Style Transfer to Videos using CNN

1 Topic

The object of this project is to implement video style transfer given an arbitrary video and an arbitrary style reference image. The goal is implemented by recognizing the content of the video frame and the style of the style reference image and then combining their content and style to create a new style transferred image.

2 Image Style Transfer

The video style transfer is based on image style transfer. VGG19 pretrained network is used for image style transfer.

We first can separate an image into content representation and style representation. Since VGG19 network is trained on the large image set ImageNet for object recognition, we can regard VGG19 can extract the image content very well and it is very suitable for the task of separating the content and style information.

We define the content representation as the feature map of layer 'block5_conv2' and the style representation as the correlations between the feature maps of layer 'block1_conv1', 'block2_conv1', 'block3_conv1', 'block4_conv1' and 'block5_conv1'.

So, the image style transfer means to find out a new image which minimizes the distance to content representation of the original image as well as the distance to style representation of the style reference image.

Suppose $\vec{p}$ is the original image and $\vec{x}$ is the generated image. The distance of content representation of $\vec{p}$ and $\vec{x}$ in layer l is

$$L_{content}(\vec{p},\vec{x},l)=\frac{1}{2}\sum_{i,j}\left(F_{ij}^{l}-P_{ij}^{l}\right)^2$$

where F_{ij}^{l} is the activation of the ith filter at position j in layer l of $\vec{x}$ and P_{ij}^{l} is the activation of the ith filter at position j in layer l of $\vec{p}$.

The different layer l will result in different transfer result. For example, a higher convolutional layer will have more meaningful transferred image content while a lower convolutional layer will have more blur image content. In this project, I have chosen l = 'block5_conv2' in order to preserve the most original image content.

Suppose $\vec{a}$ is the style reference image and $\vec{x}$ is the generated image. The distance of style representation of $\vec{a}$ and $\vec{x}$ is

$$L_{style}(\vec{a},\vec{x}) = \sum_{l=0}^{L} w_l E_l$$

where $E_l = \frac{1}{4N_l^2 M_l^2}\sum_{i,j}\left(G_{ij}^l - A_{ij}^l\right)^2$ is the loss of layer l, w_l is the weight of layer l, $G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l$ is the gram matrix for $\vec{x}$ in layer l and A_{ij}^l is the gram matrix for $\vec{x}$ in layer l, N_l and M_l are the height and the width of the feature map of layer l.

When matching the style representations up to higher layers in the network, local image structures are matched on an increasingly large scale, leading to a smoother and more continuous visual experience. So in this project, we have l = ['block1_conv1', 'block2_conv1', 'block3_conv1', 'block4_conv1', 'block5_conv1'] which matches the style representations up to the highest layers to have more appealing images. And we have $w_l = 0.2$ for all the five layers.

The total loss is

$$L_{loss}(\vec{p},\vec{a},\vec{x}) = \alpha L_{content}(\vec{p},\vec{x},l) + \beta L_{style}(\vec{a},\vec{x})$$

where α and β are the weights of content representation loss and style representation loss. In this project, we set the ratio of α and β to 10^{-4}.

We first can initialized the new mixed image to a white noise image, then calculate the loss function from its style and content presentation. Gradient descent can be applied to this white noise image to decrease the loss function. The Gradient descent can be implemented by back propagation on the VGG19 network.

Gradient descent from a white noise image can take a large number of iterations for it to converge to a combined image. To accelerate the process of convergence, we can initial the image with the original image instead.

So, the task of image style transfer does not need to train a new neural network, instead it uses the pretrained network VGG19 to separate the content and the style of images and uses gradient descent for combining. Thus, there is no need for training and testing data set. The only thing we need is an original image and a style referenced image, then we can output a combined image.

3 Result of Image Style Transfer

The following Figures 1 - 14 show an example of image style transfer, including the original image, the style reference image and the style transferred image at the iterations of 10, 50, 100, 200, 300, 400, 500, 600, 700, 800, 900 and 1000.

We can see that, the change of the transferred images is very tiny between each iteration. But the image after 1000 iterations is obviously more styled than that only after 10 iterations. If we continue doing the gradient descent, the transferred image will have more style structures from the style reference image. Each gradient iteration will take about 20 seconds.

4 Video Style Transfer

Applying the above image style transfer to each frame of video can get the style transfer video. Each new created frame is initialized with the original video frame.

It will take about 20 seconds for one gradient descend iteration (using Google colab's GPU). If we have 10 iterations for each video frame, it will need more than 3 minutes for a single frame.

In the paper "Artistic Style Transfer for Videos[2]" by Ruder et al, an initialization method is to initial the state of gradient descent of frame $i+1$ with the transferred image i. However, this initialization method will lead to a wired phenomenon that each frame is more stylized than the previous frame, since if only proceeding 10 iterations for each frame, that frame will not be stylized completely (the gradient descent is not convergence). So, passing the stylized frame to the next frame's initial state will make the next frame more stylized.

Thus, to avoid the different stylization of the frames, in this project, I only initial the gradient descent with the original video frames.

