Ninth International Conference on Intelligent Control and Information Processing
November 9-11, 2018; Wanzhou, China

Frame Difference-Based Real-Time Video Stylization in Video Calls

Zheyang Xiong
Nansha College Preparatory Academy,
511458 Guangzhou, China

Cornelius Weber
University of Hamburg,
Department of Informatics,
22527 Hamburg, Germany

Xiaolin Hu
Tsinghua University,
Department of Computer Science and Technology,
Beijing 100084, China
Email:xlhu@mail.tsinghua.edu.cn

***Abstract*—The naive video stylization method is to perform neural-style transfer on individual frames, but this method would result in a flickering effect, which is particularly visible in static regions. Previous remedies extract optical flow from a video and use this information to stabilize the stylized videos. However, computing optical flow is complex and time-consuming. We consider stylizing videos in which the background is fixed and only the foreground object moves, which is the case in video calls. We propose a simple method to stylize such videos in real time based on frame difference. The main idea is to use the frame difference to detect foreground and rebuild it in the next frame while maintaining the stylized background from the previous frame. This method is easy to implement and can stylize videos in real time with stabilized frames.**

Keywords*—Deep learning, real-time video processing, image stylization, morphology transformation, computer vision.

I. Introduction

As the internet is becoming increasingly connected nowadays, video call is becoming more popular. People can communicate "face-to-face" using this technology. In most video call software, while there are many fancy visual effects such as colorful filters and animated gif, some users would also want their video calls to be artistic, by which the videos will have the same style as a famous painting.

The idea of generating an image with a style of a famous painting, style transfer, was proposed in [1]. They proposed to use a pre-trained convolutional neural network that can mathematically capture the style (using a Gram Matrix) and content of various images. Given a work of fine art, this approach can take an arbitrary image and generate an optimized image with the same content as the original, but in the style of the work of [2] proposed another method that trains a feed-forward CNN and renders a stylized image immediately when input the content image and style image. These image style transfer approaches have been widely used in artistic software like Prisma and Artisto.

However, when methods of image style transfer are applied to videos frame by frame, they will produce flicker artifacts in the stylized video, as is shown in Fig. 1. To solve this issue, [3] proposed a new way to transfer style in video sequences by adding a temporal constraint. Their approach relies on optical flow calculation. They extract the optical flow from the video to calculate the temporal and long-term constraints, penalize the deviation along point trajectories, and make the video sequence stable. Other methods based on feed-forward networks are also proposed to stylize video with a faster speed [4]–[6].

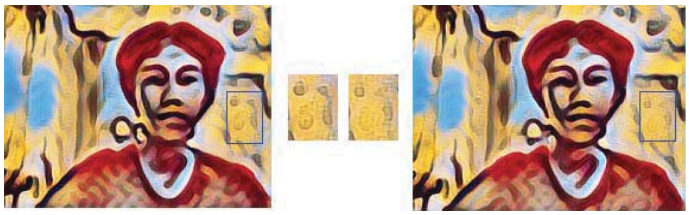

Fig. 1. Style transfer in video that produce flicker artifact. Two images are consecutive stylized frames. The zoom-in areas from the blue rectangle show flicker artifacts where the backgrounds of stylized frames are inconsistent between two consecutive frames.

To stylize consistent videos, i.e. videos without flicker, with faster speed, we propose a new method that captures the deviation between each frame, the frame difference. In our method, we first stylize the current frame by a feed-forward network [2]. In order to make the stylization consistent, we calculate the difference in gray channel between current frame and previous frame and determine which part of the frame is in motion. Then, using a binary mask, we only update the area that is in motion (foreground) and preserve the motionless area (background). This new methodology can compute consistent stylized video sequences with a speed much faster than using optical flow calculation. Since our method utilizes frame-difference to detect foreground and background, it is based on the assumption that the camera itself rarely moves, which can be applied in video calls.

