Portrait Style Transfer Using Neural Network

ZHANG Yuechen, XING Jinbo

December 7, 2019

Abstract

Headshot portraits are popular subjects in photography. But to transfer them into a compelling style such as painting is still a big challenge for researchers. Algorithms that automate or assist the stylization of photos do not perform well on headshots because of deforming facial structures. Besides, the existing specific headshot style transfer algorithms cannot balance efficiency and effect well and have different degrees of drawbacks. In this report, we present an enhanced version for portrait style transfer based on the work of Selim et al. This can allow one to do the fast stylization for headshot photos with fewer flaws such as color leakage, low-resolution, and mis-alignment artifacts. The core of our approach is a combination of techniques to robustly transfer the style of one example headshot into a new one with our enhancement. The results of this project show that we can successfully handle diverse styles by different artists.

Keywords: Headshot portraits, style transfer, face alignment, feature decoder, neural network, texture synthesis

1 Introduction

Headshot portraits are popular subjects in photography. In addition to pursuing reality, sometimes, professional photographers also spend a lot of time and pay effort to edit headshot digital photos to achieve some special styles. Different styles give people different impressions. Some styles are simple, such as high-contrast and black-and-white photos. However, the editing process is repeated and requires some advanced skills for processing facial components such a s eyes, mouth, etc. It's hard to satisfy people who are sensitive to the balance of color in portrait photos. Besides, some styles like painting are more complex and harder to apply. Generic painting transfer techniques often deform facial structures that cannot create satisfying artworks. People even want high-resolution and high-quality stylized headshot portrait photos look like some famous artworks, such as Van Gogh self-portraits and Mona Liza.

Portrait style transfer is not a blank field and becomes popular nowadays. In the beginning, non-photo-realistic rendering [Kyprianidis et al., 2012] comes out that attract more researchers to get involved in this field. Current techniques can be classified into two main categories: neural-based and texture-based. Neural-based techniques [Gatys et al., 2015] present the most recent generic style transfer technique. However, it is designed for generic images and hence when applied to headshot portraits, it often deforms facial structures. Selim et al. proposed a method to solve this problem by combining VGG features, face alignment, and gain map [Selim et al., 2016]. However, it has problem of artifacts caused by alignment. And all neural-based methods have some common drawbacks. The first one is color leaky which caused by the mis-matching of the style representation. The second one is the loss of high-frequency details because of the feature extraction of convolutional neural networks. The third one is low quality caused by the limitation of computation power in neural network. Texture-based method [Fišer et al., 2017] doesn't suffer from image warping artifacts but has extremely low efficiency for the generation of guide channels involved in computation and optimization. These observations motivate us to propose a method for balancing time and effect.

We present an enhanced version for portrait style transfer based on the work of Selim et al. Instead of optimizing the output image initialized as the input portrait, we introduce a decoder network to decode VGG features of the gain map as the initialization to accelerate the stylizing process [Selim et al., 2016]. A histogram match loss is introduced for matching style histograms during the back propagation. Besides, a new alignment network is used for facial alignment to avoid artifacts caused by SIFT flow, which is applied in the Selim's work[Liu et al., 2010]. With this fast stylization method, without guide channel producing, we can apply a texture synthesis method to transfer high-frequency texture from high-quality style images [Texler et al., 2019]. Besides, a semantic matting mask for portrait is used, which could help to distinguish regions in the face and background [Shen et al., 2016], to reduce the color leaking problem.

Our contribution could be summarized as follows:

- We propose a new face alignment method for portrait style transfer with a low failure rate.
- We show the effectiveness of a decoder network to initialize the transfer output. Thus to achieve a fast portrait style transfer with preservation of features on local correspondent regions.
- With local feature transfer, it makes the texture synthesis method possible to generate high-quality stylized portrait images.

2 Related Works

2.1 Neural-based Portrait Style Transfer

As a topic on non-photo-realistic rendering (NPR) [Kyprianidis et al., 2012], there are several methods and applications in image style transfer. Shih et al. present a style transfer technique on realistic portraits using a gain map on style features to match the local spatial distribution of light after an alignment step [Shih et al., 2014]. However, the portrait style can only match for natural style images in this approach, it fails when using painting style as reference. Besides, according to the work of Gatys et al., a neural-based image style could be extracted by a pre-trained deep convolutional network [Gatys et al., 2016]. In Gatys' work, a neural-based system for image style transfer using deep classification network is put forward, which evaluates distances in content and style representations of VGG16 features to optimize the transfer output [Simonyan and Zisserman, 2014; Gatys et al., 2016]. This approach can get stylized results for arbitrary inputs and references, but spatial features could not fit well for portrait transfer.

Selim et al. put forward their work as a combination of feature gain map and neural style transfer [Selim et al., 2016]. With the local feature refinement step using feature gain map, it enables paint style portrait images translating local features to regions with a high perceptual similarity of the refined input image. The limitation of this work is the long processing time and difficulty in face alignment, which will be discussed in the next subsection. Despite optimization on output images, several works are trying to move the optimization step on features by training a decoder on stylized features for output image generation [Lu et al., 2017].

There are learning-based methods for style transfer using perceptual loss that could get a stylized result in a fast process [Johnson et al., 2016]. In addition, some generative adversarial network structures for portrait style transfer can get a real-time result with high-quality [Futschik et al., 2019]. However, these approaches aim at training a network for a certain style, so they need more time and data to train for a new style, so these methods cannot handle arbitrary style images. A style-based generator architecture solved the problem for translation on realistic portrait photos [Karras et al., 2019]. However, it is still hard to obtain a large-scale stylized portrait dataset. Whether non-realistic features in these images could be learned is also questionable.

