulyanov et al. [26] later introduced instance normalization, which greatly improves the qualitative results of feed-forward style transfer networks. Common to all approaches is the limitation that each model can only learn to transform the style according to one specific style template. However, recent techniques also allow multi-style networks [29,6].

**Painted animations.** Litwinowicz [16] proposed an approach to create temporally consistent artistic videos by adding artificial brush strokes to the video. To gain consistency, brush strokes were generated for the first frame and then moved along the optical flow field. Different artistic styles were received by modifying various parameters, such as thickness, of these brush strokes, or by using different brush placement methods. Hays et al. [8] built upon this idea and proposed new stylistic parameters for the brush strokes to mimic different artistic styles. O'Donovan et al. [20] introduced an energy minimization problem for this task. The optimization objective included a temporal constraint by penalizing changes in shape and width of the brush strokes compared to the previous frame.

**Video style transfer.** In an earlier conference version of the present paper, we demonstrated for the first time the capability of the optimization-based approach to stylize videos in a consistent way by incorporating additional loss functions into the objective function [22]. In the present paper, we extend this previous work to fast, network-based style transfer and to spherical images and videos.

Recently, Chen et al. [2], too, implemented video style transfer using feed-forward style transfer networks. Different from this work, Chen et al. improved consistency by reusing features from the intermediate layers of the style transfer network from the previous frame: they are warped using optical flow and combined with feature maps extracted from the current frame in non-occluded regions before passed to the decoder part of the network. More recently, Gupta et al. [7] and Huang et al. [9] presented style transfer networks that do not use optical flow at test time. Similar to the present work, temporal inconsistencies are penalized by a consistency loss based on the optical flow field. As input, however, their style transfer networks do not use the last stylized frame warped, but two video frames only (Huang et al.) or one frame in combination with the unwarped previous stylized frame (Gupta et al.).

## 3 Style transfer in still images

In this section, we briefly review the image style transfer approaches introduced by Gatys et al. [5] and Johnson et al. [12], which are used as basis for our video style transfer algorithms.

The goal is to generate a stylized image $\mathbf{x}$ showing the content of an image $\mathbf{p}$ in the style of an image $\mathbf{a}$. Gatys et al. formulated an energy minimization problem consisting of a *content loss* and a *style loss*. The key idea is that features extracted by a convolutional network carry information about the content of the image, while the correlations of these features encode the style.

We denote by $\phi^l(\cdot)$ the function implemented by the part of the convolutional network from input up to layer $l$. We denote the feature maps extracted by the network from the original image $\mathbf{p}$, the style image $\mathbf{a}$, and the stylized image $\mathbf{x}$ by $\mathbf{P}^l = \phi^l(\mathbf{p})$, $\mathbf{S}^l = \phi^l(\mathbf{a})$ and $\mathbf{F}^l = \phi^l(\mathbf{x})$, respectively. The dimensionality of these feature maps we denote by $N_l \times M_l$, where $N_l$ is the number of filters (channels) in the layer, and $M_l$ is the spatial dimensionality of the feature map, that is, the product of its width and height.

The content loss, denoted as $\mathcal{L}_{content}$, is the mean squared error between $\mathbf{P}^l \in \mathbb{R}^{N_l \times M_l}$ and $\mathbf{F}^l \in \mathbb{R}^{N_l \times M_l}$. This loss need not be restricted to only one layer. Let $L_{content}$ be the set of layers to be used for content representation, then we have:

$$\mathcal{L}_{content}(\mathbf{p}, \mathbf{x}) = \sum_{l \in L_{content}} \frac{1}{N_l M_l} \sum_{i,j} \left(F^l_{ij} - P^l_{ij}\right)^2. \quad (1)$$

The style loss is also a mean squared error, but between the correlations of the filter responses expressed by their Gram matrices $A^l \in \mathbb{R}^{N_l \times N_l}$ for the style image $\mathbf{a}$ and $G^l \in \mathbb{R}^{N_l \times N_l}$ for the stylized image $\mathbf{x}$. These are computed as $A^l_{ij} = \sum_{k=1}^{M_l} S^l_{ik} S^l_{jk}$ and $G^l_{ij} = \sum_{k=1}^{M_l} F^l_{ik} F^l_{jk}$.

As above, let $L_{style}$ be the set of layers we use to represent the style, then the style loss is given by:

$$\mathcal{L}_{style}(\mathbf{a},\mathbf{x}) = \sum_{l\in L_{style}} \frac{1}{N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^l - A_{ij}^l\right)^2 \qquad (2)$$

Overall, the loss function is given by

$$\mathcal{L}_{image}(\mathbf{p},\mathbf{a},\mathbf{x}) = \alpha\mathcal{L}_{content}(\mathbf{p},\mathbf{x}) + \beta\mathcal{L}_{style}(\mathbf{a},\mathbf{x}), \quad (3)$$

with weighting factors $\alpha$ and $\beta$ governing the relative importance of the two components.

The stylized image is computed by minimizing this energy with respect to $\mathbf{x}$ using gradient-based optimization. Typically it is initialized with random Gaussian noise. However, the loss function is non-convex, therefore the optimization is prone to falling into local minima. This makes the initialization of the stylized image important, especially when applying the method to frames of a video.

Instead of solving an optimization problem, a much faster method directly learns a style transfer function for one particular style mapping from an input image to its stylized correspondence. Such a function $\Phi_{\mathbf{w}}$ can be expressed by a convolutional neural network with parameters $\mathbf{w}$. The network is trained to minimize the expected loss for an arbitrary image $\mathbf{p}$:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \mathbf{E}_{\mathbf{p}}\left[\mathcal{L}_{image}\Big(\mathbf{p},\mathbf{a},\Phi_{\mathbf{w}}(\mathbf{p})\Big)\right]. \qquad (4)$$

Johnson et al. [12] directly evaluate $\mathcal{L}_{image}$ in each iteration of the training phase and use the loss and gradients from this function to perform a backward pass. Since this function does not compare the output of the network to a ground truth but acts as a perceptual quality measure, this is called a perceptual loss function.
The network architecture chosen by Johnson et al. [12] is a fully convolutional neural network, which uses downsampling by strided convolution to increase the receptive field and upsampling to produce the final output. Furthermore, the network utilizes residual connections which help the network converge faster.

The quality of network-based style transfer can be significantly improved with instance normalization [26] – a technique inspired by batch normalization [11]. Instance normalization performs contrast normalization for each data sample. Details are provided in the Appendix.

## 4 Optimization-based coherent video style transfer

In this section, we explain our video style transfer approach formulated as an energy minimization problem. This includes two extensions that improve long-term consistency and image quality during camera motion.

We use the following notation: $\mathbf{f}^{(i)}$ is the $i^{th}$ frame of the original video, $\mathbf{a}$ is the style image and $\mathbf{x}^{(i)}$ are the stylized frames to be generated. Furthermore, we denote by $\mathbf{x'}^{(i)}$ the initialization of the style optimization algorithm at frame $i$. By $x_j$ we denote the $j^{th}$ component of a vector $\mathbf{x}$.

### 4.1 Short-term consistency by initialization

Independent initialization with Gaussian noise yields two consecutive frames that are stylized very differently, even when the original frames hardly differ. The most basic way to improve temporal consistency is to initialize the optimization for the frame $i+1$ with the already stylized frame $i$. Areas that have not changed between the two frames are then initialized with the desired appearance, while the rest of the image has to be rebuilt through the optimization process.

This simple approach is insufficient for moving parts of the scene, because the initialization does not match. Thus, we take the optical flow into account and initialize the optimization for frame $i+1$ with the previous stylized frame warped by the optical flow: $\mathbf{x'}^{(i+1)} = \omega_i^{i+1}\left(\mathbf{x}^{(i)}\right)$. Here $\omega_i^{i+1}$ denotes the function that warps a given image using the optical flow field that was estimated between images $\mathbf{f}^{(i)}$ and $\mathbf{f}^{(i+1)}$. Only the first frame of the stylized video is initialized randomly.

Every reliable optical flow technique can be used for computing the optical flow. We experimented with two popular algorithms: DeepFlow [27] and EpicFlow [21]. Both are based on Deep Matching [27]: DeepFlow combines it with a variational approach, while EpicFlow relies on edge-preserving sparse-to-dense interpolation.

### 4.2 Temporal consistency loss

To enforce stronger consistency between adjacent frames we additionally introduce an explicit consistency penalty to the objective function. Disoccluded regions as well as motion boundaries are excluded from the penalizer, which allows the optimizer to rebuild those regions. To detect disocclusions, we perform a forward-backward consistency check of the optical flow [24]. Let $\boldsymbol{\omega} = (u, v)$ be the optical flow in forward direction and

$\hat{\boldsymbol{\omega}} = (\hat{u}, \hat{v})$ the flow in backward direction. Denote by $\widetilde{\boldsymbol{\omega}}$ the forward flow warped to the second image:

$$\widetilde{\boldsymbol{\omega}}(x, y) = \boldsymbol{\omega}((x, y) + \hat{\boldsymbol{\omega}}(x, y)). \tag{5}$$

In areas without disocclusion, this warped flow should be approximately the opposite of the backward flow. Therefore we mark as disocclusions those areas where the following inequality holds:

$$|\widetilde{\boldsymbol{\omega}} + \hat{\boldsymbol{\omega}}|^2 > 0.01(|\widetilde{\boldsymbol{\omega}}|^2 + |\hat{\boldsymbol{\omega}}|^2) + 0.5 \tag{6}$$

Motion boundaries are detected using the following inequality:

$$|\nabla \hat{\mathbf{u}}|^2 + |\nabla \hat{\mathbf{v}}|^2 > 0.01|\hat{\mathbf{w}}|^2 + 0.002 \tag{7}$$

Coefficients in inequalities (6) and (7) are taken from Sundaram et al. [24].