In this paper, there is also a temporal consistency loss penalty term added to the total loss function to increase the continuity between the video frames. The temporal consistency loss penalty requires the detection of dis occluded regions and motion boundaries.

Suppose $\boldsymbol{w}$ is the optimal flow in forward direction, $\widehat{\boldsymbol{w}}$ is the optimal flow in backward direction. Then $\widetilde{\boldsymbol{w}}$ is the forward flow warped to the second image

$$\widetilde{\boldsymbol{w}}(x,y) = \boldsymbol{w}(x,y) + \widehat{\boldsymbol{w}}(x,y)$$

In the areas of dis occlusion, the warped flow is approximately the opposite of the backward flow. So, the dis occlusion areas can be detected by where the inequality holds

$$|\widetilde{\boldsymbol{w}} + \widehat{\boldsymbol{w}}|^2 > 0.01(|\widetilde{\boldsymbol{w}}|^2 + |\widehat{\boldsymbol{w}}|^2) + 0.5$$

Motion boundaries are detected by the following inequality

$$|\nabla\hat{x}|^2 + |\nabla\hat{y}|^2 > 0.01|\widehat{\boldsymbol{w}}|^2 + 0.002$$

Then, the temporal consistency loss penalty term can be written as

$$L_{temporal}(x,w,c) = \frac{1}{D}\sum_{k=1}^{D} c_k \cdot (x_k - w_k)^2$$

Where $c \in [0,1]$ is weight of each pixel's loss, $c^{(i-1,i)}$ between frames $i-1$ and i equals to 0 in dis occluded regions and at motion boundaries , and equals to 1 at everywhere else.

Then, the total loss can be written in the form

$$L_{loss}\left(p^{(i)}, a, x^{(i)}\right) = \alpha L_{content}\left(p^{(i)}, x^{(i)}, l\right) + \beta L_{style}\left(a, x^{(i)}\right) + \gamma L_{temporal}\left(x^{(i)}, w_{i-1}^{i}\left(x^{(i-1)}\right), c^{(i-1,i)}\right)$$

5 Result of Video Style Transfer

I have done style transfer for two pieces of video with two different style reference images. The first video is transferred with 5 iterations of gradient descent each frame. The second video is transferred with 50 iterations of gradient descent each frame.

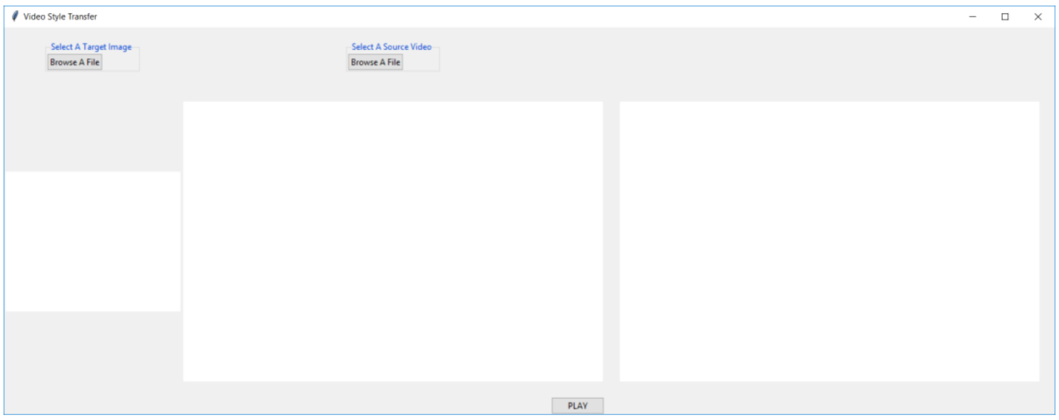
The figures 15 – 28 show some examples of the transferred video frames. We can see that frames with 50 iterations are more stylized than those with only 5 iterations.

For 4 iterations, it might take about 2 minutes each frame with Google's colab GPU. For 50 iterations, it might take about 20 minutes each frame. So, it certainly cannot finish processing in the real-time in the GUI demo where I just show two already transferred videos in the GUI.
Link to the GUI demo: **https://youtu.be/_CorC_JRoXw**