2.2 Face Alignment

As a computer vision technology, face alignment identifies the geometric structure of faces in digital images. Based on translation, scale, and rotation, it attempts to obtain a canonical alignment of the face. A lot of face alignment methods cannot align features precisely, especially for face contours [Zhang et al., 2016]. They use multi-task cascaded convolutional networks to detect eyes, nose, and mouth as five facial feature points, followed with affine transformations. Selim's work [Selim et al., 2016] detects facial landmarks in two images which are standard image A and image B to be aligned using Saragih's method [Saragih et al., 2009]. It can generate 66 points marking eyes, eyebrows, nose, mouth and lower facial contours. With these points, they align B to A by image morphing [Ucicr, 1992] followed by SIFT-flow [Liu et al., 2008]. Besides, Shih [Shih et al., 2014] also adds rough alignment [Joshi et al., 2010] and applies improved sift-flow [Liu et al., 2010]. Although they can get better alignment results than the previous methods, there still exists a flaw that they can not align higher facial contours and it looks weird.

2.3 Stylization with Texture Synthesis

Compared to traditional neural-based stylization, stylization with texture synthesis can preserve texture details to a large extent. It's a great improvement for style transfer nowadays for its high-resolution and high-quality results. Fišer introduces an example-based stylization approach that they use pixel-level or patch-based synthesis to get texture preserved stylization [Fišer et al., 2016, 2017]. Their main idea is, given the guidance image, for each patch or pixel, they try to find the most similar patch or pixel with minimum loss in the high-resolution style image to synthesize the final stylized image. Although they can get better and more realistic results with more details, there is a big problem in efficiency for its optimization algorithm. More than one minute is needed to stylize one image. Sỳkora et al. [Sỳkora et al., 2019] improves Fišer's algorithm and gets two orders faster competitive results to make real-time performance possible. Texler et al. combine neural methods with patch-based synthesis to achieve compelling stylization with high-quality for the high-resolution imagery [Texler et al., 2019].

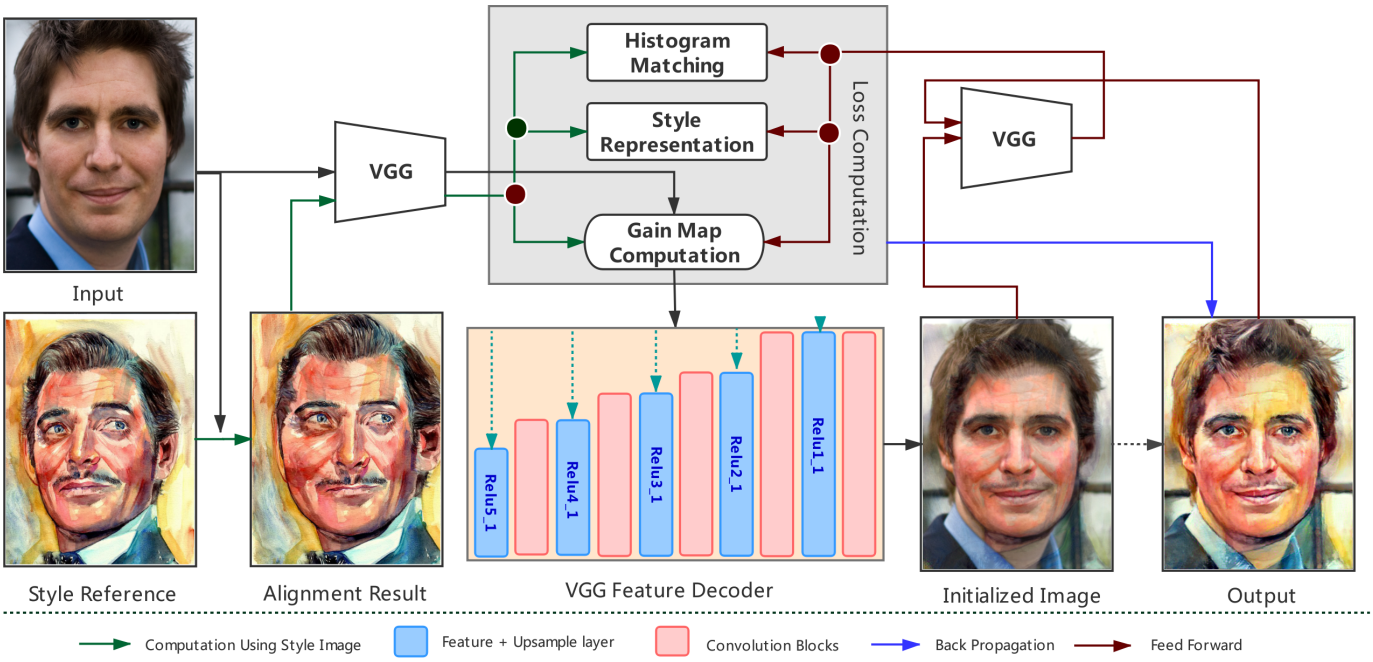


Figure 1: Overview of our portrait style transfer

3 Methods

The overall pipeline in our portrait style transfer is shown in figure 1. Comparing with the work of Selim et al., we add a VGG feature decoder to initialize the image input with gain map features [Selim et al., 2016; Lu et al., 2017]. We also add a histogram match for each extracted feature map. With an enhanced landmark-based alignment network [Huber et al., 2016], we can avoid artifacts caused by SIFT-flow to align the face outline.

3.1 Face Alignment

The overall pipeline of face alignment is shown in figure 2. Some methods we use are off-the-shelf algorithms and we improve them in amount of details according to our project.