The temporal consistency loss function penalizes deviations from the warped image in regions where the optical flow is consistent and estimated with high confidence:

$$\mathcal{L}_{temporal}(\mathbf{x}, \boldsymbol{\omega}, \mathbf{c}) = \frac{1}{D} \sum_{k=1}^{D} c_k \cdot (x_k - \omega_k)^2 \,. \tag{8}$$

Here $\mathbf{c} \in [0, 1]^D$ is a per-pixel weighting of the loss and $D = W \times H \times C$ is the dimensionality of the image. We define the weights $\mathbf{c}^{(i-1,i)}$ between frames $i-1$ and $i$ as follows: 0 in disoccluded regions (as detected by forward-backward consistency) and at the motion boundaries, and 1 elsewhere. The overall loss takes the form:

$$\mathcal{L}_{shortterm}\big(\mathbf{f}^{(i)}, \mathbf{a}, \mathbf{x}^{(i)}\big) = \tag{9}$$
$$\alpha \mathcal{L}_{content}\big(\mathbf{f}^{(i)}, \mathbf{x}^{(i)}\big) \;+\; \beta \mathcal{L}_{style}\big(\mathbf{a}, \mathbf{x}^{(i)}\big) \;+$$
$$\gamma \mathcal{L}_{temporal}\big(\mathbf{x}^{(i)}, \omega_{i-1}^{i}(\mathbf{x}^{(i-1)}), \mathbf{c}^{(i-1,i)}\big) \,. \tag{10}$$

We optimize the frames sequentially, thus $\mathbf{x}^{(i-1)}$ refers to the already stylized frame $i-1$.

We also experimented with the more robust absolute error instead of squared error for the temporal consistency loss; results are shown in section A.4.

### 4.3 Long-term consistency

The short-term model only accounts for consistency between adjacent frames. Therefore, areas that are occluded in some frame and disoccluded later will likely change their appearance in the stylized video. This can be counteracted by taking the long-term motion into account, i.e., not only penalizing deviations from the previous frame, but also from temporally distant frames. Let $J$ denote the set of indices each frame should take into account relative to the frame number, e.g., with $J = \{1, 2, 4\}$, frame $i$ takes frames $i-1$, $i-2$, and $i-4$ into account. The loss function with long-term consistency is given by:

$$\mathcal{L}_{longterm}\big(\mathbf{f}^{(i)}, \mathbf{a}, \mathbf{x}^{(i)}\big) =$$
$$\alpha \mathcal{L}_{content}\big(\mathbf{f}^{(i)}, \mathbf{x}^{(i)}\big) \;+\; \beta \mathcal{L}_{style}\big(\mathbf{a}, \mathbf{x}^{(i)}\big) \;+$$
$$\gamma \sum_{j \in J : i-j \geq 1} \mathcal{L}_{temporal}\big(\mathbf{x}^{(i)}, \omega_{i-j}^{i}(\mathbf{x}^{(i-j)}), \mathbf{c}_{long}^{(i-j,i)}\big) \tag{11}$$

The weights $\mathbf{c}_{long}^{(i-j,i)}$ are adjusted such that each pixel is only connected to the closest possible frame from the past. Since the optical flow computed over more frames is more erroneous than over fewer frames, this results in nicer videos. This is implemented as follows: let $\mathbf{c}^{(i-j,i)}$ be the weights for the flow between image $i-j$ and $i$, as defined for the short-term model. The long-term weights $\mathbf{c}_{long}^{(i-j,i)}$ are then computed as:

$$\mathbf{c}_{long}^{(i-j,i)} = \max\big(\mathbf{c}^{(i-j,i)} - \sum_{k \in J : i-k > i-j} \mathbf{c}^{(i-k,i)},\; \mathbf{0}\big) \,, \tag{12}$$

where the maximum is applied element-wise. This means, we first apply the usual short-term constraint. For pixels in disoccluded regions we look into the past until we find a frame in which these have consistent correspondences. An empirical comparison of $\mathbf{c}^{(i-j,i)}$ and $\mathbf{c}_{long}^{(i-j,i)}$ is shown in the supplementary video (see section A.1).

### 4.4 Multi-pass algorithm

We found that the output image tends to have less contrast and to be less diverse near image boundaries than in other areas of the image. This effect appears particularly in dynamic videos with much camera motion. Disocclusion at image boundaries constantly creates small areas with new content that moves towards other parts of the image. This leads to a lower image quality over time when combined with our temporal constraint. Therefore, we developed a multi-pass algorithm that processes the whole sequence in multiple passes and alternates between the forward and backward direction. Every pass consists of a relatively low number of iterations without full convergence. At the beginning, we process every frame independently. After that, we blend frames with non-disoccluded parts of previous frames warped according to the optical flow, then run the optimization algorithm for some iterations initialized with this blend. We repeat this blending and optimization to convergence.

Let $\mathbf{x'}^{(i,j)}$ be the initialization of frame $i$ in pass $j$ and $\mathbf{x}^{(i,j)}$ the corresponding output after some iterations of the optimization algorithm. When processed in

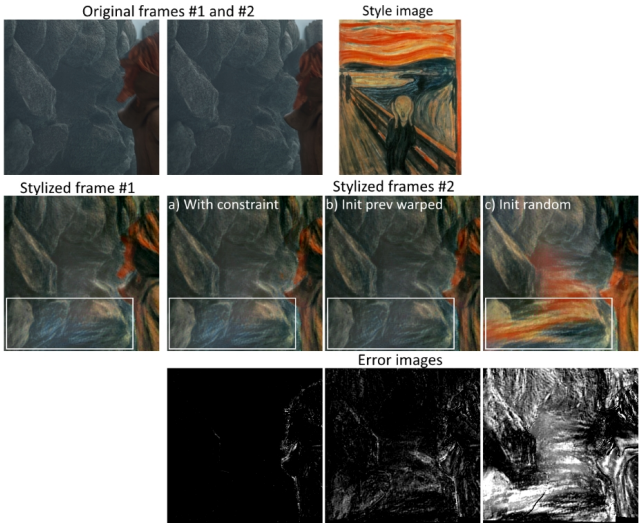


**Fig. 2** Close-up of a scene from Sintel, combined with *The Scream* painting. **a)** With temporal constraint **b)** Initialized with previous image warped but without the constraint **c)** Initialized randomly. The marked regions show most visible differences. *Error images* show the contrast-enhanced absolute difference between frame #1 and frame #2 warped back using ground truth optical flow, as used in our evaluation. The effect of the temporal constraint is clearly visible in the error images.

forward direction, the initialization of frame $i$ is created as follows:

$$\mathbf{x'}^{(i,j)} = \begin{cases} \mathbf{x}^{(i,j-1)} & \text{if } i = 1, \\ \delta\mathbf{c}^{(i-1,i)} \circ \omega_{i-1}^{i}\left(\mathbf{x}^{(i-1,j)}\right) + & \\ (\overline{\delta}\mathbf{1} + \delta\overline{\mathbf{c}}^{(i-1,i)}) \circ \mathbf{x}^{(i,j-1)} & \text{else.} \end{cases} \tag{13}$$

Here $\circ$ denotes element-wise vector multiplication, $\delta$ and $\overline{\delta} = 1 - \delta$ are the blend factors, $\mathbf{1}$ is a vector of all ones, and $\overline{\mathbf{c}} = \mathbf{1} - \mathbf{c}$.

Analogously, the initialization for a backward direction pass is:

$$\mathbf{x'}^{(i,j)} = \begin{cases} \mathbf{x}^{(i,j-1)} & \text{if } i = N_{\text{frames}} \\ \delta\mathbf{c}^{(i+1,i)} \circ \omega_{i+1}^{i}\left(\mathbf{x}^{(i+1,j)}\right) + & \\ (\overline{\delta}\mathbf{1} + \delta\overline{\mathbf{c}}^{(i+1,i)}) \circ \mathbf{x}^{(i,j-1)} & \text{else.} \end{cases} \tag{14}$$

The multi-pass algorithm can be combined with the temporal consistency loss described above. We achieved good results when we disabled the temporal consistency loss in several initial passes and enabled it in later passes after the images had stabilized.

## 5 Fast coherent video style transfer

### 5.1 Consistency by architecture

Before introducing temporal consistency constraints into a network based approach, we investigated whether the temporal consistency of the output of the stylization network can be improved simply by using an architecture that is robust to transformations of the input image. Figure 3 shows that the network of Johnson et al. is invariant to shifts of exactly four pixels, but its output is significantly disturbed by smaller shifts. The loss of shift invariance is caused by the use of downsampling (strided) layers in the network, with a total downsampling factor of 4.

Shift invariance can be retained by avoiding downsampling and using a different technique for multi-scale context aggregation. This can be achieved with dilated convolutions [28], which expand the convolutional filters instead of downsampling the feature maps. The experiments in Section 7.6 show that this indeed leads to improved temporal consistency, but at the cost of a heavily increased computational burden, especially in terms of memory. This makes the approach impractical under most circumstances. Furthermore, this technique does not account for transformations that are more complex than axis-aligned shifts. Thus, in the following we present more efficient and more general techniques to improve temporal consistency in network based stylization.

### 5.2 Training with the prior image

Our main approach builds upon the architecture by Johnson et al. [12] but extends the network to take additional inputs. In addition to the content image, the network gets an incomplete image representing prior knowledge about the output, which we refer to as *prior image*, as well as a per-pixel mask telling the network which regions of the prior image are valid. For video stylization, the prior image is the previously stylized frame warped according to the optical flow we calculated for the original video sequence. The mask is given by the occlusions between two frames and the motion boundaries. Occlusions are detected through a forward-backward consistency check, as described in the previous section, i.e., we calculate the optical flow both in forward and backward direction and test whether both estimates agree. At pixels where the estimates do not agree, and at motion boundaries, we set the certainty mask to zero. Everywhere else, we set this mask to 1.

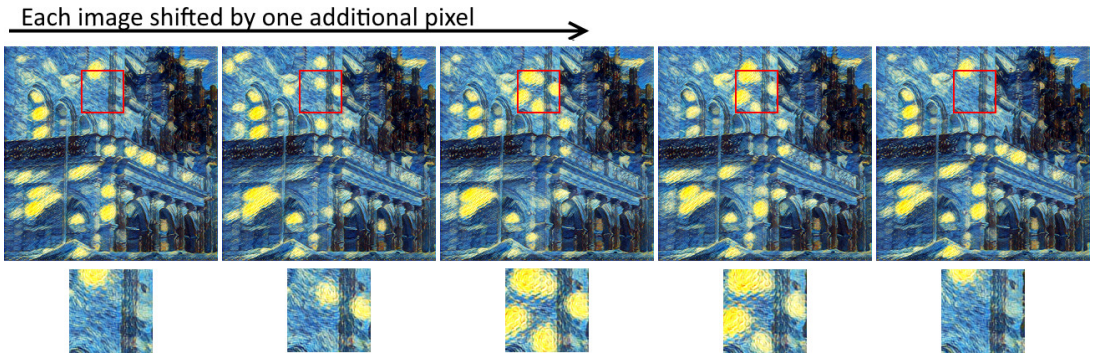


**Fig. 3** Images generated by feed-forward style transfer. Starting from the left, each image is shifted by one additional pixel in x and y direction. The first and the fifth image, where the shift accumulates to a multiple of four pixels, are stylized nearly identically. The other images are stylized very differently from the first image. This shows that there is no shift invariance unless the shift exactly matches the stride used within the network.