Link to the GitHub is: https://github.com/ruiwan1996/CSCE636-Video-Transfer

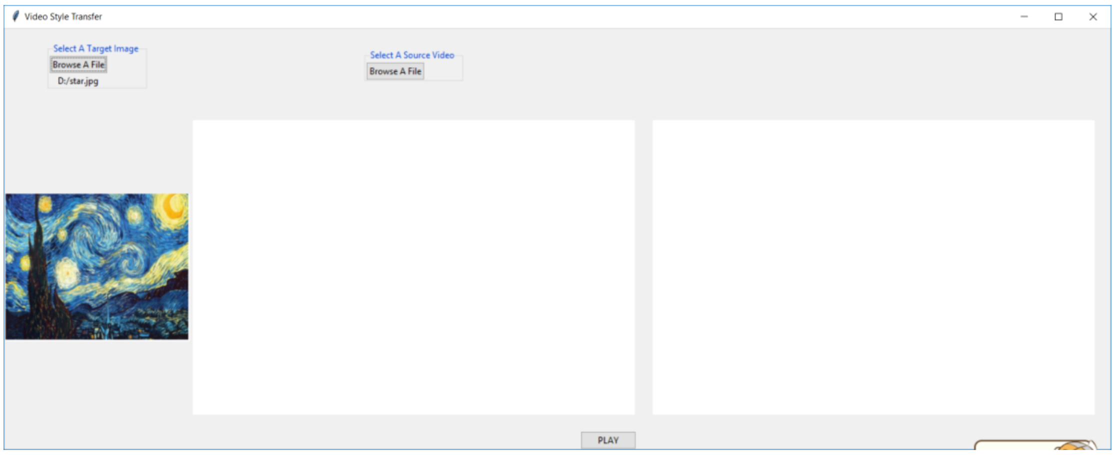
Note: The temporal consistency improvement part of the project is not finished yet. I will update the GUI demo and the GitHub.

6 About GUI



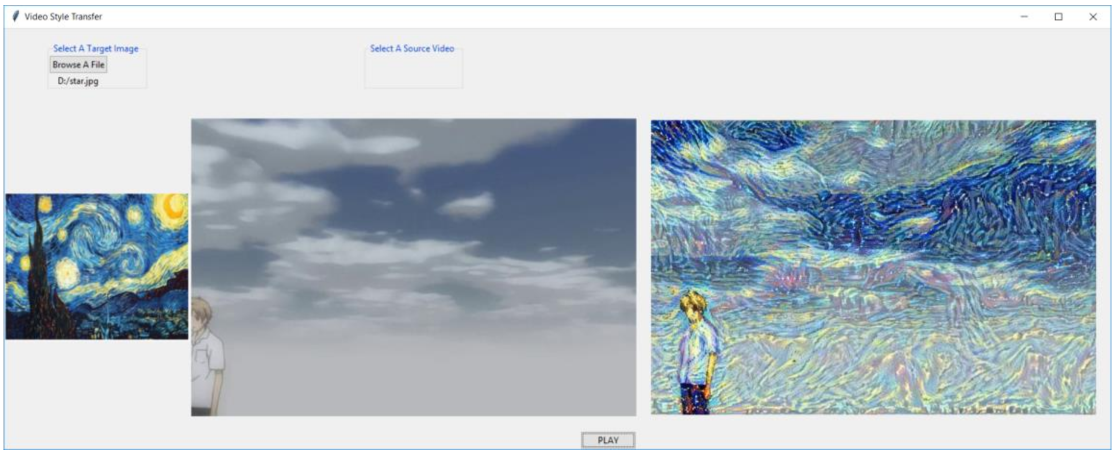
The initial GUI looks like the above.

First we can click ‘Select A Target Image – Browse A File’ and choose a style reference image. After picking the style reference image, the image will be shown on the GUI.



Next, we can click ‘Select A Source Video – Browse A File’ and choose a video. After choosing the video will not be played on the GUI, until we click the ‘PLAY’ button.

After we click on ‘PLAY’, both the original video and the transferred video will be shown on the GUI.