3.1.1 Face And Landmarks Detection

The face detector we use is made by using the classic Histogram of Oriented Gradients (HOG) feature combined with a linear classifier [Dalal and Triggs, 2005], an image pyramid, and a sliding window detection scheme. This detector is general and can detect many types of semi-rigid objects in addition to human faces. This method has been encapsulated into function 'get_frontal_face_detector()'. The facial landmarks detection use dlib's implementation of Kazemi and Sullivan's method [Kazemi and Sullivan, 2014]. Compared to Selim's headshot portrait style transfer [Selim et al., 2016] that uses 66 facial feature points in alignment, we use totally 89 feature points which contains original 81 feature points in face including entire facial contours, eyes, eyebrows, nose, mouth, and 8 points which are corners of the image and half way points between those corners in the image to achieve a more accurate result. By doing so, we can solve most of the alignment failure problem in Selim's alignment. The process is shown in figure 2(1) and we can get facial landmarks shown in figure 2(b) and 2(e).

3.1.2 Face Cropping

In order to get better result in face morphing, we have to crop the face from original style images according to position of face in input images. The process result is shown in figure 2(3) and the details are as follows. We first detect face and landmarks in the input image and get the distance from out-most features points to image border in four directions. Then we detect face and landmarks in the style image and crop it according to the distance we just get. Finally, we need to correct the facial landmark coordinates because of the alternation of input image size. After cropping in this way, we can get two roughly aligned images for further processing in morphing, which are figure 2(b) and 2(f).

3.1.3 Face Morphing And SIFT-Flow

Morphing is an important step in face alignment because it greatly affects the result in style transfer. The basic idea is from Ucicr's work [Ucicr, 1992]. It involves two parts for cropped style images: delaunay triangulation and affine warping. The result of delaunay triangulation is a list of

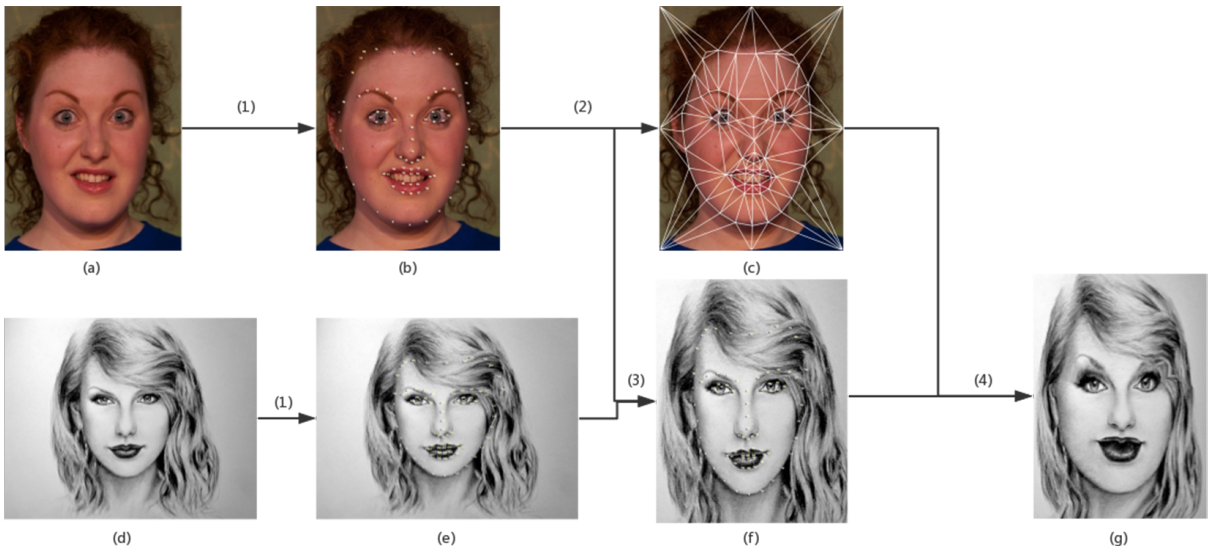


Figure 2: The overview of face alignment. (a) input image, (b) landmarks of input image, (c) delaunay triangulation result, (d) style image, (e) landmarks of style image, (f) cropping result, (g) final face alignment result, (1) facial landmarks detection, (2) delaunay triangulation, (3) cropping style image according to position of the portrait in input image, (4) affine warping

triangles represented by coordinates, which are shown in figure 2(c). Then we can find the indices of these points. Using these indices, we can apply affine transform from triangles in the style image to corresponding triangles in the input image. Then we warp triangles from the same direction to get warped style images. This process shown in figure 2(4) aligns facial landmarks [Selim et al., 2016] and gets the result in figure 2(g). However, we find it has good enough results because of extra facial landmarks. SIFT-Flow [Liu et al., 2010] is a method to align an image to its nearest neighbors in a large image corpus containing a variety of scenes by dense correspondence. We try to apply it to get more precise alignment, however, it fails in amounts of images and generates distorted results sometimes, which can be shown in figure 6(d).

3.2 Portrait Style Transfer

3.2.1 Feature Gain Map

Based on Selim's work, VGG features $F_l[\cdot]$ in layer l of input image I and style reference R could form a modified feature map $F[M]$ defined by following equations [Selim et al., 2016]:

$$F_l[M] = F_l[I] \times G_{lc} \tag{1}$$

$$G_{lc} = max(min(G_l, gmax), gmin) \tag{2}$$

$$G_l = \frac{F_l[R]}{F_l[I] + \varepsilon} \tag{3}$$

G_{lc} is the clamped gain map feature G_l in layer l. In Eq. 2, $(gmin, gmax)$ is the range of feature gains from the style reference image. $F[M] \approx F[R]$ without any clamp operation. ε is a small number to avoid division by zero. Note that all multiplication and division operations are element-wise. With this feature modification, we can then use it as the target of the content loss instead of $F[I]$ in Gatys' work [Gatys et al., 2016].