The goal of the network is to produce a new stylized image, where the combination of perceptual loss and deviation from the prior image (in non-occluded regions) is minimized.
The network $\Phi^{vid}_{\mathbf{w}}$ with parameters $\mathbf{w}$ takes as input a frame $\mathbf{f}^{(t)}$, the previously generated frame warped and masked by the occlusion mask $\omega^t_{t-1}(\mathbf{x}^{(t-1)})$ and yields the output $\mathbf{x}^{(t)}$. We can apply this function recursively to obtain a video. The first frame in the sequence is generated by an image network.

$$\mathbf{x}^{(t)} = \begin{cases} \Phi^{img}\left(\mathbf{f}^{(t)}\right) & \text{if } t = 1 \\ \Phi^{vid}\left(\mathbf{f}^{(t)},\ \omega^t_{t-1}(\mathbf{x}^{(t-1)}),\ \mathbf{c}^{(t-1,t)}\right) & \text{if } t > 1 \end{cases} \tag{15}$$

Ideally, network parameters $\mathbf{w}$ would minimize the video loss function, as defined in Section 4.2, for an arbitrary video with $N_{frames}$ frames:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \mathbf{E}_{\{\mathbf{f}^{(t)}\}}\left[\sum_{t=2}^{N_{\text{frames}}} \mathcal{L}_{video}\left(\mathbf{f}^{(t)},\ \mathbf{a},\ \mathbf{x}^{(t)},\ \mathbf{x}^{(t-1)}\right)\right], \quad \text{with } \mathbf{x}^{(t>1)} = \Phi^{vid}_{\mathbf{w}}\left(\mathbf{f}^{(t)}, \omega^t_{t-1}(\mathbf{x}^{(t-1)}), \mathbf{c}^{(t-1,t)}\right), \tag{16}$$

where $\mathbf{x}^{(1)} = \Phi^{img}_{\mathbf{w}'}(\mathbf{f}^{(1)})$ for a given pre-trained parameter set $\mathbf{w}'$.

However, joint optimization for a whole video is computationally infeasible. Hence, we resort to recursively optimizing two-frame losses. Our basic approach is illustrated in Figure 4. The network is trained to generate a successor frame to the first frame of a video:

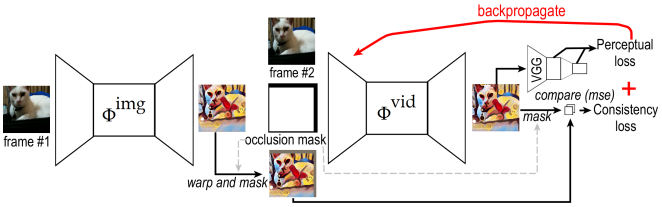


**Fig. 4** Basic training procedure for a style transfer network with prior image (e.g. last frame warped).

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \mathbf{E}_{\mathbf{f}^{(1)},\mathbf{f}^{(2)}}\left[\mathcal{L}_{video}\left(\mathbf{f}^{(2)},\ \mathbf{a},\ \mathbf{x}^{(2)},\ \mathbf{x}^{(1)}\right)\right], \quad \text{with } \mathbf{x}^{(2)} = \Phi^{vid}_{\mathbf{w}}\left(\mathbf{f}^{(2)}, \omega^2_1(\mathbf{x}^{(1)}), \mathbf{c}^{(1,2)}\right). \tag{17}$$

We found this approach to yield unsatisfactory results when applied recursively to stylize several successive frames of a video. While the network learns to fix occlusions and to preserve the prior image as much as possible, there is nothing that prevents the propagation of errors and the degeneration of the image quality over time. In particular, we observed blurring and the loss of content accuracy in the image. In the following we introduce two improvements to avoid the degeneration of image quality.

### *5.2.1 Mixed training*

Our first approach to counteract the propagation of errors aims at forcing the network to refine the stylization at every step, rather than just copying the prior image. To this end, we randomly add training samples without prior image, i.e., the network has to generate a completely new image. We set the occlusion mask to zero everywhere, which also disables the temporal loss. Formally, the objective function changes to

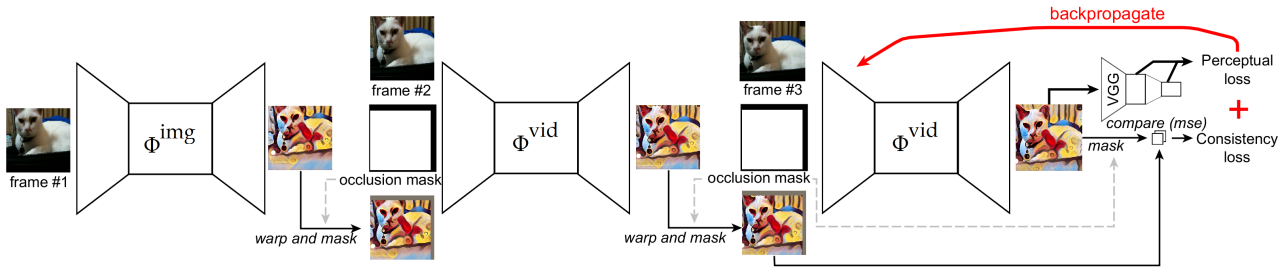


**Fig. 5** Training procedure for our multi-frame approach, here shown with three frames. We found that back-propagating only one frame already improves the quality of generated videos a lot. Back-propagating more frames would have required decreasing the size of the network due to memory restrictions.

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \mathbf{E}_{\mathbf{f}^{(1)},\mathbf{f}^{(2)}} \left[ \mathcal{L}_{video}\Big(\mathbf{f}^{(2)},\ \mathbf{a},\ \mathbf{x}^{(2)},\ \mathbf{x}^{(1)}\Big) \right] + \mathbf{E}_{\mathbf{p}} \left[ \mathcal{L}_{video}\Big(\mathbf{p},\ \mathbf{a},\ \Phi^{vid}_{\mathbf{w}}\Big(\mathbf{p},\mathbf{0},\mathbf{0}\Big),\ \mathbf{0}\Big) \right] \quad \text{with } \mathbf{x}^{(2)} = \Phi^{vid}_{\mathbf{w}}\Big(\mathbf{f}^{(2)}, \omega_1^2(\mathbf{x}^{(1)}), \mathbf{c}^{(1,2)}\Big) \tag{18}$$

where $\mathbf{0}$ is a vector of the same dimension as the input image with zeros everywhere.

We increase the weight for the consistency loss proportionally to the percentage of empty-prior images. This aims for still achieving temporal consistency at test time, while focusing on content generation during training. We experimented with various fractions of zero-prior samples, and got good results when we set this fraction to 0.5.

### *5.2.2 Multi-frame training*

Our second approach to improve longer-term stylizations presents more than just the first two frames of a sequence during training. We rather recursively pass $\mathbf{f}^{(1)}$ to $\mathbf{f}^{(n-1)}$ through the network, i.e., training includes the potentially degenerated and blurred output, $\mathbf{x}^{(n-1)}$, as the prior image and requires the network to generate the non-degenerated ground truth $\mathbf{x}^{(n)}$. This forces the network to learn not only to copy the prior image but also to correct degeneration. If the network did not correct errors and blurriness in the prior image, this would lead to a large loss. Thus, the overall loss function eventually overrules the temporal consistency constraint.

Formally, we generalize the objective function shown in Equation (17) to

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \mathbf{E}_{\mathbf{f}^{(t)},\mathbf{f}^{(t+1)}} \left[ \mathcal{L}_{video}\Big(\mathbf{f}^{(t+1)},\ \mathbf{a},\ \mathbf{x}^{(t+1)},\ \mathbf{x}^{(t)}\Big) \right], \quad \text{with } \mathbf{x}^{(t+1)} = \Phi^{vid}_{\mathbf{w}}\Big(\mathbf{f}^{(t+1)}, \omega_t^{t+1}(\mathbf{x}^{(t)}), \mathbf{c}^{(t,t+1)}\Big). \tag{19}$$

Since the network does not produce meaningful stylizations in the beginning of the training phase, we first start training the network with $t = 1$ and gradually increase $t$ in later iterations.

### *5.2.3 Training data*

We used the Hollywood2 video scene dataset [18], a collection of 570 training videos. Some videos contain multiple shots. We split videos along shots and extracted 4,193 distinct shots from that dataset. Since the dataset contains many frame pairs with little motion, we only took the five most differing frame tuples per shot. In more detail, we computed the squared distance between subsequent frames per tuple and took those five tuples which ranked highest on that measure. In total, we acquired 20,965 frame tuples. The size of a tuple corresponds to the five steps in our multi-frame training approach.

Additionally, we created a large synthetic set of video sequences based on the Microsoft COCO dataset [15] with 80,000 training images to increase the diversity in our training data. To create each of these sequences, we selected a window from the image and moved it between 0 to 32 pixels in either direction to create consecutive frames. Moreover, we used a zoom out effect, i.e., the window size was increased by up to 32 pixels in either direction when generating a following frame, with all crops being resized to the dimensions of the original window size. Translation strength and resize factor were sampled uniformly and independently for each axis for each training sample at training time. The advantage of the synthetic data is that it comes with ground truth optical flow and occlusion maps.