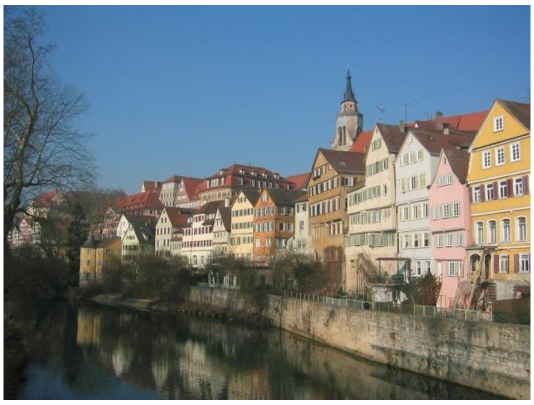


Figure 1 Original Image

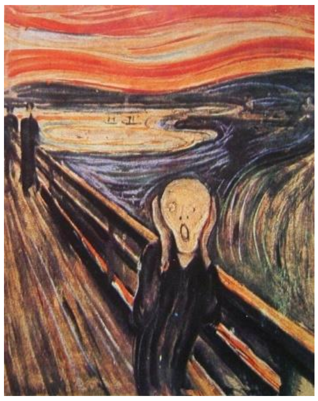


Figure 2 Style Reference Image

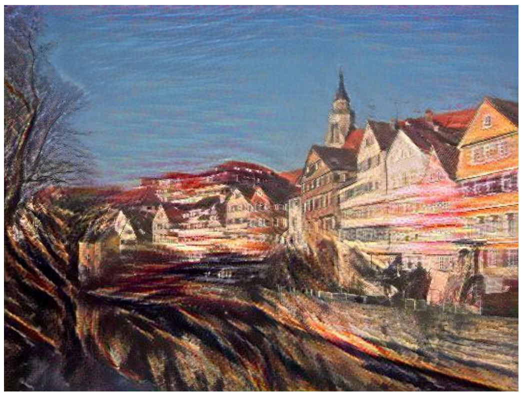


Figure 3 Iteration 10

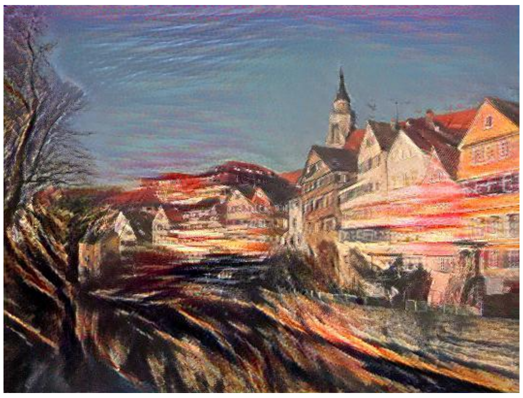


Figure 4 Iteration 50

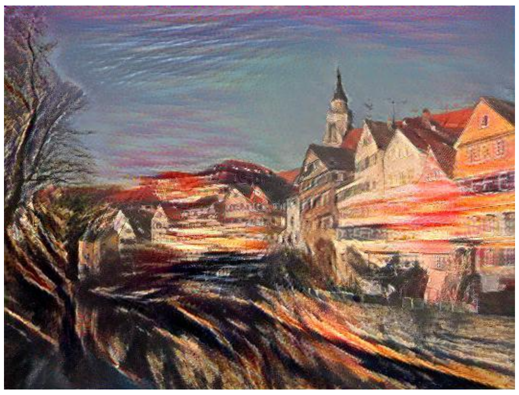


Figure 5 Iteration 100

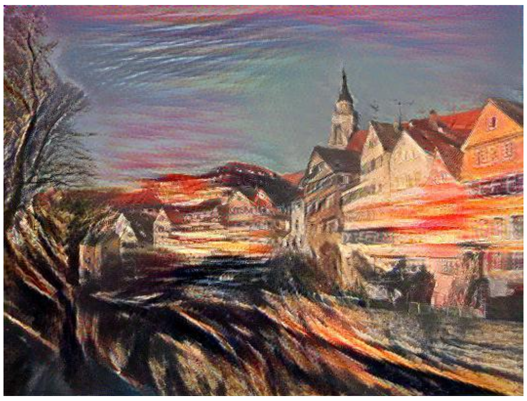


Figure 6 Iteration 200

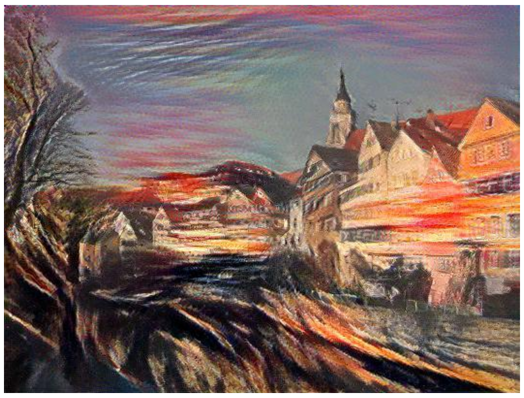


Figure 7 Iteration 300

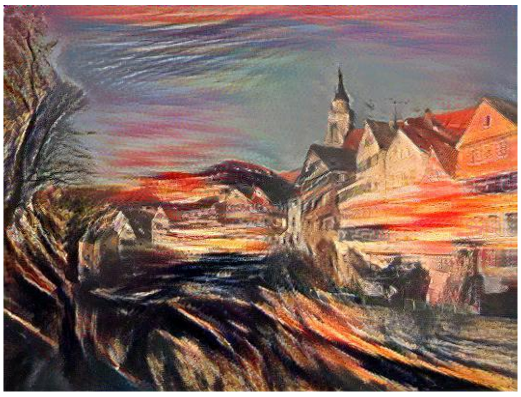


Figure 8 Iteration 400

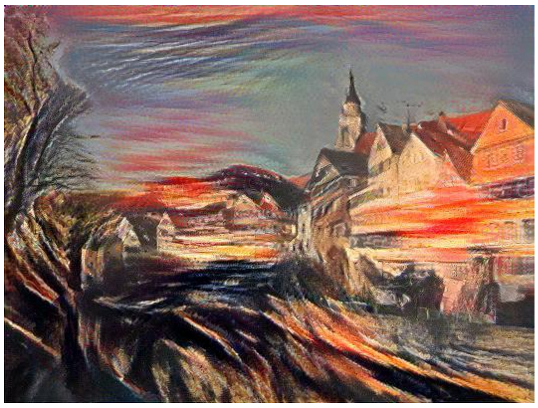


Figure 9 Iteration 500