3.2.2 Loss Functions

Applying the modified feature map $F[M]$, the loss function L of our style transfer is shown as follows:

$$L = \alpha_c l_c + \alpha_s l_s + \alpha_h l_h \tag{4}$$

In Eq. 4, l_c is defined as the modified content loss, l_s as the style loss, and l_h as the histogram matching loss. $\alpha_{(\cdot)}$ are corresponding weights on these loss terms.

$$l_c = \sum_l^L (\alpha_l \frac{1}{2N_l D_l} \sum_{ij} (F_l[O] - F_l[M])_{ij}^2) \tag{5}$$

Content loss. In Eq. 5, L is the layer subset of all l in the feature extractor network. For each layer, there is a weight α_l to balance the contributions of different layers. N_l is the channel number of the feature extracted by the activation layer l. D_l is the number of elements in each channel (width $\times$ height). This is a mean square distance of $F[O]$ and $F[M]$, to constrain the content distortion during the optimization.

$$l_s = \sum_l^L (\beta_l \frac{1}{2N_l^2} \sum_{ij} (gram(F_l[O]) - gram(F_l[M]))_{ij}^2) \tag{6}$$

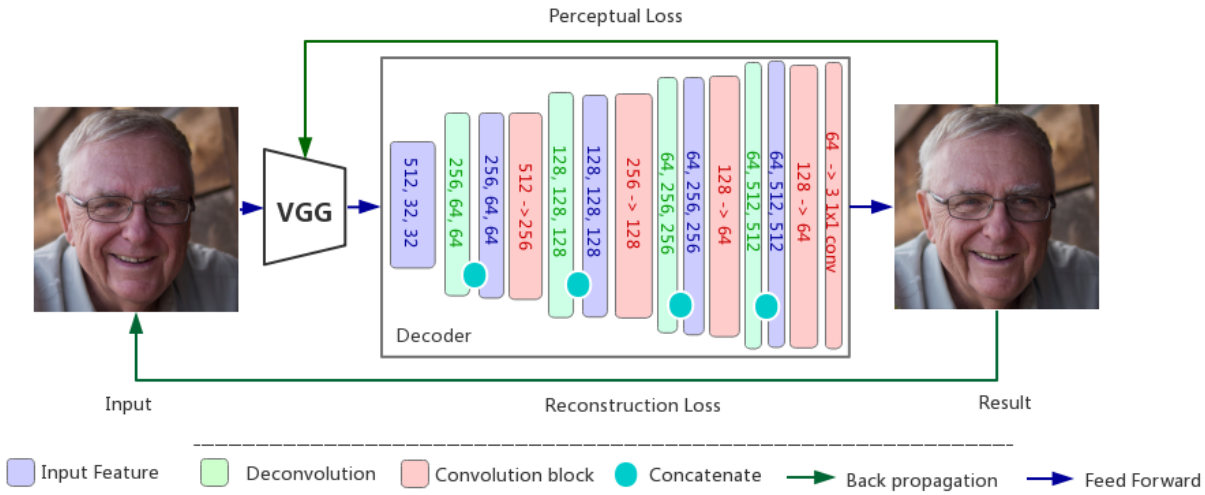


Figure 3: Network architecture of our feature decoder. Same as the layer set L we choose in the style transfer, input features are extracted from VGG19's $relu_{l\_1}$, $l \in \{1, 2, 3, 4, 5\}$. In each convolution block except for the last convolution layer, it contains 2 convolution layers with 3×3 kernel size, instance normalization and a relu activation is applied.

$$gram(F) = F_{(N,D)}F^T_{(N,D)}, D = WH \tag{7}$$

Style loss. Style loss in our work is similar to Gatys' work [Gatys et al., 2016]. Similar to content loss, the weight parameter β_l balances contribution of different layers. Style loss is the mean square distance of gram matrices of $F[O]$ and $F[M]$, which are style representations of input and style reference. Gram matrix is computed by reshaping the feature from (N, W, H) to (N, D), followed by doing the inner product of elements for every two channels. It has a size of $N \times N$. Gram matrix represents the channel-wise style correspondence, so the style loss tries to optimize the feature of style.

$$l_h = \sum_l^L (\frac{1}{2N_l D_l} \sum_{ij} (F_l[O] - H(F_l[O], F_l[S]))^2_{ij}) \tag{8}$$

Histogram loss. Inspired by the effectiveness of the work of Risser et al., a histogram loss is added to maintain the feature map histogram [Risser et al., 2017]. In Eq. 8, $H(x, y)$ is the histogram matching function of x to y. Histogram matching processes channel-wised on VGG features. The histogram loss is the mean square loss between the output feature and the matched feature.

3.2.3 VGG Feature Decoder

Besides the traditional gain map approach, inspired by Lu's work [Lu et al., 2017], a decoder network is applied on VGG features to initialize the output image O. With the decoder, output with a gain map feature could be generated as the initialized output.

Network architecture. The decoder architecture is shown in Fig. 3. The feature F is extracted by VGG 19 network [Simonyan and Zisserman, 2014]. We apply the decoder part of the U-net architecture [Ronneberger et al., 2015]. For each feature group extracted from one activation layer l, we apply a de-convolution layer to upscale the feature size, as well as reduce the number of channels. A concatenation process is performed on the feature with the same size. A residual convolution block including two convolution layers is added after each concatenation. After the upsample-concatenation-convolution process for all input features, a convolution layer with 1×1 kernel size is applied to get an RGB-channel output O.

Loss function. The design of loss function for feature decoder aims to reconstruct the input image in the output. The decoder loss function L_e is shown in the following equations.

$$L_e = \gamma_r l_r + \gamma_p l_p \tag{9}$$

$$l_r = \frac{1}{D} \sum_{ij} (O - I)^2_{ij} \tag{10}$$

$$l_p = \sum_l^L (\alpha_l \frac{1}{2N_l D_l} \sum_{ij} |F_l[O] - F_l[I]|_{ij}) \tag{11}$$

In Eq. 9, $\gamma_\cdot$ is the weight parameter to balance two loss items. l_r is denoted as the reconstruction loss. Which is the mean square distance between input I and output O. D is denoted as the element (pixel) number in the input image. l_p is denoted as the perceptual loss. Proved by Johnson's work, the perceptual feature in the VGG network could perfectly represent the image [Johnson et al., 2016]. Thus, l_p is designed as the L1 loss of VGG perceptual features F in the layer set L. $N_l, D_l.\alpha_l$ have the same definition as it is defined in Eq. 5.