## 6 Spherical images and videos

Virtual reality (VR) applications become increasingly popular, and the demand for image processing methods applicable to spherical images and videos rises. Spherical reality media is typically distributed via a 2D projection. The most common format is the equirectangular projection. However, this format does not preserve shapes: the distortion of the projection becomes very large towards the poles of the sphere. Such non-uniform distortions are problematic for style transfer. Therefore, we work with subdivided spherical images that consist of multiple rectilinear projections. In particular, we will use cubic projection, which represents a spherical image with six non-distorted square images, as illustrated in Figure 6. For style transfer in this regime, the six cube faces must be stylized such that their cut edges are consistent, i.e., the style transfer must not introduce false discontinuities along the edges of the cube in the final projection. Since applications in VR environments must run in real time, we only consider the fast, network-based approach in this section.

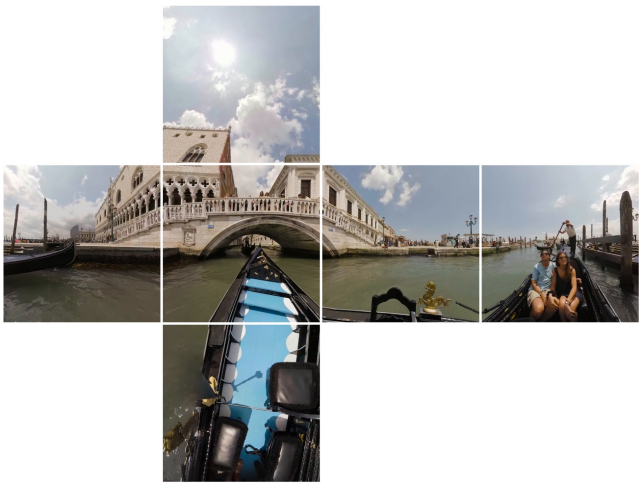

**Fig. 6** Cubemap projection used for stylizing spherical images in this paper. The generated images must be consistent along the boundaries of neighboring cube faces. Every cube face has four neighbors.

### 6.1 Border consistency

In order to implement the consistency constraint, we extend the cube faces such that they overlap each other in the proximity of the face boundaries; see Figure 7 (top).

We stylize the cube faces sequentially and enforce subsequent cube faces to be consistent with already stylized edges from adjacent cube faces. This is similar to the video style transfer network, and we can use the same techniques described in detail in the previous section, such as mixed training and multi-frame training. Instead of using a previously generated frame from a video as the prior image, we use regions of adjacent cube faces overlapping with the current cube faces as prior image. The regions are transformed to match the cube face projection; see Figure 7 (bottom) and the Appendix.

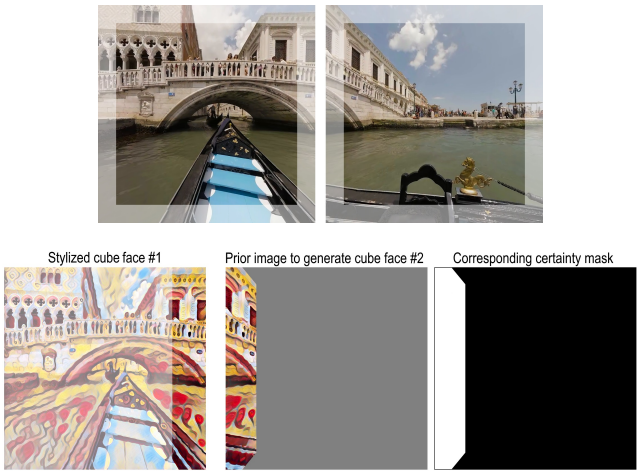


**Fig. 7 Top:** Two extended cube faces of a spherical video with overlapping border regions, **bottom:** A stylized cube-face and the prior image generated from that.

### 6.2 Training dataset

A style transfer network trained with regular video frames did not produce good results for stylizing cube faces, where a perspective transformed border of the image is given as the prior image. Therefore, we extended our dataset by training samples particularly designed for this task.

We stylized and transformed the image borders of samples from the Microsoft COCO dataset in such a way that it mimics the perspective transformed borders of a cube face prior image. This process is illustrated in Figure 8. We created a training set with $320,000$ samples (four samples for each image in COCO) this way.

### 6.3 Spherical videos

The extensions for video style transfer and for spherical images can be combined to process spherical videos. This yields two constraints: (1) each cube face should be

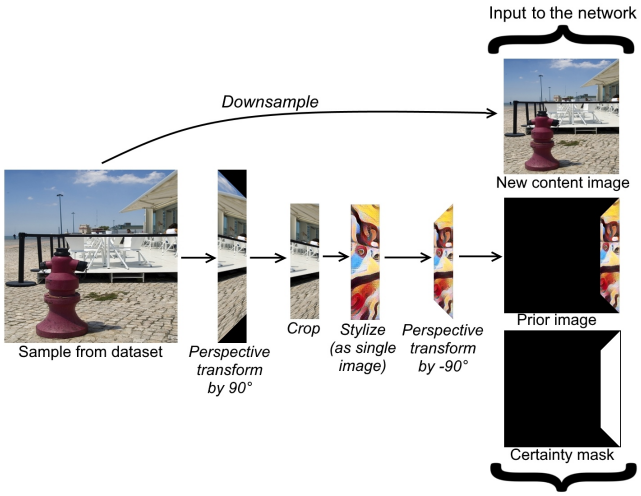


**Fig. 8** Illustration of the training data generation process for a network to adapt to perspectively transformed border regions.

consistent along the motion trajectory; (2) neighboring cube faces must have consistent boundaries.

To implement the first constraint, we calculate optical flow for each cube face separately, which allows us to warp stylized cube faces the same way as for regular planar videos. To implement the second constraint, we blend both the warped image from the last frame and the transformed border of already stylized neighboring cube faces.

## 7 Experiments on video style transfer

In this section, we present the experimental results obtained with different versions of the presented methodology. Since it is hard to visualize qualitative differences in video content and spherical images appropriately on paper, we additionally refer to the supplemental video linked in Section A.1.

### 7.1 Dataset

We used the MPI Sintel Dataset [1] including 23 training scenes with 20 to 50 frames of resolution $1024 \times 436$ pixels per scene. To keep the runtime at an acceptable level, we tested our optimization-based approach with a subset of five diverse scenes. Our feed-forward approach, being orders of magnitudes faster, was additionally evaluated on the full set.

The Sintel dataset provides ground truth optical flow and ground truth occlusion areas, which allows a quantitative study. To evaluate the short-term consistency, we warped each stylized frame at time $t$ back with the ground truth flow and computed the mean squared difference to the stylized frame $t-1$ in regions without disocclusion. Each method was evaluated with different style templates, as shown in Section A.2.

### 7.2 Implementation details

#### *7.2.1 Optimization-based approach*

Our implementation[1] is based on a Torch [3] implementation called *neural-style*[2]. We used the following layers of the VGG-19 network [23] for computing the losses: *relu4_2* for the content and *relu1_1*, *relu2_1*, *relu3_1*, *relu4_1*, *relu5_1* for the style. The energy function was minimized using L-BFGS. For precise evaluation we incorporated the following strict stopping criterion: the optimization was considered converged if the loss did not change by more than 0.01% during 50 iterations. This typically resulted in roughly 2000 to 3000 iterations for the first frame and roughly 400 to 800 iterations for subsequent frames when optimizing with our temporal constraint, depending on the amount of motion and the complexity of the style image. Using a convergence threshold of 0.1% cuts the number of iterations and the running time in half, and we found it still produces reasonable results in most cases. However, we used the stronger criterion in our experiments for the sake of accuracy.

For videos of resolution $350 \times 450$ we used weights $\alpha = 1$ and $\beta = 20$ for the content and style losses, respectively (default values from *neural-style*), and weight $\gamma = 200$ for the temporal losses. However, the weights should be adjusted if the video resolution is different. We provide the details in section A.2.

We ran our multi-pass algorithm with 100 iterations per pass and set $\delta = 0.5$. Since at least 10 passes are needed for good results, this algorithm needs in total more computation time than the single-pass approaches.

Optical flow was computed with the DeepMatching, DeepFlow, and EpicFlow implementations provided by the authors of these methods. We used the "improved-settings" flag in DeepMatching 1.0.1 and the default settings for DeepFlow 1.0.1 and EpicFlow 1.00.

#### *7.2.2 Network-based approach*

Our network-based style transfer implementation[3] is based on Johnson et al. [12]'s Torch implementation,

[1] `https://github.com/manuelruder/artistic-videos`
[2] `https://github.com/jcjohnson/neural-style`
[3] `https://github.com/manuelruder/fast-artistic-videos`

**Table 1** The number of frames and the training schedule for the multi-frame model.

| Iteration | Number of frames |
|---|---|
| 0 – 60,000 | 2 |
| 60,001 – 70,000 | 3 |
| 70,001 – 120,000 | 5 |

called *fast-neural-style*[4]. The architecture follows the design principle of Johnson et al., and we used the following layers of the VGG-16 network for the perceptual loss: *relu3_3* for the content and *relu1_2*, *relu2_2*, *relu1_2*, *relu3_3*, *relu4_3* for the style target. In the proposed style transfer network, we exchanged the upsampling convolution by two nearest neighbor upsampling layers with a regular convolution layer in between. This greatly reduces the checkerboard patterns visible in the original work. In addition, we replaced the batch normalization by instance normalization, as proposed in [26], and used a single padding layer in the beginning. The network was trained with the ADAM optimizer, a learning rate of 0.001, a batch size of 4 for the strided convolution architecture, and a learning rate of 0.0005 with a batch size of 2 for the dilated convolution variant due to memory restrictions. We trained the two-frame model for 60,000 iterations. The multi-frame model was fine-tuned from a pre-trained two-frame model. The number of iterations and frames are shown in Table 1. For spherical image fine-tuning, we started from the trained multi-frame model and trained for another 40,000 iterations using the specialized dataset described in section 6.2. To calculate optical flow for the videos from the Hollywood2 dataset, we used FlowNet 2.0 [10], which is a recent state-of-the-art for optical flow estimation running at interactive frame rates. To overcome GPU memory limitations when training with dilated convolutions, we downsampled the frames by a factor of 0.65. We set the weight of the temporal consistency to 100 when using mixed training and to 50 otherwise.

### 7.3 Comparison to baselines

We benchmarked the following video stylization approaches: the optimization-based approach with per-frame random initialization, with initialization from the previous frame, and with our short-term consistency loss, where optical flow was provided by DeepFlow. Moreover, we ran the frame-wise network-based approach and our multi-frame network.