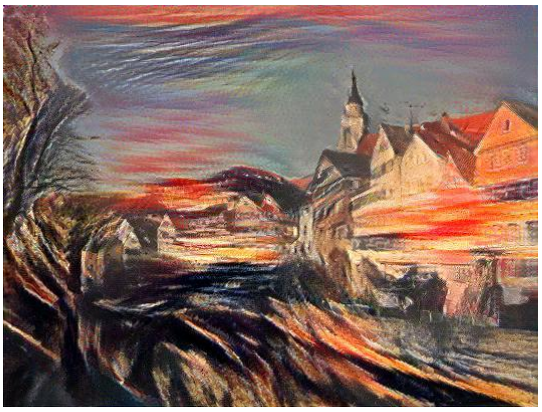


Figure 10 Iteration 600

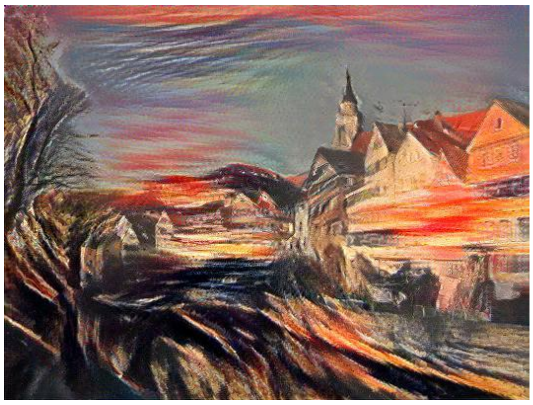


Figure 11 Iteration 700

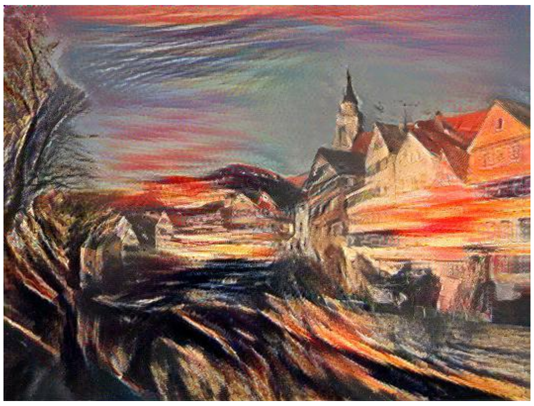


Figure 12 Iteration 800

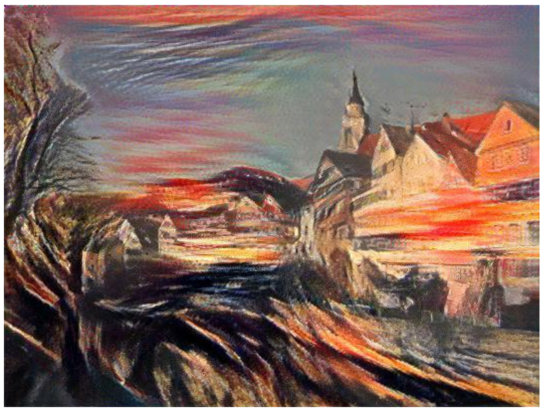


Figure 13 Iteration 900

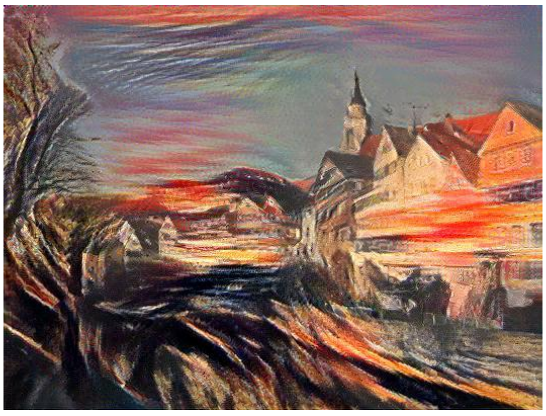


Figure 14 Iteration 1000

Example of Image Style Transfer

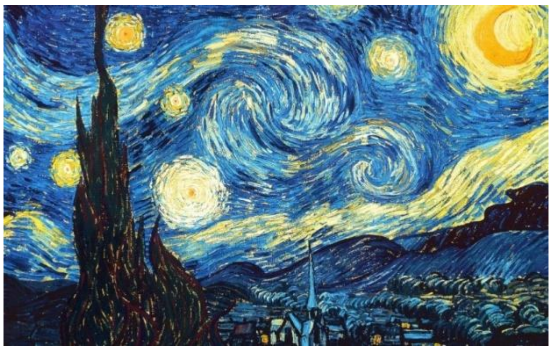


Figure 15 Style Reference Image

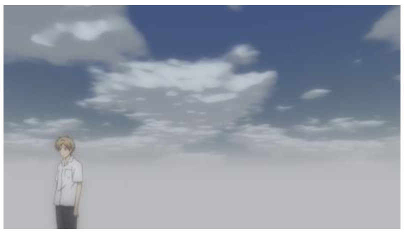


Figure 16 Original Frame 24

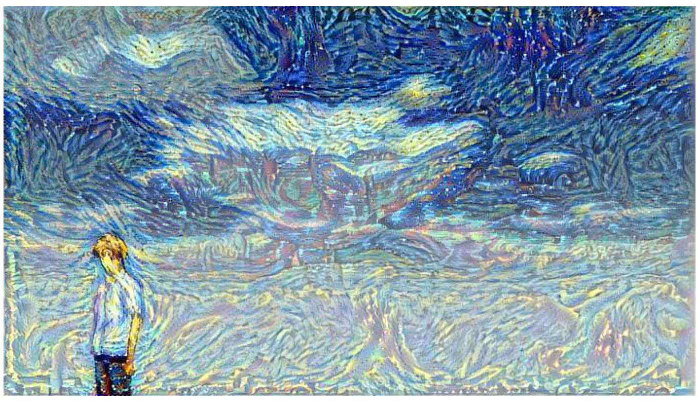


Figure 17 Transferred Frame 24