II. Related Works

A. Style Transfer on Images

1) Optimization-based: The ground-breaking paper [1] finds an impressively effective method with a descriptive network. They notice that when reconstructing images from feature maps in a pre-trained VGG-19 [7], it will keep the details of content in lower layers, and style in upper layers. They then proposed an approach to optimize a new image (starting from Gaussian white noise) that has similar neural activation in feature maps of a given content image in lower layers while also having similar correlations (as defined by an

978-1-5386-5860-4/18/$31.00 ©2018 IEEE

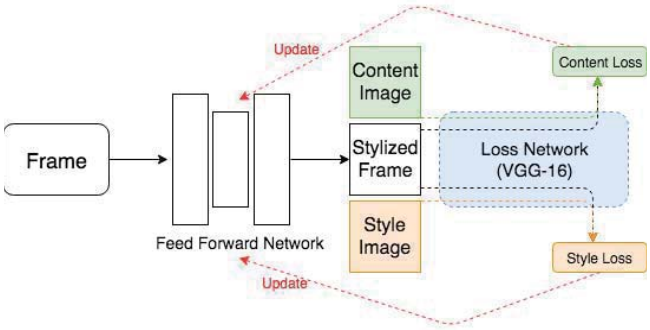

Fig. 2. The architecture of the feed-forward network based on [2].

inner product of feature maps) as a given style image in its upper layers. They define two loss functions for optimization: style loss for learning style features and content loss for preserving the original content information. After sufficient epochs of optimization, the new image will have the style from the style image while preserving the content from the content image.

Nevertheless, optimization-based style transfer is extremely time-consuming because an optimization process is needed for each image to be stylized.

2) Network-based: In order to address the shortcoming of optimization, [2] proposed to train a feed-forward generative image transform CNN by using a similar perceptual loss in corresponding style and content layers defined in the loss network (VGG-16) [7], which eliminates the time-consuming optimization. [8] further develop this method by suggesting instance normalization to replace batch normalization .

B. Style Transfer on Videos

However, when methods of image style transfer are implemented on videos (i.e., stylizing every individual frame with a single style image), the output video sequences will have flicker artifacts and inconsistent video frames.

1) Optimization-based: In order to make stylized video temporally consistent, [3] introduce a method for video stylization based on optimization . The optical flows between frames are extracted (using DeepFlow [9]) and taken into account to calculate the temporal constraint that penalizes deviation along point trajectories. This process allows the areas that have not changed or been occluded to be initialized with the desired appearance on the next frame, while the occluded regions are rebuilt based on the optimization. Furthermore, a long-term constraint can be calculated between a frame and another frame several frames before. This method can generate frames that have consistent correspondences with previous frames. Thus, the whole video sequence is consistent and stable.

2) Network-based: However,the approach in [3] can bring huge computational burden, which is not suitable for real-time video stylization. [4] proposed another method to achieve real-time video stylization . They mimic the architecture of feed-forward network in [2] and add temporal loss on the network. Their temporal loss also needs optical flow calculation, and