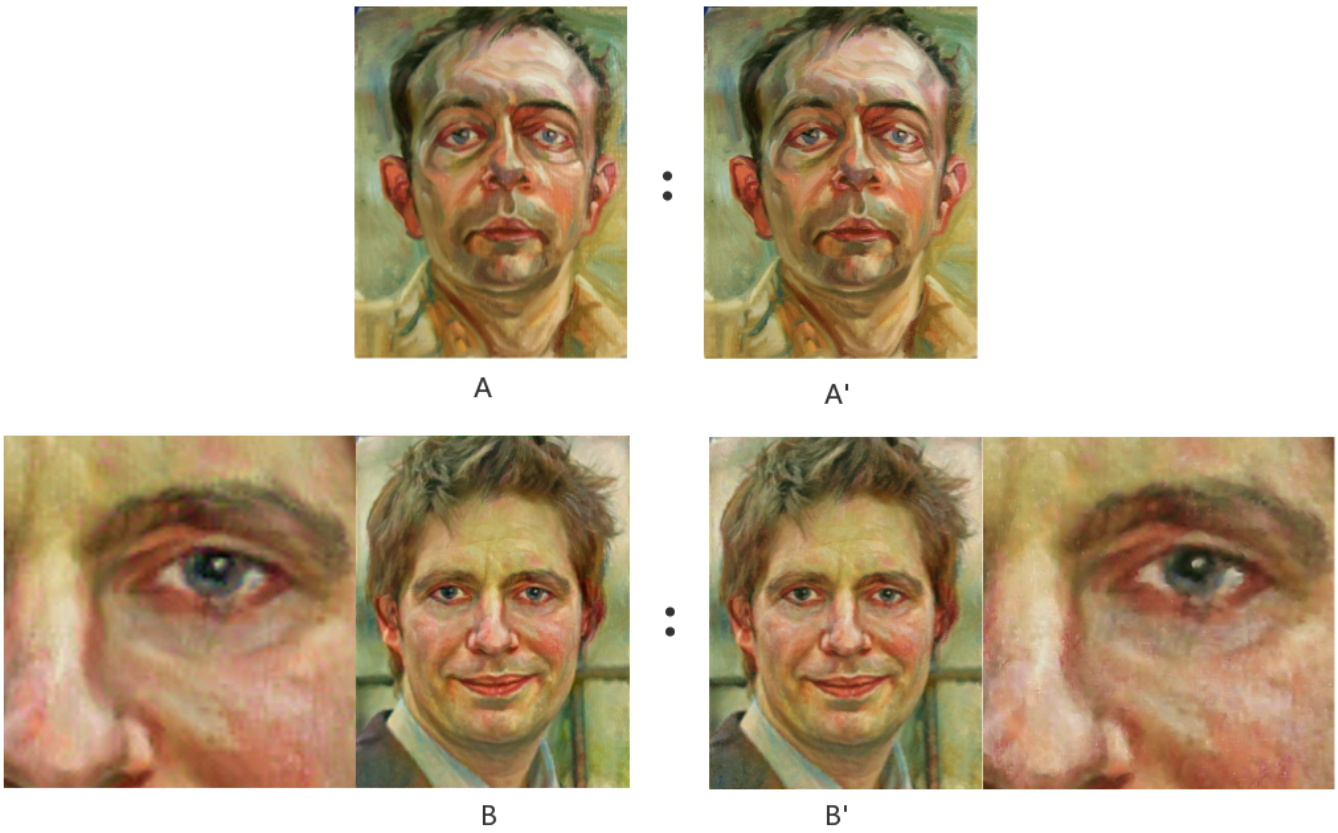


Figure 4: The concept of image analogies in guided texture synthesis: (A') high-resolution and high-quality style portrait, (A) blurred A', (B) neural-based A' style transfer result, (B') final result

3.3 Guided Texture Synthesis

Guided texture synthesis we use here is mainly from Fišer's method [Fišer et al., 2016]. They propose this method for the illumination-guided example-based stylization of 3D renderings. We just simplify and change it a little bit for our purpose. We change it to texture-guided (color-guided) example-based texture synthesis of low-resolution images. The result is that given high-resolution style portrait imagery, we can generate stylized headshot portrait images with same quality based on texture guidance as the result of neural-based style transfer.

Images in this process are shown in Figure 4. Given a high-resolution style portrait image A', we blur it to image A by reducing followed by enlarging. The quality of image A has been obviously reduced with unchanged resolution. Then we get the final stylized image B from the neural-based style transfer result above. By applying Fišer's texture synthesis method [Fišer et al., 2016] to image B, We can generate a high-resolution and high-quality stylized image B' retaining the original high-frequency details. The main idea of their algorithm is to find the best matching patch in A' to substitute that in B for generating whole image B' according to the texture (color) guidance from A to A'. The optimization is based on Eq. 12, which is originally proposed by Hertzmann et al[Hertzmann et al., 2001].

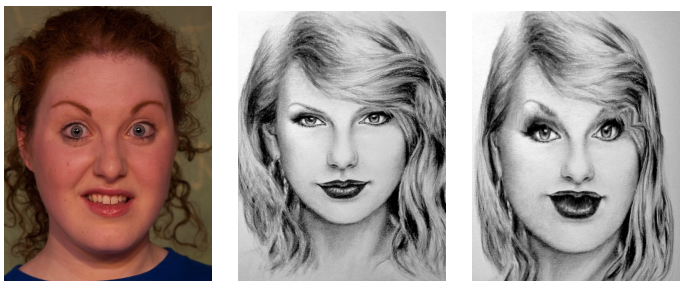
$$E(\mathbf{A}, \mathbf{B}, p, q, \mu) = \|A'(p) - B'(q)\|^2 + \mu \|A(p) - B(q)\|^2 \tag{12}$$

For each patch $q \in \mathrm{B}'$, a best matching patch p is found in the source A' such that Eq. 12 is minimized. Here $\mathbf{A} = \{A, A'\}$, $\mathbf{B} = \{B, B'\}$, μ is a weight that controls the balance.