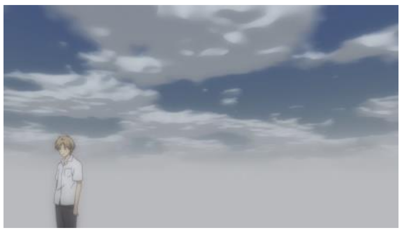


Figure 18 Original Frame 100

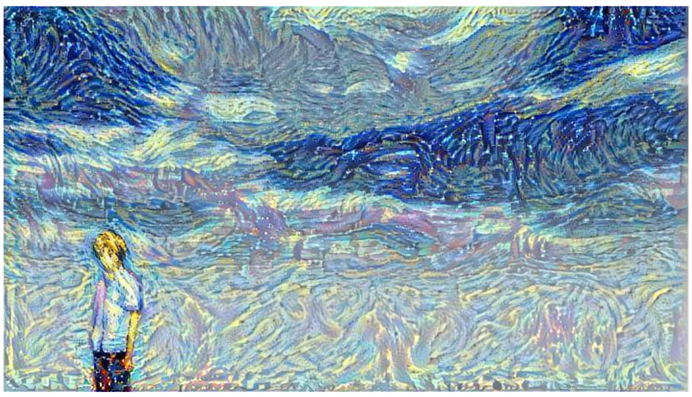


Figure 19 Transferred Frame 100

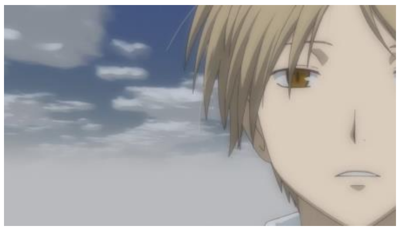


Figure 20 Original Frame 455

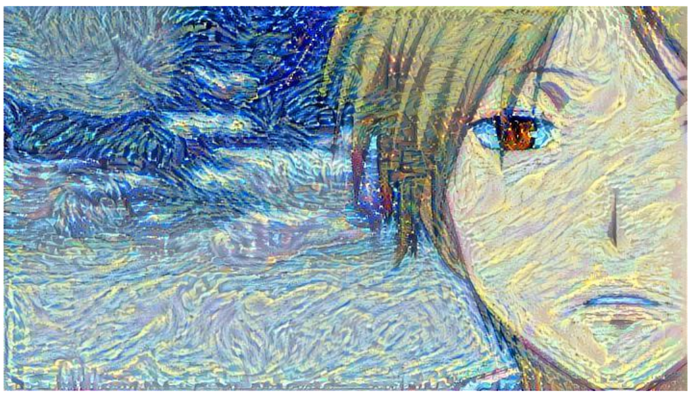


Figure 21 Transferred Frame 455

Examples of Transferred Frames with 5 Iterations

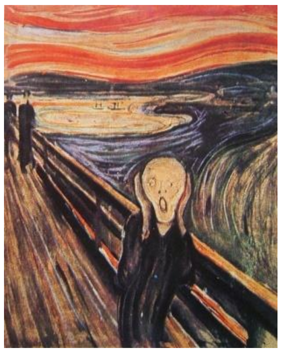


Figure 22 Style Reference Image

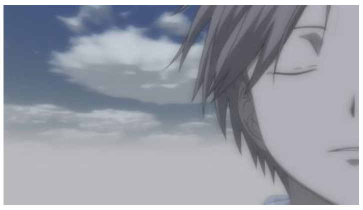


Figure 23 Original Frame 500

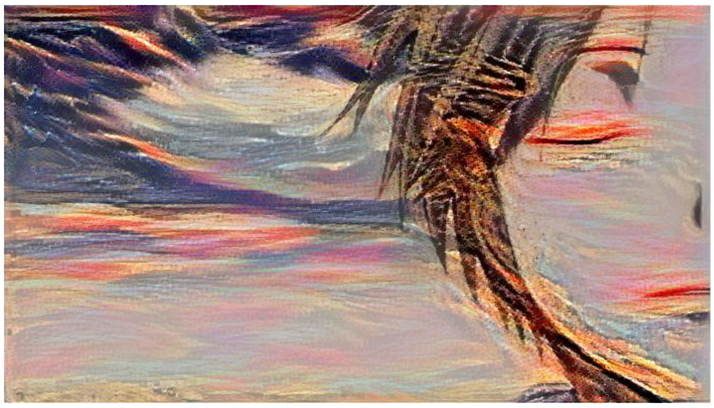


Figure 24 Transferred Frame 500