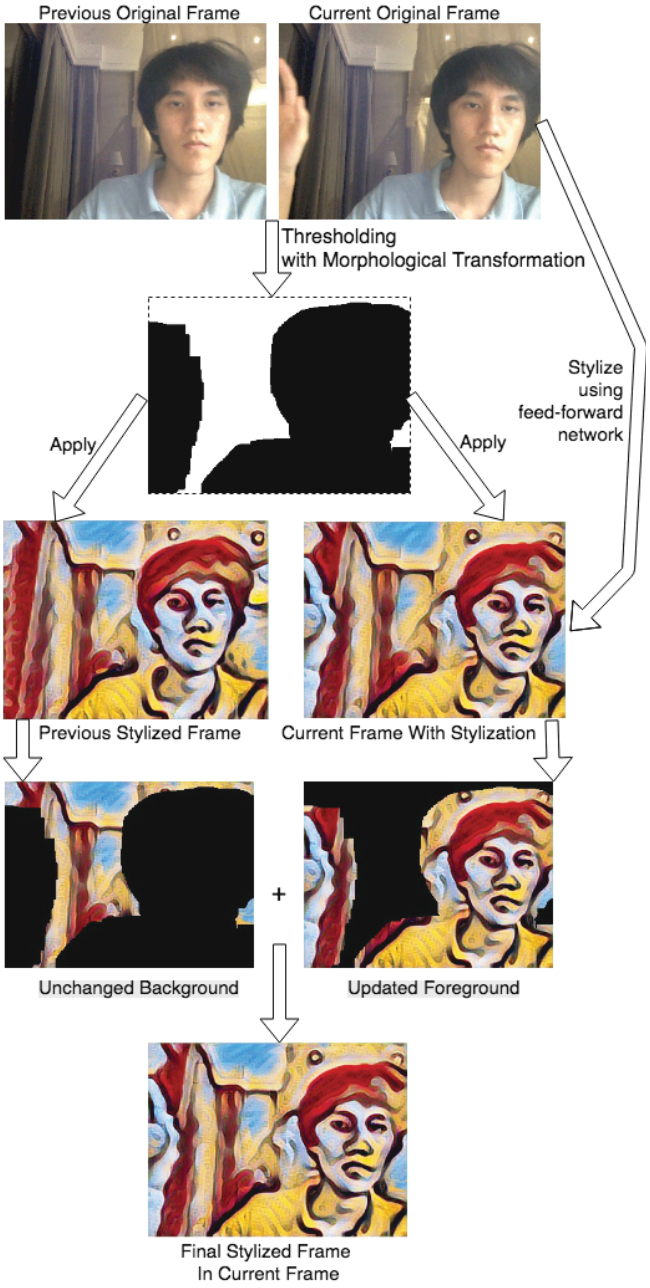

Fig. 3. The diagram of our method. To obtain the current stylized frame, we first generate a binary mask from consecutive original frames; then we apply the mask on the last stylized frame to preserve the background and use the mask on the new stylized frame to produce the new foreground; then background and foreground are combined to eventually form the current stylized frame.

they train the network by feeding two consecutive frames to enhance pixel-level temporal consistency. After the generative network is trained, no optical calculation is needed in the test stage. On the other hand, [5] proposed a recurrent convolutional neural network that also does not need optical flow at test time . In this manner, feed-forward networks in [4], [5] can yield temporally consistent stylized videos in real-time.

Recently, a new network-based method was proposed in

[6] that can yield consistent stylized videos, which is an improvement on their optimization-based method. This reduces runtime by utilizing a faster optical flow extractor – FlowNet 2.0 [10]. They also construct a video transfer network, in which consistency loss is measured with warped frame and output frame. Their new method does need optical flow at test time, but FlowNet ensures a fast optical flow extraction speed.

III. Frame-Difference-Based Real-Time Stylization In Videotelephony

To make video stylization fast and consistent, we propose a new method that only stylizes the foreground (such as human faces, body, and hand gestures) but keeps the background unchanged. Because in videotelephony, the camera rarely moves, we use frame-difference to discern between background and foreground. Our stylization part is based on the method in [2], but we use frame difference to detect foreground and only update foreground in stylized video to make stylization consistent.

A. Feed-forward Network for Stylization

We first stylize each new image by using feed-forward network in [2]. This architecture combines a feed-forward network with a loss network (VGG-16), shown in Fig. 2. During training, a stylized image will first be computed through a feed-forward network. Then, iteratively, a random content image from a large dataset and a target style image will be used to calculate content loss and style loss via the loss network (VGG-16). Through back-propagation and optimization of the two losses, the feed-forward network will be trained. Once the feed-forward network is trained, each stylized image passed through the feed-forward network will preserve the spatial original features of the original while adding the stylistic features from the style image. We then only update the foreground of the stylized frame to make the video consistent.

B. Frame Difference for Foreground Detection

In order to yield temporal consistency, we expect the background (the area that does not move) to remain unchanged throughout the video sequence while re-stylizing the foreground (area of object that moves).

First, we define $F^t \in R^{H\times W}$ as a matrix containing pixel information in the grey channel at the t-th original frame (the current frame), where the gray channel is calculated by the average value of the R, G, and B channels. Thus, we can obtain the frame differences [11] between every adjacent pair of frames to calculate pixel differences from frame to frame as:

$$D^t = F^t - F^{t-1} \tag{1}$$

where F^{t-1} is a matrix of pixel information for the $(t-1)$-th frame (the previous frame); $F^t \in R^{H\times W}$ is a matrix of pixel information for the t-th frame (the current frame), and $t = 1, 2, 3, ...$ for F^t; $D^t \in R^{H\times W}$ is a matrix containing pixel differences between the t-th frame and the $(t-1)$-th frame, and $t = 2, 3, 4, ...$ for D^t.