4 Results

4.1 Content-Feature Trade-off

Since the network and feature has differences among images, the balance parameter of content and style loss has an important effect on outputs. An instance of this balance is shown in Fig. 5. We set a proper $alpha_s$ for all experiments to get a stylized result (5(e)) without style outmatching or content distortion (5(d) and 5(f)).



(a) Input (b) Style Reference (c) Alignment

(d) $\alpha_s = 5 \times 10^{-7}$ (e) $\alpha_s = 5 \times 10^{-6}$ (f) $\alpha_s = 5 \times 10^{-5}$

Figure 5: An example of the content-feature trade-off. With different weight parameters, the sketch style and original content vary differently, from content dominant (color preserved) to style dominant (content distort) with the parameter increasing.

4.2 Comparisons

First, we compare our face alignment result with the previous method of Selim et al. [Selim et al., 2016]. The comparison is shown in figure 6. We use face cropping based on the position of the portrait in the input image to do preliminary alignment. In figure 6(c), we can see the flaw in higher facial contour because 68 facial landmarks do not contain enough feature points. The drawback of SIFT-Flow has been explained for its distorted results. Figure 6(e) can show better face alignment results with 81 facial landmarks by our improvement. Our method can align almost all facial features, including facial contours, eyebrows, eyes, nose, and mouth. Because of the better alignment results and gain map, we can get better style transfer results with reduced errors, especially in facial matching regions.

Another important comparison with different methods is the headshot portrait style transfer, which can be found on the last page. In Fig. 9, we compare our method with two classical methods, Gatys et al. and *DeepArt* web service [Gatys et al., 2016]. Three different styles with two input images are used in this example. The Gatys' work can not hold the color balance on the output due to the misalignment on spatial features. Meanwhile, *DeepArt* can balance the color on a human face, but it can not transfer localized features in the style reference. For example, the bear in the second row disappears in the *DeepArt* result. Our approach better captures the texture of the styles than Gatys et. al. and *DeepArt*. In addition, it matches facial features in corresponding regions over both approaches. Last but not least, all three methods cannot do well on transforming the realistic style images (the 3r̂d and 6t̂h row in Fig. 9), which could be handled by the work of Shih et.al [Shih et al., 2014].

In order to solve the color leakage problem, we try to do portrait matting to reduce the influence caused by background. There exist some related methods, especially for portraits, such as Shen's CNN based method[Shen et al., 2016]. For convenience, the free APIs we adopted to remove the background of portraits are provided by the website "remove.bg". It is totally automatic and almost perfect. The matting result can be shown in figure 7(d). However, there still exist some problems when removing the background in this way. After applying face morphing on style images, it may be failed to remove background in those images. The comparison is shown in figure 7. We can see color leakage of yellow in normal style transfer result in figure 7(c), this can be reduced after portrait matting in figure 7(e).

4.3 Application on Portrait Video Stylizing

Based on the style transfer technique we presented, we can extend the portrait stylizing to video. Jamriška et al. presented a fast video stylizing method by texture synthesis with a small number of stylized key-frames [Jamriška et al., 2019]. However, examples in this paper are using hand painting as a style reference, which is hard for non-expert users. As figure 8 shows, this method (b) could be combined with our portrait transfer (a). We select several keyframes to generate stylized frames i ii and iv. Then, applying the method of Jamriška et.al, stylization result for non-keyframes such as iii could be generated by texture synthesis without any style transfer optimizations. Thus, we can generate a stylized portrait video with different poses in a fast way, about 3 second per frame on a CPU device (to compare with, the method of Fišer et al. for portrait stylizing has a long processing time for about 200 seconds per frame on a GPU device because time is consumed by the guide channels' construction [Fišer et al., 2017]).

5 Discussion

5.1 Implementation Details

Dataset and image pre-processing. For our style transfer results, we use a Flickr-image dataset as input images, online searched style portraits as a set of style references. For the training part of the VGG feature decoder, we use stylized human portraits using the Gatys' method [Gatys et al., 2016]. In this stylization, input images are from the FFHQ dataset, which is generated in the styleGAN work [Karras et al., 2019]. Style images are from Kaggle's style

Figure 6: Face alignment from style image to input image. (a) input image, (b) style image, (c) face alignment using 68 facial landmarks, (d) face alignment followed by SIFT-Flow, (e) our approach for alignment with 81 facial landmarks

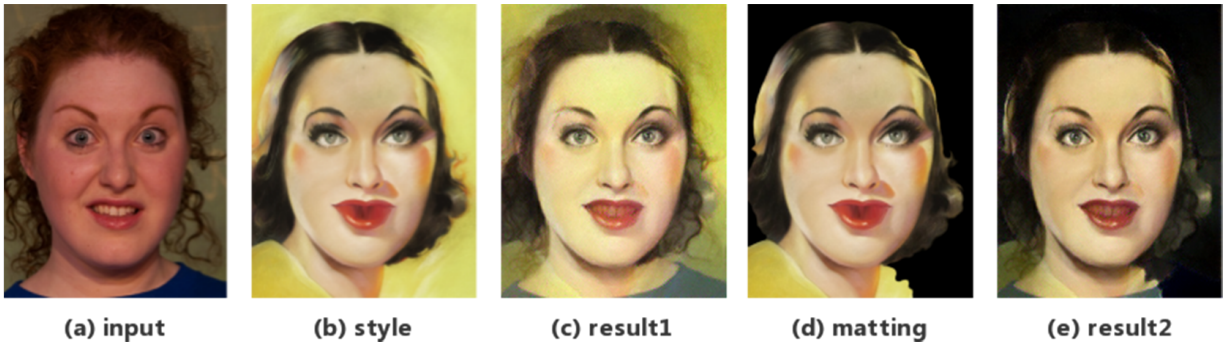


Figure 7: Comparison of matting or not.

image dataset. We randomly pick one input image and one style image, resize them to 512×512 and apply the style transfer method. In this way, we generated 2600 stylized portrait images. Actually, with experiments, we find that the decoder could be trained by arbitrary images (such as just using the style reference images as input).