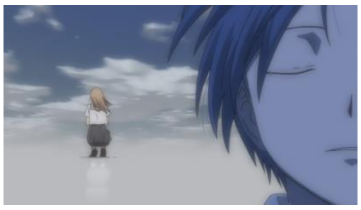


Figure 25 Original Frame 600

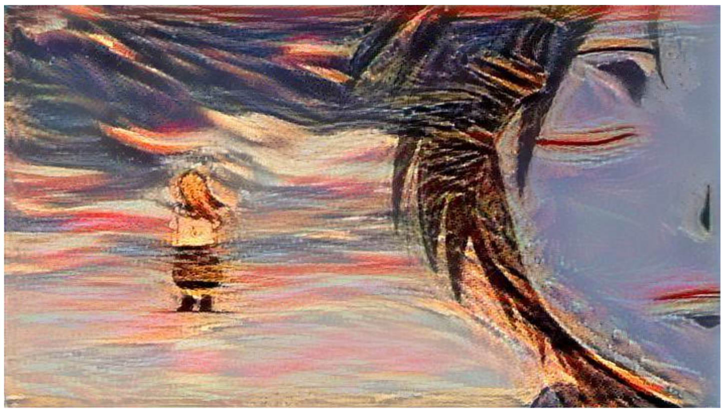


Figure 26 Transferred Frame 600

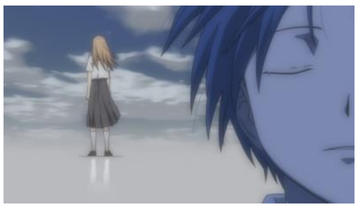


Figure 27 Original Frame 700

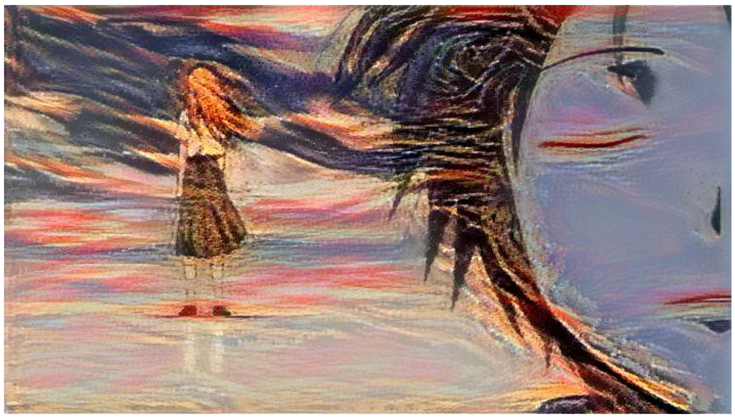


Figure 28 Transferred Frame 700

Examples of Transferred Frames with 50 Iterations

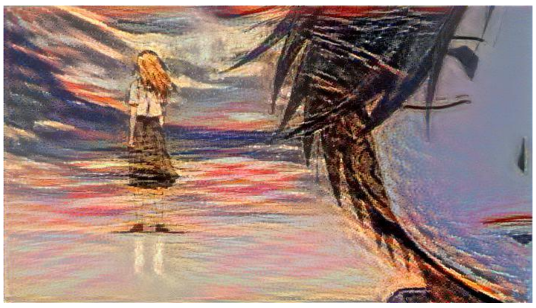


Figure 29 50 iterations

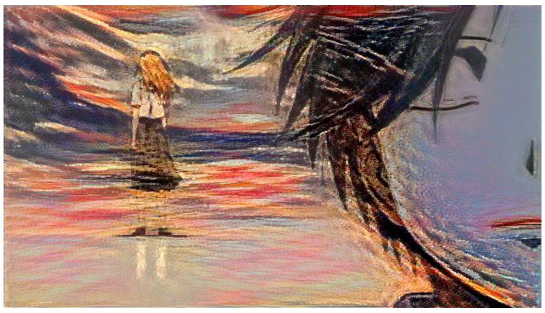


Figure 30 100 iterations

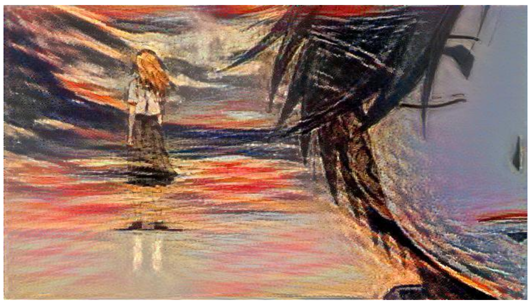


Figure 31 200 iterations

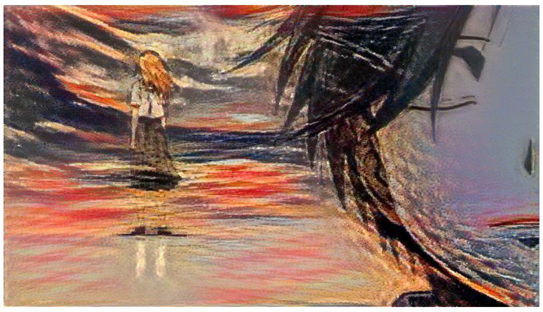


Figure 32 300 iterations