We can thereafter judge whether the pixel at a particular location is in motion from the $(t-1)$-th frame to the t-th frame by setting up a threshold value. We define a value $d^t_{i,j}$, which is the value of frame difference at pixel (i, j) between the t-th frame and the $(t-1)$-th frame, where $i = 1, 2, , H$, $j = 1, 2, , W$. If $d^t_{i,j}$ exceeds the threshold value, we consider the pixel at location (i, j) to be in motion from the $(t-1)$-th frame to the t-th frame. We thereafter define a binary mask B^t to judge whether the pixel is in motion from F^{t-1} to F^t, where $b^t_{i,j}$ is the pixel information at location (i, j) in B^t,

$$b^t_{i,j} = \begin{cases} 1, |d^t_{i,j}| \le T \\ 0, |d^t_{i,j}| > T. \end{cases} \tag{2}$$

By this equation, the pixel at (i, j) is determined to be in motion between the $(t-1)$-th frame and the t-th frame if $d^t_{i,j}$ exceeds the threshold value T. Else, it is considered background.

Then, we keep the stylized background from the $(t-1)$-th frame and add stylized areas in motion in the t-th frame:

$$S^t_{new} = S^{t-1} \odot B^t + S^t \odot (\mathbf{1} - B^t), \tag{3}$$

where S^t is the stylized t-th frame using feed-forward network in [2], B^t is being duplicated three times to become a binary mask in RGB channels. The $\odot$ calculation is the Hadamard product between matrices. $\mathbf{1} \in \mathcal{R}^{H\times W}$ is a matrix full of 1. The complete process of our method is illustrated in Fig. 3.

C. Morphology Transformation

Usually, the frame difference cannot fully detect the foreground of the frame, so the binary mask B^t is incomplete on the foreground while some noise in the background might be erroneously detected as foreground. Thus, we use morphology transformation on B^t to enlarge the area of the foreground and to prevent detecting background as foreground due to noise.

Morphology transformation, including erosion and dilation, is widely used in computer vision. It is basically an operation that deals with digital images, especially binary images. Judging whether a pixel is background or not only using a threshold on frame difference can incur much noise and only capture the lineament of the foreground. Fig. 4 shows that simply using the frame difference method to detect foreground between two frames is not effective. We can only capture the outline of foreground, and there is much noise in the background that is incorrectly detected as foreground. We therefore apply the closing operation to the binary mask to remove noise and fill the area of foreground.

Closing is a morphological operation on images; it consists of dilation and erosion [12]. The equation for closing is

$$A \bullet B = (A \oplus B) \ominus B, \tag{4}$$

where $\oplus$ and $\ominus$ denote dilation and erosion, respectively.

Let A be a binary mask consisting of 0 and 1, B a morphological kernel with kernel size k. The dilation of A and B is defined [12] by:

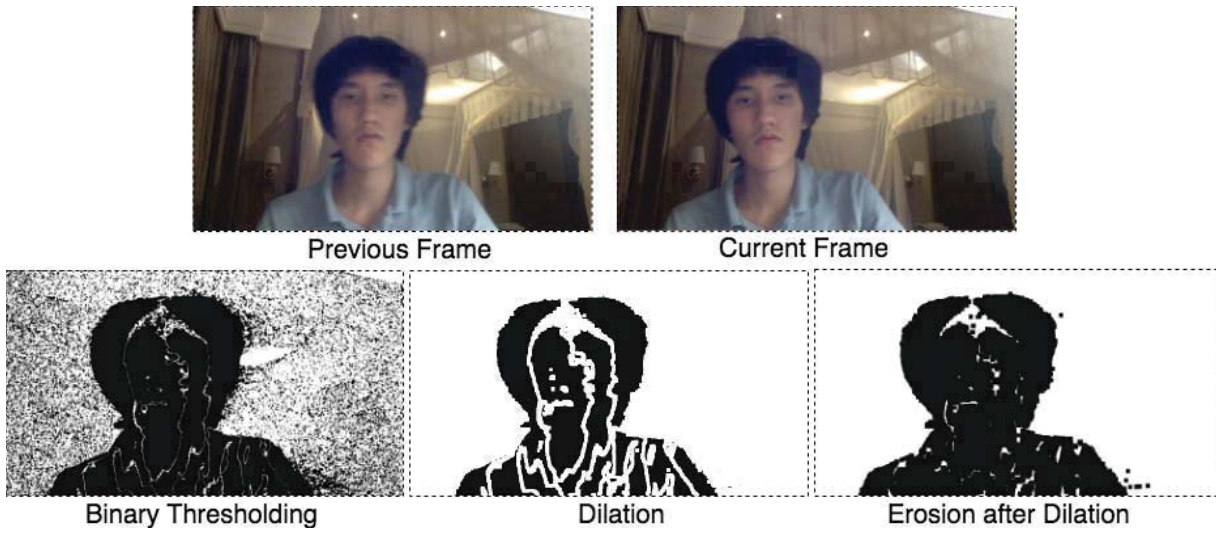

Fig. 4. An example of morphological transformations. The first row shows two original frames: the first figure is the previous frame F^{t-1}, and the second figure is the current frame F^t. The first figure of the second row shows a binary mask B^t; the second figure shows the outcome of dilation; the third figure shows the outcome of erosion after dilation.

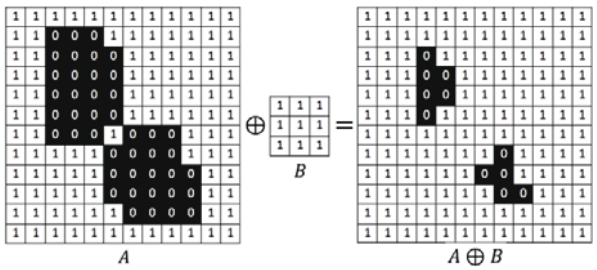

Fig. 5. An example of dilation where B is a 3×3 square matrix full of 1.

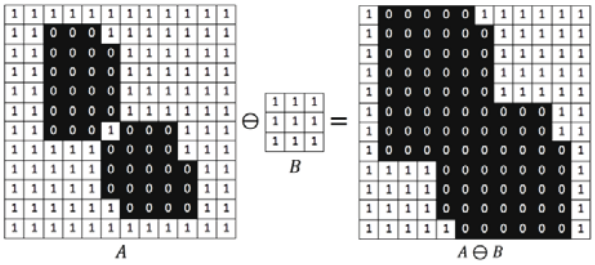

Fig. 6. An example of erosion where B is a 3×3 square matrix full of 1.

$$A \oplus B = \bigcup_{b \in B} A_b, \tag{5}$$

where A_b is defined by the translation of A by b. See Fig. 5 for an example.

The erosion of the binary image A to a kernel B is defined [12] by:

$$A \ominus B = \bigcap_{b \in B} A_{-b}, \tag{6}$$

where A_{-b} is the translation of A by $-b$. See Fig. 6 for an example.

Since erosion can expand the black area and dilation can expand the white area, we need to dilate the noise area in B^{t-1} first in order to prevent making the noise as the foreground. After that, we erode the image several times to fill the outline of the incomplete foreground and make it complete and accurate.

IV. Results

Our feed-forward network was trained and validated on the Microsoft COCO dataset [13] with 1000 epochs.

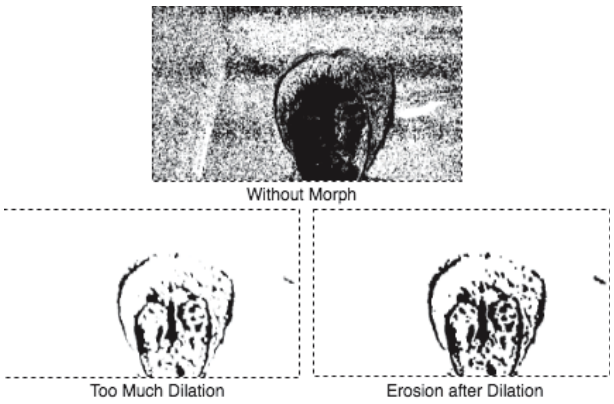

Fig. 7. An example of morphological transformation with too much dilation.