Quantitative results are shown in Table 2, averaged for 6 style templates also used by Gatys et al. shown in Section A.2.1. Among optimization-based approaches, the most straightforward method – processing every frame independently – performs roughly an order of magnitude worse than the version with temporal consistency. Initialization with the previous stylized frame leads to better results than random initialization, but is still 2.3 to 16 times worse than our approach, depending on the scene.

The baseline for our network-based approach (independent per-frame) shows less flickering than the previous baseline, since this approach is more deterministic by design. Nevertheless, we consistently outperform this baseline by a factor of two or more. Interestingly, the network-based approach even outperforms the optimization-based approach in some cases while being much faster. The network-based method is more accurate on the *ambush* scenes. These include very large motion, which leads to more errors in the optical flow. We conclude that the network-based method is able to partially correct for these errors in optical flow.

### 7.4 Results of a user study

In the previous section we have only evaluated the temporal consistency, but not the perceptual quality of the stylized videos. We performed a comprehensive user study using Amazon Mechanical Turk (AMT) to evaluate the perceptual quality of videos stylized with our two best approaches: optimization-based with DeepFlow and network-based with multi-frame and mixed training (for details see Sections 7.5 and 7.6). We performed 20 trials for each combination out of the five tested scenes and the five tested styles, resulting in a total of 500 trials. In each trial a user was presented the style template, a still frame from the original video and two stylized versions of the video clip, one video processed with the optimization-based approach and the other processed with a network. The order of the videos was randomized. Inspired by [7], users were asked three questions: “Which of the two videos better reflects the style of the painting?”, “Which of the two videos has less flickering?” and “Overall, which of the two videos do you think looks better?”. Users had to select either one of the videos or “undecided”.

Table 3 shows the results of the user study, separately for each style image. There was a quite clear preference towards video clips generated with the network approach, even though this approach is just an approximation of the much costlier optimization problem. This could be explained by the fact that our network approach is more robust to errors in the optical flow and can fill disocclusions more consistently due to less randomness. This reduces artifacts compared to the optimization-based approach.

[4] `https://github.com/jcjohnson/fast-neural-style`

| Type | Method | alley_2 | ambush_5 | ambush_6 | bandage_2 | market_6 | Runtime |
|---|---|---|---|---|---|---|---|
| Optimization | Random init | 0.019 | 0.027 | 0.037 | 0.0180 | 0.023 | 540 s |
| Optimization | Prev frame init | 0.010 | 0.018 | 0.028 | 0.0041 | 0.014 | 260 s |
| Optimization | Ours | **0.00061** | 0.0062 | 0.012 | **0.00084** | **0.0035** | 180 s |
| Network | Per-frame | 0.0062 | 0.011 | 0.016 | 0.0043 | 0.0089 | 0.2 s |
| Network | Ours | 0.0016 | **0.0042** | **0.0079** | 0.0015 | 0.0039 | 0.4 s |

**Table 2** Short-term temporal consistency of video style transfer. We report average mean squared error over the test set (lower is better). Pixel values were between 0 and 1. Run time is reported in seconds per frame. We show the results of our optimization-based and network-based approaches, as well as three baselines: optimization-based stylization initialized with random noise or the previous frame, respectively, and independent per-frame network-based processing.

| Subject | Nude | Picasso | Scream | Shipwreck | Woman Hat | Sum |
|---|---|---|---|---|---|---|
| Better reflects the style | 44/**55**/0 | 28/**72**/0 | 34/**61**/1 | 39/**60**/1 | 46/**53**/1 | 191/**301**/3 |
| Less flickering | 34/**53**/12 | 23/**68**/9 | 26/**59**/11 | 31/**63**/6 | 31/**62**/7 | 145/**306**/45 |
| Overall preference | 30/**68**/1 | 12/**86**/2 | 15/**80**/1 | 21/**76**/3 | 31/**66**/3 | 109/**376**/10 |

**Table 3** Results of the Amazon Mechanical Turk user study. The first number shows how many users picked the optimization-based version, the second one how many picked feed-forward and the last number represent the number of users who picked "undecided".

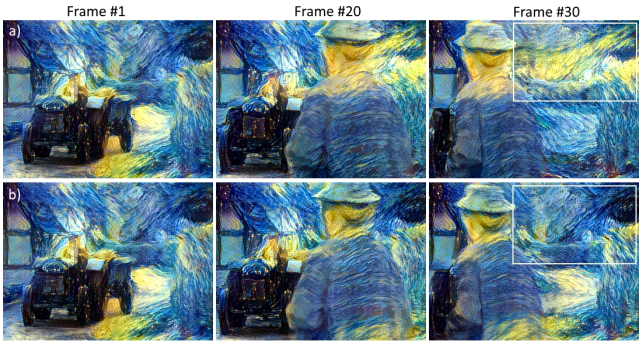


**Fig. 9** Scene from Miss Marple, combined with The Starry Night painting. **a)** Short-term consistency only. **b)** Long-term consistency with $J = \{1, 10, 20, 40\}$. The corresponding video is available at `https://youtu.be/SKql5wkWz8E#t=1m40s`.

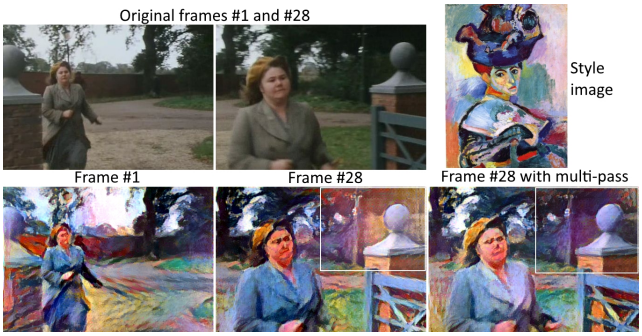


**Fig. 10** The multi-pass algorithm applied to a scene from Miss Marple. With the default method, the image becomes notably brighter and loses contrast, while the multi-pass algorithm better preserves the image quality over time. The corresponding video can be found at `https://youtu.be/SKql5wkWz8E#t=2m17s`.

### 7.5 Detailed analysis of the optimization-based approach

We performed an ablation study of the variants of our optimization-based approach to show the contribution of the different components and the choice of the optical flow method. The results are provided in Table 4. Initialization with the motion-compensated previous frame performs already much better than initializing just with the previous frame. The explicit temporal penalty additionally improves over this result. Comparing the two optical flow techniques, on average DeepFlow yields slightly better results than EpicFlow.

To show the benefit of the proposed long-term consistency and multi-pass technique, we cannot run a quantitative evaluation, since there is no long-term ground truth optical flow available. Thus, we present qualitative results in Fig. 9 and Fig. 10, respectively. Please see also the supplementary video.

Fig. 9 shows a scene with a person walking through the scene. Without our long-term consistency model, the background looks very different after the person passed by. The long-term consistency model keeps the background unchanged. Fig. 10 shows another scene with fast camera motion. The multi-pass algorithm avoids the artifacts introduced in new image areas over time by the basic algorithm.

### 7.6 Detailed analysis of the network-based approach

Table 5 shows the result of an ablation study for different variants of our network-based approach. We measure temporal, style and content loss. The latter two

| initialization | penalizer | opt. flow | alley_2 | ambush_5 | ambush_6 | bandage_2 | market_6 |
|---|---|---|---|---|---|---|---|
| prev | no | n/a | 0.010 | 0.018 | 0.028 | 0.0041 | 0.014 |
| prev warped | no | DeepFlow | 0.0016 | 0.0063 | **0.012** | 0.0015 | 0.0049 |
| prev warped | yes | DeepFlow | **0.00061** | **0.0062** | **0.012** | 0.00084 | 0.0035 |
| prev warped | yes | EpicFlow | 0.00073 | 0.0068 | 0.014 | **0.00080** | **0.0032** |

**Table 4** Ablation study for the optimization-based approach. We report average mean squared error over the test set (lower is better). Pixel values in images were scaled between 0 and 1.

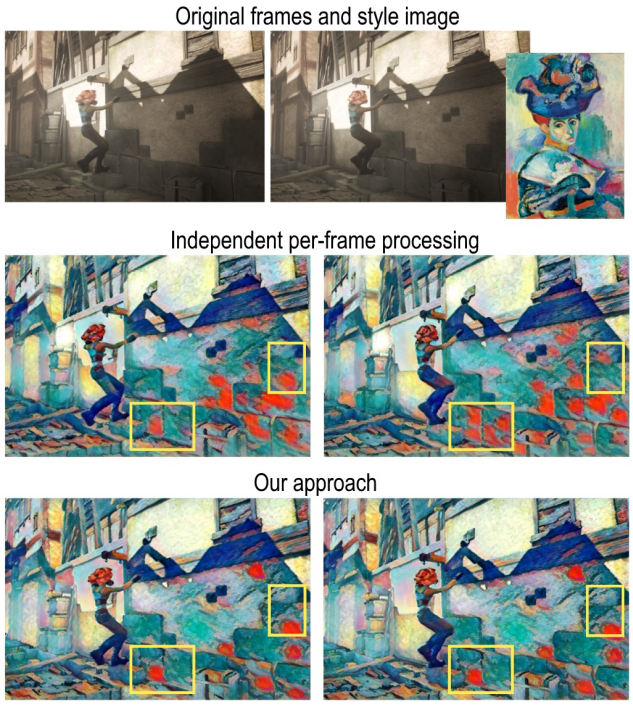


**Fig. 11** Comparison of temporal consistency: Our method (multi-frame mixed) has less flickering than independent per-frame processing [12]. Rectangles indicate relevant differences. The corresponding video is available at `https://youtu.be/SKql5wkWz8E#t=0m42s`

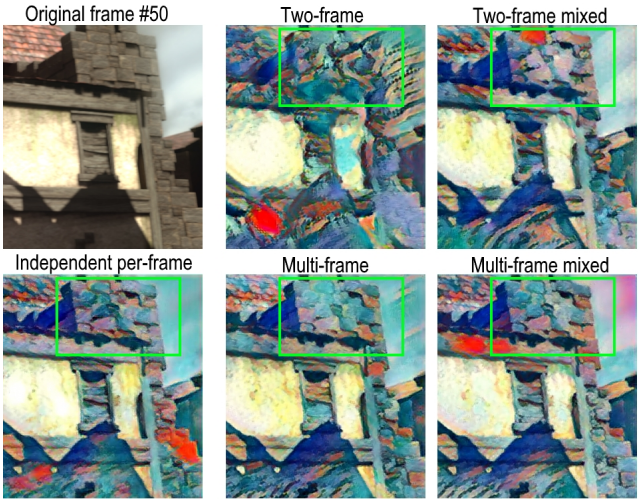


**Fig. 12** Comparison of quality: Our advanced approaches retain visual quality and produce a result similar to [12] applied per-frame. The straightforward two-frame training, though, suffers from degeneration of quality. Rectangles indicate relevant differences. The corresponding video is available at `https://youtu.be/SKql5wkWz8E#t=2m47s`

losses aim to measure propagation of error. If the image starts to blur or degenerate over time, this should increase the style and content loss. The numbers show results on the complete Sintel training dataset averaged over all scenes, frames and styles. We used a different set of style templates; inspired by [12], see Section A.2.2. Since the absolute losses vary heavily, we rather averaged the loss as percentage of the loss of the baseline, which is the independent per-frame processing approach.