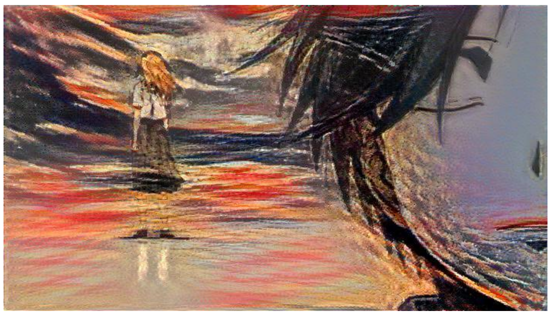


Figure 33 400 iterations

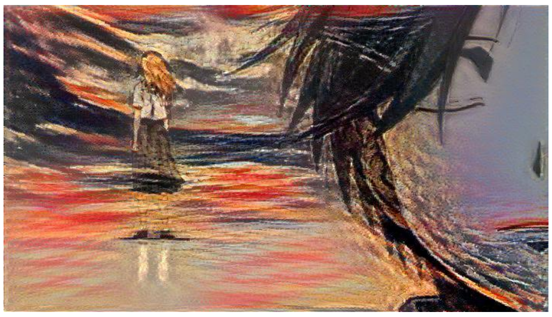


Figure 34 500 iterations

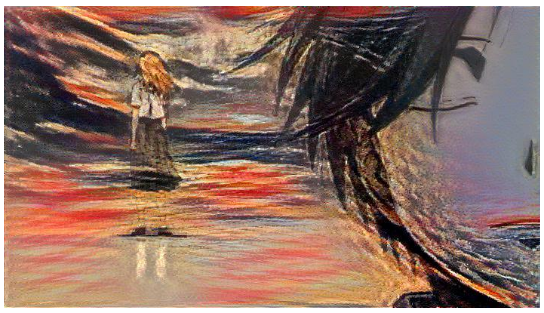


Figure 35 600 iterations

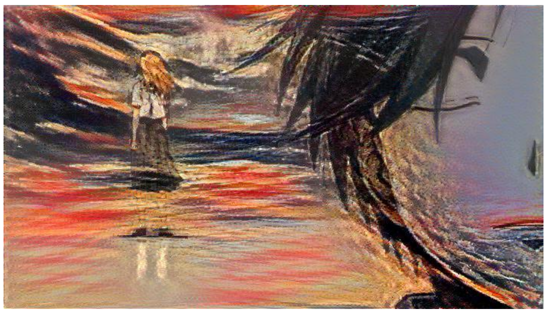
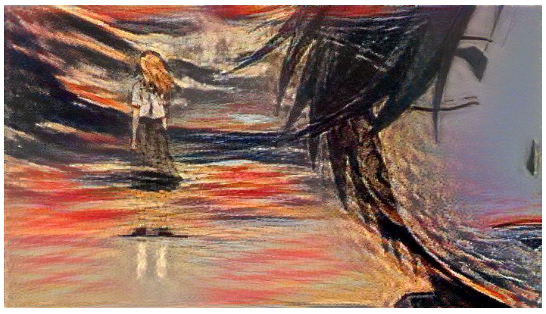


Figure 36 700 iterations



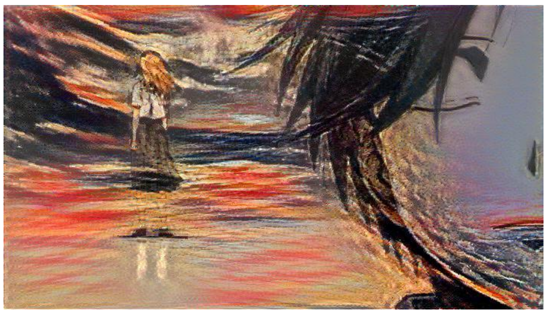
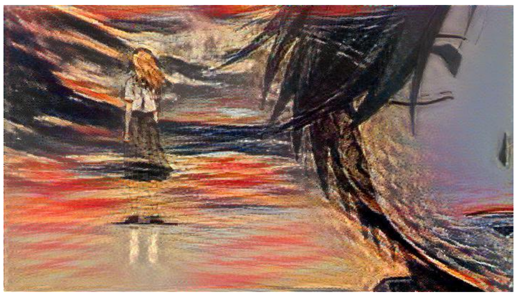


Figure 37 800 iterations

Figure 38 900 iterations



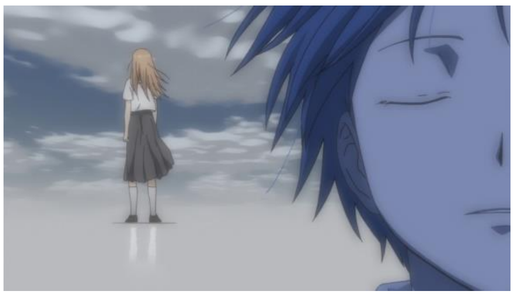


Figure 39 100 iterations

Figure 40 Original Frame

Examples of Different Iterations for a Frame