A. Foreground Detection

With binary thresholding and , we can obtain an accurate mask that distinguishes between foreground and background. Examples of morphological transformations are shown in Fig. 4. The first image of the second row shows B^t without morphological operation; as we can see, there is much pixel noise in the background being detected as foreground (black

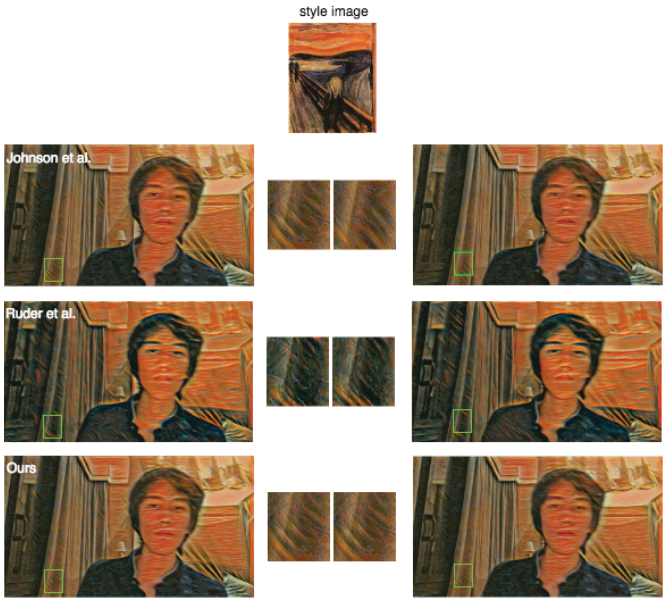

Fig. 8. The style image is *The Scream*, shown in the first row. The second row shows two consecutive stylized frames using the method in [2]; areas enclosed by green rectangles indicate stylized backgrounds are variant between the consecutive frames. The third row shows two consecutive stylized frames using the network-based method in [6]; areas enclosed by green rectangles indicate stylized backgrounds are consistent. The fourth row shows two consecutive stylized frames using our method; areas enclosed by green rectangles indicate stylized background are nearly unchanged.

pixels). The second image of the second row shows B^t with two iterations of dilation and kernel size of 3 (3×3 square matrix with all value of 1); the background noise is effectively removed from B^t while keeping the outline of the foreground. The third image of the second row shows B^t with five iterations of erosion and kernel size of 5 (5×5 square matrix with all value of 1) after dilation; most part of the foreground has been filled with black (indicating detection as foreground), and there is little noise near the background.

For morphological transformation, we recommend three or four iterations of dilation before erosion to eliminate noise while preserving the outline of foreground. If too many iterations of dilation are applied, as is shown in the second row of Fig. 7, even erosion cannot completely capture the foreground completely. We apply multiple steps of dilation/erosion because this allows us to use a smaller kernel than when applying only one step.

B. Comparison to Previous Methods

Although there are several ways to stylize videos, there is only a limited number of ways to stylize video in real-time. In our paper, we compare our method with the method in [2] and [6]. The comparison is shown in Fig. 8, from which we can see that our method and the method in [6] yield consistent stylized background. The areas enclosed by green rectangle on the third row indicate that our method improves by producing unchanged background, whereas the method in [2] produces variant background between frames. In addition, our method can produce similar consistency results as [6].

TABLE I
TIME ELAPSED FOR ONE STYLIZATION BY DIFFERENT METHODS.

Method	Time/frame	Consistency
[2]	0.11s	Flickering
[6]	0.43s	Consistent
Ours	0.13s	Consistent

Fig. 9. The orange area shows the "ghost effect" where some feature of the background is not smoothly integrated with the whole frame.