Dilated convolution, our proposal to achieve consistency without using prior images, yields a noticeable improvement in temporal consistency while keeping the style and content losses constant, see Table 5. To overcome GPU memory limitations, the frames were downsampled by a factor 0.65. To get a fair comparison, we also downsampled the frames for strided convolution and show results relative to this downsampled baseline.

**Table 5** Comparison of network-based approaches. Values are shown as a percentage of the baseline approach *BL* (independent per-frame processing). The following methods are evaluated: Per-frame processing with dilated convolutions (Di), two-frame training (2F), multi-frame training (MF) and mixed training as described in section 5.2.1 (2Fm, MFm).

| *Loss* | BL | Di | 2F | MF | 2Fm | MFm |
|---|---|---|---|---|---|---|
| Content | 100% | **97%** | 118% | 101% | 116% | **97%** |
| Style | 100% | 97% | 167% | 85% | 98% | **77%** |
| Temporal | 100% | 67% | **41%** | 43% | 48% | 49% |

The two-frame training procedure yields very good temporal consistency, but at the same time leads to a large increase in the content and the style loss. The best results are achieved with the mixed and multi-frame training. Both techniques are complementary and bring down the content and style loss compared to the two-frame training, while only marginally reducing temporal consistency.

Figures 12 shows a qualitative comparison, another comparison can be seen in the appendix (Figure A.5) showing different style templates. The difference between the multi-frame and the mixed training is very subtle, while the two-frame, non-mixed training produces clearly inferior stylizations. An example on how our approach minimizes inconsistencies is shown in Figure 11.

## 8 Experiments on spherical images and videos

To evaluate spherical images and videos, we downloaded 20 video scenes from YouTube containing different content (seven cityscape, six landscape and seven building interior scenes). For the spherical still image evaluation, we took three frames from each scene at a distance of approximately three seconds, i.e. 60 images.

### 8.1 Spherical images

To evaluate how well the individual cube faces fit together, i.e., if there are false discontinuities after stitching the cube faces, we measured the gradient magnitude at the cut edges (two pixel width). We report the gradient magnitude relative to the overall gradient magnitude in the image. A value of 1 means there is no unusual gradient in that region. Values larger than 1 indicate more gradients compared to the average gradient of the image.

Subsequent cube faces generated with a neighboring cube face border as prior image sometimes showed false discontinuities and/or unusual high gradients at the inner edges of the prior image, where the overlap between the images end. Therefore, we also measured gradients along the inner edges of the prior image in the stylized cube faces (four pixel width). Our detailed evaluation metric is shown in the appendix.

Results are shown in Table 6. Introducing edge consistency clearly reduces the false gradients along cut edges. However, it introduces new gradients along the inner edge from the prior image, most noticably in regions with little structure. Thus, on average, the magnitude of artificial gradients does not improve noticeably. Only when we use our specialized dataset for spherical images to fine-tune the model, as described in Section 6.2, the gradient error improved overall in the image. Fig. 13 shows a qualitative comparison. In textured regions, fine-tuning is not necessary. In homogeneous regions, fine-tuning helps to avoid artificial edges. Interestingly, fine-tuning introduces style features to cover the false edges, i.e., the gradient magnitude in our quantitative measure is still increased, but the disturbing visual effect is reduced and only noticeable when large parts of the image are untextured like, for example, the top cube face showing the sky.

**Table 6** Style transfer to spherical images. We report the gradient magnitudes at boundaries, the content and the style loss. The first column shows the baseline result when each cube face was stylized independently. "Video net" refers to a network only trained on video frames. For "Video net, ft" we fine-tuned the latter on data samples specifically made for spherical images.

| *Loss* | Ind. per face | Video net | Video net, ft |
|---|---|---|---|
| $E_{\text{grad}}$ | 1.46 | 1.41 | **1.21** |
| $E_{\text{grad}_{cut}}$ | 2.48 | 1.40 | **1.17** |
| $E_{\text{grad}_{inner}}$ | **0.95** | 1.41 | 1.23 |
| Content | 100 % | 101 % | 98 % |
| Style | 100 % | 104 % | 98 % |

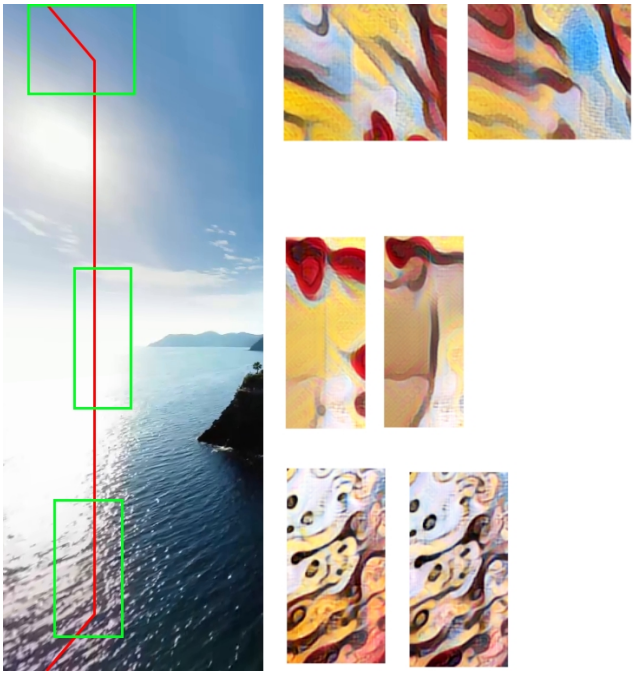

**Fig. 13** The left image shows the overlap region of a cube face from a panoramic image. The right shows close-ups for two networks. **Left:** Not fine-tuned. **Right:** Fine-tuned. In regions with little structure (top and middle), the fine-tuning strategy reduced unnatural artifacts along the inner edge of the prior image. It sometimes uses stylistic features to mask the transition (middle). In regions with more structure (bottom), both networks adapted well to the given prior.

### 8.2 Spherical videos

Table 7 shows numerical results for stylized panoramic videos at different frames of the video. We can see that

unusual high gradients decrease over time, demonstrating the algorithm's ability to improve from impaired prior images. We also observe this effect when the video is mostly still and only few objects are moving.

**Table 7** Evaluation results ($E_{\text{grad}}$ as defined above) for spherical videos. We use the multi-frame model, fine-tuned for spherical images and measure the error at different time steps.

| *Loss* | Frame #1 | Frame #2 | Frame #10 |
|---|---|---|---|
| $E_{\text{grad}}$ | 1.21 | 1.12 | 0.98 |

### 8.3 Filling missing regions

To further reduce unusual gradients, we experimented with different ways to fill in missing or disoccluded regions in the prior image. Originally, we masked missing regions with zeros, which is equivalent to the mean pixel of the ImageNet dataset[5]. By filling missing regions with random noise instead, we want to avoid that the network falsely observes sharp edges along missing regions.

Numerical analysis shows that the $E_{gradient}$ measure indeed improves slightly from 1.21 to 1.18 when filling masked regions with random noise.

Most noticeably, a random fill-in improves a failure case of our algorithm: regions with little structure in the image, see Figure 14. False gradients along the prior image boundary are less visible.

However, we observed that a random fill-in further changes the way the network fills missing regions. To get a better understanding of how the fill-in influences the network's behavior, we performed an experiment where we used random noise as fill-in during training, but used zeros at test time. Figure 15 shows how this changes the output. For this experiment, we gave the network an empty prior image, as if the whole image was missing or disoccluded, i.e. the prior image is either a vector containing only zeros or random noise. Once we removed this extra source of entropy, the network was able to rely on it during training, and the image loses variance especially in regions with little structure. We conclude that the network has learned to make use of the noise such that it creates a more diverse appearance in regions with little structure.

[5] In the pretrained VGG models, which we use for our perceptual loss, values are centered around the mean pixel of the ImageNet dataset.

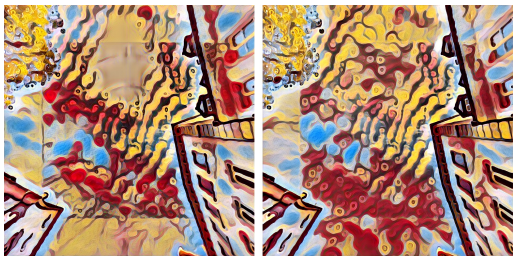

**Fig. 14** Comparison of the top cube face with different ways to fill masked regions. Left: Zeros, right: Random noise.

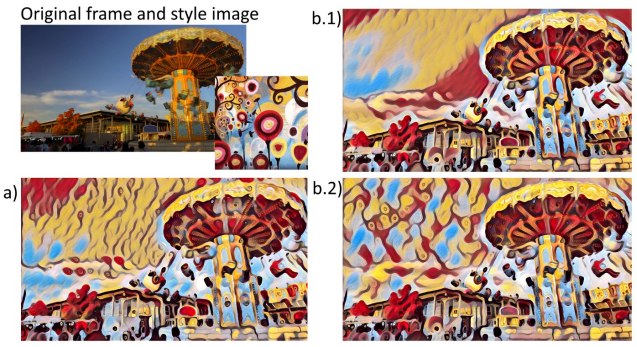


**Fig. 15** On the effect of different ways to fill disoccluded regions. a) Trained and inferred with zeros as fill-in. b.1) Trained with random noise, but inferred with zeros. b.2) Trained and inferred with random noise. Images were generated with no prior image given, but either a zero-vector or random noise, as if the whole image was disoccluded.