Portrait style transfer. In the feature gain map part, we choose $L = \{relu_{l\_1}\}, l = 1, 2, 3, 4, 5$ in VGG19's pytorch implementation. For histogram loss, we only select $relu_{4\_1}$. We add normalization and de-normalization layers on the input and output to fit the network's requirement. $gmin = 0.3, gmax = 10, \varepsilon = 10^{-6}$ are set in this step empirically. In the optimization loss function, we balance loss terms by setting parameters $\alpha_c = 1, \alpha_s = 5 \times 10^{-6}, \alpha_h = 5$. $\alpha_l = 1$ for $l = 1, 2, 5$, 0.8 for $l = 3, 4$, $\beta_l = \alpha_l$ for each l. With the optimizer applying LBFGS algorithm and learning rate $lr = 1$, we can attain the transferred image within 20 iterations (20 closures per iteration). If histogram loss is applied, the time consume is 43 seconds in average per image with size of 256×336. With out histogram matching, the time consuming could achieve 16 seconds per image, which is much quicker than Selim's optimizer settings [Selim et al., 2016]. All experiments are performed on a single GeForce GTX1080Ti device.

Time issue for face alignment and texture preserving enhancements. For a single image in face alignment, we load 2 face landmark predictor pre-trained models and 1 face detector pre-trained model. The average time for face alignment is about 4 seconds including loading models on a single GeForce GTX1080Ti device. So, it will be shorter when processing amounts of data for loading all models once. Time consuming is disappointing in texture preserving enhancements because of lots of computations on optimization. The average time of texture synthesis for an image is about 12 hours when processing and producing a high-resolution image (807×1059). This process is performed on a Google Colab CPU device.

5.2 Limitation and Future Works

Although our project works quite well for many styles of portraits and input headshot portraits, some limitations must be taken into account and it needs our future work to solve them.

In Figure 1, we show the overflow of the portrait style transfer. It still needs parameter tuning before the process to generate a good visual result. It is tedious and can be improved by parameter auto-tuning inspired by Risser [Risser et al., 2017].

Face alignment is an important part of our project for its usage in the gain map of style transfer. Because some portraits are too artistic and abstract to be detected, there-

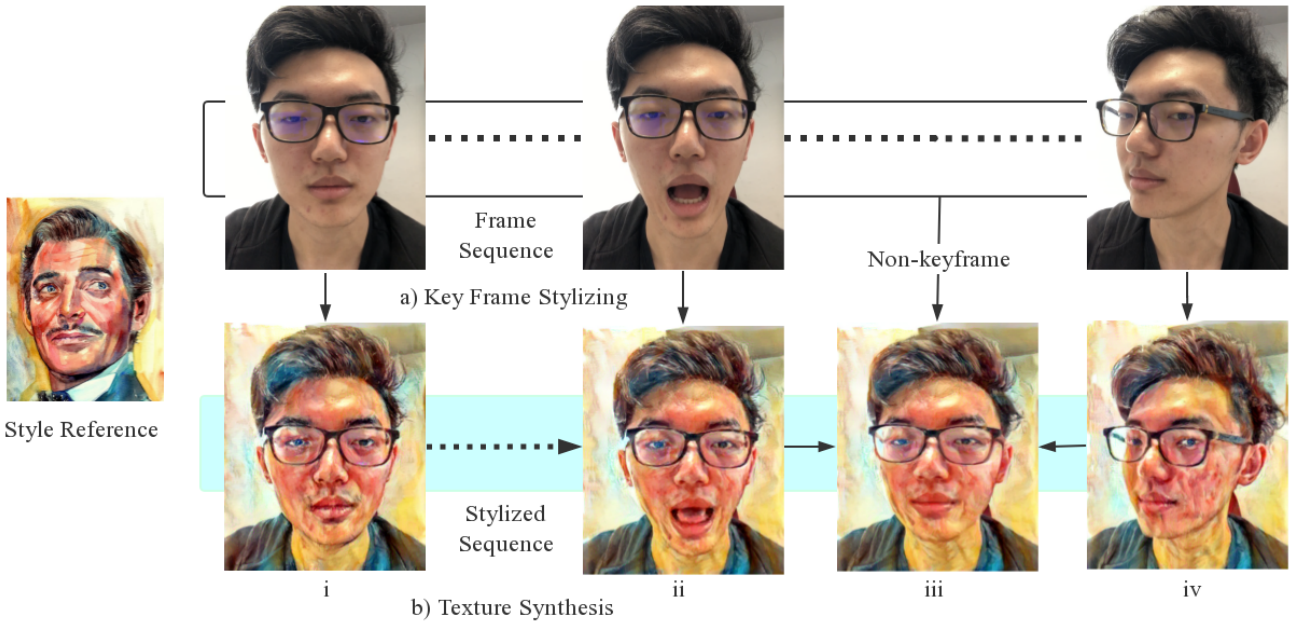


Figure 8: The pipeline of our application on portrait video style transfer.

fore, there exist some misalignment cases, some artifacts and wired results can be caused. It is not good because humans are sensitive to these kinds of facial abnormal.