By using a NVIDIA Titan X GPU, our method can transfer one single frame with a resolution of 1024×436 within 0.13 seconds and achieve approximately 8 fps in real-time stylization (an optional choice for increasing frames per second (fps) to 25 is to resize the frame size to be 320×240). A comparison of stylization speed is shown in Table 1. Our method can produce consistent stylized video with much faster speed than [6].

C. Ghost Effect

Because we use a binary mask to update the foreground and keep the background, sometimes there is a "ghost effect" in the background, as is shown in Fig. 9. "Ghost effect" refers to some artifacts staying in the background in regions that have earlier been detected as foreground from which the object (person) has now moved away. This is caused by a wrong calculation of binary mask and a failure to update foreground. These small artifacts appear while uncovering occluded regions, but not as sudden changes, so they are hard to detect.

D. Other Results

We also implement our method on other styles, subjects, and backgrounds. The outputs are reasonably pleasant and the frames are consistent, in which they have an artistic style, see Fig. 10.

V. CONCLUSION

In conclusion, our method combines the advantage of existing methods of being fast, as [2], and not producing unpleasant flicker, as [6]. Since our approach is based on the assumption that the camera does not move, it can be used in real-time video calls. If the camera moves, or if all image content changes, flicker will not be perceived, and our method becomes equivalent to [2]. While the method implemented in

Fig. 10. Video stylization using six different styles. For each style, we stylize and display four frames, each three frames later from the previous one.

this paper is not perfect, it is a great improvement on previous efforts.

ACKNOWLEDGMENT

This work was done in the Research Science Initiative program at Tsinghua University. And this work was supported in part by the National Natural Science Foundation of China under grants nos. 61332007, 61621136008, and 61273023 and by the German Research Foundation (DFG) under project Transregio Crossmodal Learning (TRR 169).

REFERENCES

[1] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, Jun. 2016, pp. 2414–2423.

[2] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *Proceedings of European Conference on Computer Vision*, Amsterdam, Netherlands, Oct. 2016, pp. 694–711.

[3] M. Ruder, A. Dosovitskiy, and T. Brox, "Artistic style transfer for videos," in *Proceedings of Pattern Recognition: 38th German Conference, GCPR 2016*, Hannover, Germany, Sep. 2016, pp. 26–36.

[4] H. Huang, H. Wang, W. Luo, L. Ma, W. Jiang, X. Zhu, *et al.*, "Real-time neural style transfer for videos," in *Proceedings of 2017 IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, Hawaii, USA, Jul. 2017, pp. 7044–7052.

[5] A. Gupta, J. Johnson, A. Alahi, and L. Fei-Fei, "Characterizing and improving stability in neural style transfer," in *Proceedings of 2017 IEEE International Conference on Computer Vision*, Venice, Italy, Oct. 2017, pp. 4087–4096.

[6] T. B. Manuel Ruder and Alexey Dosovitskiy, "Artistic style transfer for videos and spherical images." [Online]. Available: https://arxiv.org/abs/1708.04538v3.

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition." [Online]. Available: http://cn.arxiv.org/abs/1409.1556v6.

[8] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky, "Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis," in *Proceedings of 2017 IEEE Conference on Computer Vision and Pattern Recognition,* Honolulu, HI, USA, Jul. 21-26 2017, pp. 4105–4113.

[9] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid, "DeepFlow: Large displacement optical flow with deep matching," in *Proceedings of IEEE International Conference on Computer Vision*, Sydney, Australia, Dec. 2013, pp. 1385–1392.

[10] 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 *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* , Honolulu, Hawaii, USA, Jul. 2017, pp. 1647–1655.

[11] C. Chen, J. Liang, H. Zhao, H. Hu, and J. Tian, "Frame difference energy image for gait recognition with incomplete silhouettes," *Pattern Recognition Letters*, vol. 30, no. 11, pp. 977–984, Aug. 2009.

[12] R. M. Haralick, S. R. Sternberg, and X. Zhuang, "Image analysis using mathematical morphology," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, no. 4, pp. 532–550, Jul. 1987.

[13] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, *et al.*, "Microsoft COCO: Common objects in context," in *Proceedings of European Conference on Computer Vision*, Zürich, Switzerland, May 2014, pp. 740–755.