## 9 Runtimes

We measured the runtime of our algorithms on an Nvidia Titan X GPU (Maxwell architecture) and a resolution of $1024 \times 436$ pixel.

Using the relaxed convergence threshold of 0.1%, the optimization-based approach takes around eight to ten minutes for the initial frame initialized with random noise. When initialized with the warped previous frame and combined with our temporal loss, the optimization converges 2 to 3 times faster, three minutes on average.

Our feed-forward approach only takes 400 ms to stylize a new frame, including the time needed to preprocess the last frame (masking and warping, which is done on the GPU as well), but excluding time for I/O operations (e.g. writing the file to disk). This corresponds to a speedup factor of 450 compared to the optimization-based approach. A single-image processed with Johnson et al. [12] needed $200ms$ at that resolution. Training the multi-frame model takes around 24 hours for 120,000 iterations.

For panoramic images, our approach needs 3.9 seconds to stylize a full frame ($650ms$ for each cube face with resolution $768 \times 768$ including overlapping regions. Without overlapping, this equals a resolution of $640 \times 640$ per cube face). If the video is not distributed

in the cube face projection, additional time for reprojection has to be taken into account.

## 10 Conclusion

We presented a set of techniques for optimization-based style transfer in videos: suitable initialization and a loss function that enforces short-term temporal consistency of the stylized video, a loss function for long-term consistency, and a multi-pass approach. As a consequence, we can produce stable and visually appealing stylized videos even in the presence of fast motion and strong occlusion.

Furthermore, we presented an algorithm to train a neural network with a prior image in order to create coherent style transfer with a much lower run time per frame. We were able to prevent the propagation of errors when applying our model recursively: by iterating the network multiple times during training, using a longer sequence of frames, we trained our network to recognize and counteract progressed degeneration. We also generalized our model for spherical images: in this task, the prior image is not the last frame, but an edge from a neighboring cube face, perspectively transformed by 90°. We could show that our model is able to adapt to such a prior image when there is enough structure in the image.

## References


1. Butler, D.J., Wulff, J., Stanley, G.B., Black, M.J.: A naturalistic open source movie for optical flow evaluation. In: ECCV, pp. 611–625 (2012)
2. Chen, D., Liao, J., Yuan, L., Yu, N., Hua, G.: Coherent online video style transfer. CoRR **abs/1703.09211** (2017)
3. Collobert, R., Kavukcuoglu, K., Farabet, C.: Torch7: A matlab-like environment for machine learning. In: BigLearn, NIPS Workshop (2011)
4. Gatys, L.A., Ecker, A.S., Bethge, M.: Texture synthesis using convolutional neural networks. In: NIPS, pp. 262–270 (2015)
5. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: CVPR, pp. 2414–2423 (2016)
6. Ghiasi, G., Lee, H., Kudlur, M., Dumoulin, V., Shlens, J.: Exploring the structure of a real-time, arbitrary neural artistic stylization network. CoRR **abs/1705.06830** (2017)
7. Gupta, A., Johnson, J., Alahi, A., Fei-Fei, L.: Characterizing and improving stability in neural style transfer. CoRR **1705.02092** (2017)
8. Hays, J., Essa, I.: Image and video based painterly animation. In: Proceedings of the 3rd International Symposium on Non-photorealistic Animation and Rendering, NPAR, pp. 113–120 (2004)
9. Huang, H., Wang, H., Luo, W., Ma, L., Jiang, W., Zhu, X., Li, Z., Liu, W.: Real-time neural style transfer for videos. In: CVPR (2017)
10. Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., Brox, T.: Flownet 2.0: Evolution of optical flow estimation with deep networks. In: CVPR (2017)
11. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: ICML (2015)
12. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: ECCV, pp. 694–711 (2016)
13. Li, C., Wand, M.: Combining markov random fields and convolutional neural networks for image synthesis. In: CVPR, pp. 2479–2486 (2016)
14. Li, C., Wand, M.: Precomputed real-time texture synthesis with markovian generative adversarial networks. In: ECCV, pp. 702–716 (2016)
15. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollr, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: ECCV (2014)
16. Litwinowicz, P.: Processing images and video for an impressionist effect. In: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH, pp. 407–414 (1997)
17. Luan, F., Paris, S., Shechtman, E., Bala, K.: Deep photo style transfer. arXiv preprint arXiv:1703.07511 (2017)
18. Marszalek, M., Laptev, I., Schmid, C.: Actions in context. In: CVPR, pp. 2929–2936 (2009)
19. Nikulin, Y., Novak, R.: Exploring the neural algorithm of artistic style. CoRR **abs/1602.07188** (2016)
20. O'Donovan, P., Hertzmann, A.: Anipaint: Interactive painterly animation from video. Transactions on Visualization and Computer Graphics **18**(3), 475–487 (2012)
21. Revaud, J., Weinzaepfel, P., Harchaoui, Z., Schmid, C.: Epicflow: Edge-preserving interpolation of correspondences for optical flow. In: CVPR, pp. 1164–1172 (2015)
22. Ruder, M., Dosovitskiy, A., Brox, T.: Artistic style transfer for videos. In: GCPR, pp. 26–36 (2016)
23. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. ICLR (2015)
24. Sundaram, N., Brox, T., Keutzer, K.: Dense point trajectories by gpu-accelerated large displacement optical flow. In: ECCV, pp. 438–451 (2010)
25. Ulyanov, D., Lebedev, V., Vedaldi, A., Lempitsky, V.S.: Texture networks: Feed-forward synthesis of textures and stylized images. In: ICML, pp. 1349–1357 (2016)
26. Ulyanov, D., Vedaldi, A., Lempitsky, V.S.: Instance normalization: The missing ingredient for fast stylization. CoRR **abs/1607.08022** (2016)
27. Weinzaepfel, P., Revaud, J., Harchaoui, Z., Schmid, C.: DeepFlow: Large displacement optical flow with deep matching. In: ICCV, pp. 1385–1392 (2013)
28. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions. In: ICLR (2016)
29. Zhang, H., Dana, K.J.: Multi-style generative network for real-time transfer. CoRR **abs/1703.06953** (2017)

# A Appendix

## A.1 Supplementary videos

A supplementary video, available at `https://youtu.be/SKql5wkWz8E`, shows moving sequences corresponding to figures from this paper, plus a number of additional results:

- Results of the optimization-based algorithm on different sequences, including a comparison of the basic algorithm and the multi-pass and long-term algorithm
- Comparison of "naive" ($\mathbf{c}$) and "advanced" ($\mathbf{c}_{long}$) weighting schemes for long-term consistency
- Results of the feed-forward algorithm on different sequences, including results from different techniques to reduce the propagation of errors.
- Comparison between optimization-based and fast style transfer
- A demonstration of our panoramic video algorithm

Another video, showing a full panoramic video in 360°, can be found at `https://youtu.be/pkgMUfNeUCQ`.

## A.2 Style images and parameter configuration

### *A.2.1 Optimization-based approach*

Figure A.1 shows the style images chosen to evaluate the optimization-based approach and to perform the user study (except Composition VII), inspired by the selection of styles by [5].

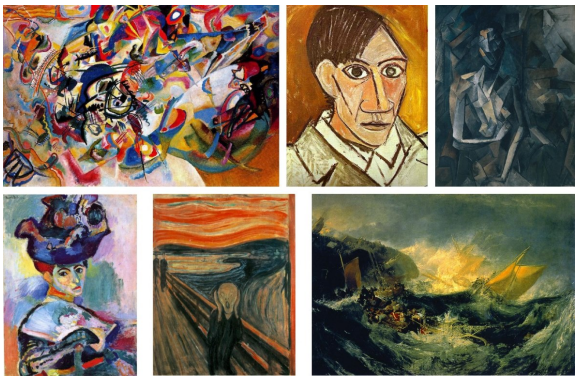

**Fig. A.1** Styles used for experiments on Sintel. Left to right, top to bottom: "Composition VII" by Wassily Kandinsky (1913), Self-Portrait by Pablo Picasso (1907), "Seated female nude" by Pablo Picasso (1910), "Woman with a Hat" by Henri Matisse (1905), "The Scream" by Edvard Munch (1893), "Shipwreck" by William Turner (1805).

### *A.2.2 Network-based approach*

Figure A.2 shows the style images used for the detailed analysis of the network-based approach and for spherical image and video evaluation, inspired by the selection of styles by [12]. In Table 8, the parameters for training individual models are shown.

**Fig. A.2** Style images used for the evaluation. From left to right, top to bottom: *Woman with a Hat* by Henri Matisse (1905), Self-Portrait by Pablo *Picasso* (1907), *The Scream* by Edvard Munch (1893), collage of the painting June Tree by Natasha Wescoat (also referred to as *Candy* by Johnson et al.), glass painting referred to as *Mosaic* by Johnson et al.

| Param | WomanHat | Picasso | Candy | Mosaic | Scream |
|---|---|---|---|---|---|
| Style Scale | 384 | 384 | 384 | 512 | 384 |
| Content Weight | 1 | 1 | 1 | 1 | 1 |
| Style Weight | 20 | 10 | 10 | 10 | 20 |

**Table 8** The individual training parameters for the style images.

## A.3 Effect of errors in optical flow estimation

The quality of results produced by our algorithm strongly depends on the quality of optical flow estimation. This is illustrated in Figure A.3. When the optical flow is correct (top right region of the image), the method manages to repair the artifacts introduced by warping in the disoccluded region. However, erroneous optical flow (tip of the sword in the bottom right) leads to degraded performance. The optimization process partially compensates for the errors (sword edges get sharp), but cannot fully recover.

## A.4 Robust loss function for temporal consistency in the optimization-based approach

We tried using the more robust absolute error instead of squared error for the temporal consistency loss. The weight for the temporal consistency was doubled in this case. Results are shown in Figure A.4. While in some cases (left example in the figure) the absolute error leads to slightly improved results, in other cases (right example in the figure) it causes large fluctuations. We therefore stick with the mean squared error in all our experiments.