In order to achieve the high-quality texture level stylization result, we apply StyLit algorithm [Fišer et al., 2016] to our neural-based style transfer result. It needs amounts of computation of optimization in the different iterations for multi-scales. The higher resolution and quality we want to transfer to, the more time it takes. In future work, we may optimize this method, especially for our purpose.

6 Conclusion

There is still a long way for discovering a state of art solution for portrait stylization. At the present stage, we try to make some enhancement on style transfer for arbitrary style references, such as new alignment method and feature decoder. Besides, we discuss the pros and cons of different methods and parameter settings. With our work, we can further extend the research to editing feature directly on a deep neural network by showing results with an inverting feature decoder. Also, it is possible to get a better result with less time cost with our design of arbitrary style transfer.

7 Acknowledgment

This project is one of the 2019 ELITE final year projects hold by the faculty of Engineering (course code: ESTR4998), the Chinese University of Hong Kong. Thanks to Professor WONG Tien-Tsin for his supervision and advice in this semester.

Here is the **work division of this project**:

- **XING Jinbo:** Implementation of face alignment (including landmark detection, cropping and morphing). Testing for guided texture synthesis method. Report: Abstract, 2.2, 2.3, 3.1, 3.3, 4.2, 5.2.
- **ZHANG Yuechen:** Implementation of portrait style transfer (including Gain Map, design of loss functions, VGG feature decoder). Report: 2.1, 3.2, 4.1, 4.3, 5.1, 6, 7.
- **Work together with equal work division:** Dataset preparation, testing, application on video stylizing, testing for other methods for comparing, matting. Report: Introduction.

References

N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. 2005.

J. Fišer, O. Jamriška, M. Lukáč, E. Shechtman, P. Asente, J. Lu, and D. Sỳkora. Stylit: illumination-guided example-based stylization of 3d renderings. *ACM Transactions on Graphics (TOG)*, 35(4):92, 2016.

J. Fišer, O. Jamriška, D. Simons, E. Shechtman, J. Lu, P. Asente, M. Lukáč, and D. Sỳkora. Example-based synthesis of stylized facial animations. *ACM Transactions on Graphics (TOG)*, 36(4):155, 2017.

D. Futschik, M. Chai, C. Cao, C. Ma, A. Stoliar, S. Korolev, S. Tulyakov, M. Kučera, and D. Sýkora. Real-time patch-based stylization of portraits using generative adversarial network. In *Proceedings of the 8th ACM/EG Expressive Symposium*, pages 33–42, 2019.

L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016.

A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 327–340. ACM, 2001.

P. Huber, G. Hu, R. Tena, P. Mortazavian, P. Koppen, W. J. Christmas, M. Ratsch, and J. Kittler. A multiresolution 3d morphable face model and fitting framework. In *Proceedings of the 11th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, 2016.

O. Jamriška, Š. Sochorová, O. Texler, M. Lukáč, J. Fišer, J. Lu, E. Shechtman, and D. Sỳkora. Stylizing video by example. *ACM Transactions on Graphics (TOG)*, 38(4):107, 2019.

J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.

N. Joshi, W. Matusik, E. H. Adelson, and D. J. Kriegman. Personal photo enhancement using example images. *ACM Trans. Graph.*, 29(2):12–1, 2010.

T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

V. Kazemi and J. Sullivan. One millisecond face alignment with an ensemble of regression trees. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1867–1874, 2014.

J. E. Kyprianidis, J. Collomosse, T. Wang, and T. Isenberg. State of the" art": A taxonomy of artistic stylization techniques for images and video. *IEEE transactions on visualization and computer graphics*, 19(5):866–885, 2012.

C. Liu, J. Yuen, A. Torralba, J. Sivic, and W. T. Freeman. Sift flow: Dense correspondence across different scenes. In *European conference on computer vision*, pages 28–42. Springer, 2008.

C. Liu, J. Yuen, and A. Torralba. Sift flow: Dense correspondence across scenes and its applications. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):978–994, 2010.

M. Lu, H. Zhao, A. Yao, F. Xu, Y. Chen, and L. Zhang. Decoder network over lightweight reconstructed feature for fast semantic style transfer. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2469–2477, 2017.

E. Risser, P. Wilmot, and C. Barnes. Stable and controllable neural texture synthesis and style transfer using histogram losses. *arXiv preprint arXiv:1701.08893*, 2017.

O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

J. M. Saragih, S. Lucey, and J. F. Cohn. Face alignment through subspace constrained mean-shifts. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1034–1041. IEEE, 2009.

A. Selim, M. Elgharib, and L. Doyle. Painting style transfer for head portraits using convolutional neural networks. pages 129:1–129:18, 2016.

X. Shen, X. Tao, H. Gao, C. Zhou, and J. Jia. Deep automatic portrait matting. In *European Conference on Computer Vision*, pages 92–107. Springer, 2016.

Y. Shih, S. Paris, C. Barnes, W. T. Freeman, and F. Durand. Style transfer for headshot portraits. *ACM Transactions on Graphics (TOG)*, 33(4):148, 2014.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

D. Sỳkora, O. Jamriška, O. Texler, J. Fišer, M. Lukáč, J. Lu, and E. Shechtman. Styleblit: Fast example-based stylization with local guidance. In *Computer Graphics Forum*, volume 38, pages 83–91. Wiley Online Library, 2019.

O. Texler, J. Fišer, M. Lukáč, J. Lu, E. Shechtman, and D. Sỳkora. Enhancing neural style transfer using patch-based synthesis. 2019.

T. Ucicr. Feature-based image metamorphosis. *Computer graphics*, 26:2, 1992.

K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, Oct 2016.

Figure 9: Headshot portraits transfer for different input photographs (a). We examine multiple style images (b). For each input-style pair we show Gatys et al., *DeepArt* and our results