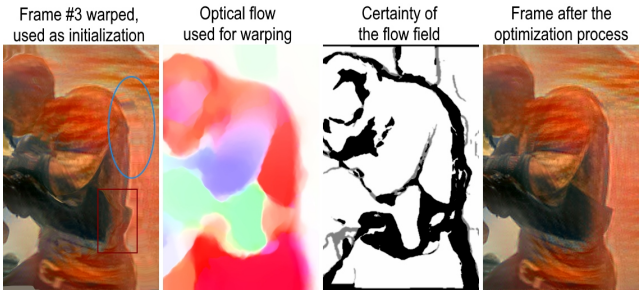


**Fig. A.3** Scene from the Sintel video showing how the algorithm deals with optical flow errors (red rectangle) and disocclusions (blue circle). Both artifacts are somehow repaired in the optimization process due to the exclusion of uncertain areas from our temporal constraint. Still, optical flow errors lead to imperfect results. The third image shows the uncertainty of the flow filed in black and motion boundaries in gray.

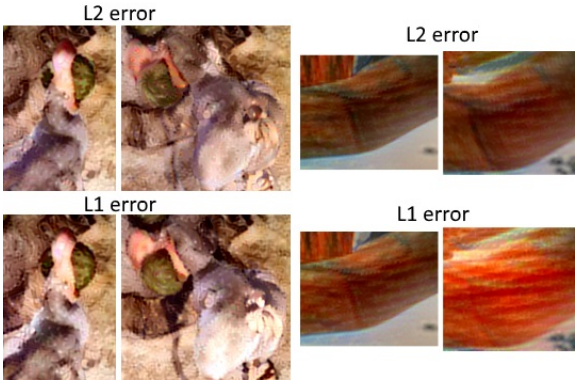


**Fig. A.4** **Left**: Scene from Ice Age (2002) where an absolute error function works better, because the movement of the bird was not captured correctly by the optical flow. **Right**: Extreme case from Sintel movie where a squared error is clearly superior.

## A.5 Batch and instance normalization.

Batch normalization [11] normalizes the feature activations for individual feature maps in a mini-batch after each layer of a neural network. This has been found to be advantageous especially for very deep neural networks, where the variance over the output is likely to shift during training. Let $x \in \mathbb{R}^{B\times C\times H\times W}$ be a tensor with batch size $B$, $C$ channels and spatial dimensions $H \times W$, and let $x_{bchw}$ be the $bchw$-th element in this tensor. Then, the mean and the variance are calculated as $\mu_c = \frac{1}{BHW}\sum_{b=1}^{B}\sum_{h=1}^{H}\sum_{w=1}^{W} x_{bchw}$ and $\sigma_c^2 = \frac{1}{BHW}\sum_{b=1}^{B}\sum_{h=1}^{H}\sum_{w=1}^{W}(x_{bchw}-\mu_c)^2$.

The batch normalization layer performs the following operation to compute the output tensor $y$:

$$y_{bchw} = \gamma\frac{x_{bchw}-\mu_c}{\sqrt{\sigma_c^2+\epsilon}} + \beta, \tag{20}$$

with learnable scale and shift parameters $\gamma$ and $\beta$.

In contrast, instance normalization normalizes every single instance in a batch, that is, contrast normalization is performed for each individual instance. Therefore, the mean and the variance are computed per instance and feature map. We define a separate mean and variance for each instance as $\mu_{bc} = \frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W} x_{bchw}$ and $\sigma_{bc}^2 = \frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W}(x_{bchw}-\mu_{bc})^2$.

The instance normalization layer performs the following operation to compute the output tensor $y$:

$$y_{bchw} = \frac{x_{bchw}-\mu_{bc}}{\sqrt{\sigma_{bc}^2+\epsilon}}. \tag{21}$$

## A.6 Comparison of different methods to reduce the propagation of error in the network approach

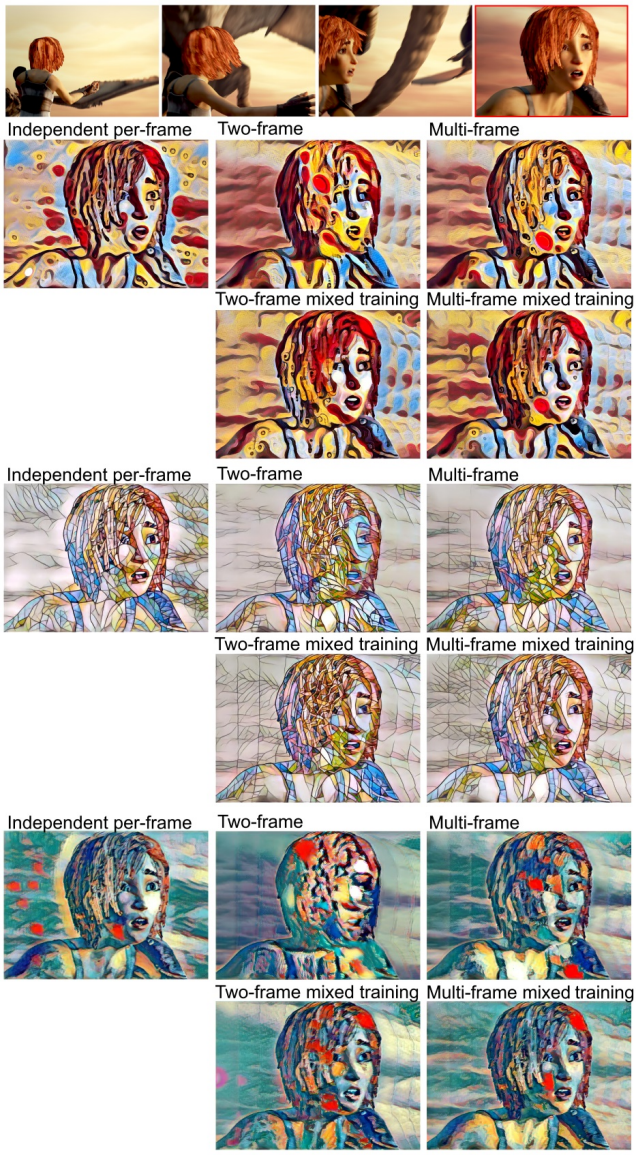


**Fig. A.5** Comparison of quality: Even for scenes with fast-motion, our advanced approaches retain visual quality and produce a result similar to Johnson et al. applied per-frame. The straightforward two-frame training, though, suffers from degeneration of quality.

## A.7 Convergence of our style transfer network

Our network converges without overfitting, as seen in Figure A.6. For the validation loss, we always use the average loss of 5 consecutive frames, processed recursively, so that the validation objective stays constant during the training of the multi-frame model. We used 120,000 iterations for the sake of accuracy, but Figure A.6 also indicates that the training can be stopped earlier in practice.

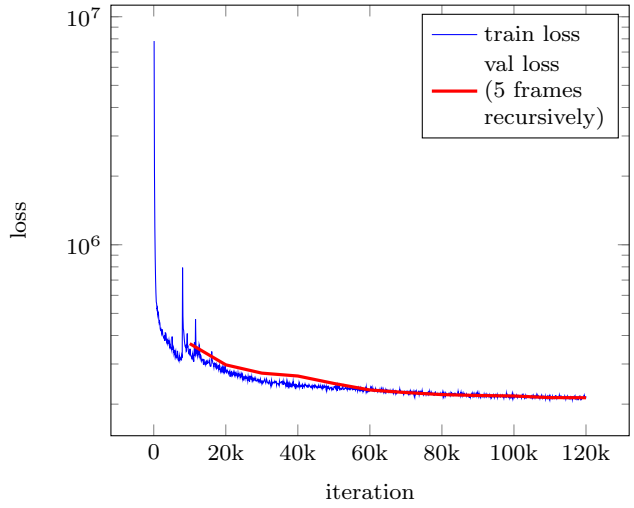


**Fig. A.6** Training loss (blue) and validation loss (red) for a multi-frame training on a logarithmic scale. Length of the frame sequence used for training, depending on the iteration number: 0-50k: 2; 50k-60k: 3; 60k-120k: 5. The validation loss is computed every 10k iterations (starting from iteration 10k) and is always the average loss for 5 consecutive frames processed recursively. Therefore the validation loss is larger than the training loss in the beginning, but decreases as our multi-frame training begins.

## A.8 Reprojection for border consistency in spherical images

For perspective transformation of a border region we virtually organize the cube faces in a three dimensional space, so that we can use 3D projection techniques such as the pinhole camera model. According to the pinhole camera model, for a point $(x_1, x_2, x_3)$ in a three-dimensional Cartesian space, the projection $(y_1, y_2)$ on the target plane is calculated as:

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = -\frac{d}{x_3} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \tag{22}$$

where $d$ is the distance from the projection to the origin. We arrange the already stylized cube face at an angle of 90° to the projection plane to project its border into the plane of another cube face.

## A.9 Evaluation metric for spherical images

To evaluate if there are unusually high gradients in a given region compared to the rest of the image, we calculate the ratio of gradient magnitudes in that region to gradient magnitudes in the overal image. We take the maximum color gradients per pixel, i.e. we define our image gradient in $x$ direction as $G_x(\mathbf{p}) = \text{cmax}\Big(\mathbf{p}[red]_x, \mathbf{p}[green]_x, \mathbf{p}[blue]_x\Big)$. On that basis, we define the ratio $r_x = \frac{||\ \mathbf{s}_x \circ G_x||_1}{||\mathbf{s}_x||_1} \Big/ \frac{||G_x||_1}{D}$ where $D$ is the dimensionality of the image and $\mathbf{s}$ is a binary vector which encodes the region where we want to test for unusual gradients (in $x$ direction), such that the vector is 1 in the desired region and 0 everywhere else. By $\circ$ we denote the element-wise multiplication.

The error metric $E_{gradient}$, which is a weighted average between horizontal and vertical gradients, is then calculated as follows:

$$E_{gradient} = \frac{||\mathbf{s}_x||_1 \cdot r_x + ||\mathbf{s}_y||_1 \cdot r_y}{||\mathbf{s}_x||_1 + ||\mathbf{s}_y||_1} \tag{23}$$