An Attempt of Facial Feature Style Transfer

Jindong Zhou
Shanghaitech University
No.393, Huaxia Middle Road, Pudong New Area, Shanghai, China
zhoujd@shanghaitech.edu.cn

Chengrui Zhang
Shanghaitech University
No.393, Huaxia Middle Road, Pudong New Area, Shanghai, China
zhangchr@shanghaitech.edu.cn

Di Wu
Shanghaitech University
No.393, Huaxia Middle Road, Pudong New Area, Shanghai, China
wudi@shanghaitech.edu.cn

Abstract

This paper presents a combination of facial feature mapping and deep image analogy strategies for face style transfer. In this method, a point-to-point match and deconvolution process is utilized to transform source image's style to the target. For the purpose of better transfer result, we take the advantages of landmarks detection and face morph algorithm. Our result achieves a splendid visual effect. Furthermore, an optimized method based on feature region matching is proposed to improve the performance of out model, and obtain about 10% - 15% acceleration in total.

1. Introduction

This work is to realize a facial features style transformation. The basic idea and method come from the 'style transfer' work. To be specific, the goal is to transfer one's face features in a real photo or selfie into another kind of facial style like film or game characters. Fig.1 shows an input pair.

It is similar to those style transform works, but we may try to tune the content as well. The content here means, for example, the size and the shape of the face features like eyes and lips. They should be tuned first in both input faces and then the style, which could be the color, luster and so on. We will focus on the details of the face features because the original style transfer work like [4] seems looking more at the structure or the whole style. More details will be discussed later.

Figure 1. Input pairs

In general, the whole transformation process will be divided into several steps. Based on the two input pictures, including a real photo and a target style picture, the first step is to locate the positions of facial features of characters in both pictures, which in fact is a technique called face landmarks detection. It has been perfected by many researchers, so we just take them into use. Finally we choose the methods provided by 'Dlib'(dlib.net).

Secondly, based on the landmarks, we will apply a face morphing or features mapping to those two pictures. That operation enable the positions of the facial features of both faces to locate at similar points in the pictures. After this pre-treatment, the style face will be twisted to match its facial features to the real one.

Next, the style of the whole face will be changed based on the algorithm raised by this work [4]. With the help of the matching in the former step, besides the general style, the detailed style transfer of the facial features will be more

Figure 2. A pair of results.

smoothly. So the results are visually better. Also, we modified the algorithm to accelerate the process.

Finally, the ideal output can be a picture which have the structure of the original face but the style of the other. As Fig.2 shows.

2. Related Work

The whole image style transfer has been studied by amount of researchers. For the first published work, 'Image Style Transfer Using Convolutional Neural Networks' in 2016 [2], they proposed an style transfer model to transform the style of a source image into the target image by using deep convolutional neural networks. In this model, it generates the transformed image by minimizing the lost function by two components—semantic information and texture information. The former one measures the similarity of the target image and the transformed image, which is quantified by calculating the 2-norm difference between the deep features of each image. The latter one utilizes covariance matrices, which can capture the correlations between deep features within an image, to obtain the texture similarity of those two images.

Based on this paper, several related works were proposed to improve the style transfer model. The Carlos work, in 2017, showed a targeted style transfer method using instance aware semantic segmentation. This method performed the object transformation by deep network-based image modification hybridized with semantic segmentation, trying to segment and stylize objects selected by users in the target image. For another work, 'Style Transfer for Headshot Portraits' [5], proposed an idea that morphing the source image's face size into target's, in order to get a better mapping between those two images.

Different with deep neural network method, in 2017, the Microsoft Research came up with a strategy called deep image analogy [4], which provide a fantastic model to transform visual attribute (such as color, tone, texture, and style) from the source image to the target, especially for the pair with similar semantic structure. For example, one image could be that of a painting or a sketch while the other is a photo of a real scene, and both depict the same type of scene. This model proposed deep image analogy based on 'image analogy' [3] and modified the Nearest-neighbor Field Search method. Their result is shown in Fig.3.

3. Data

Since our work is for a style transform between only two pictures, and the convolution neural network to extract the picture features, which in the structure of VGG19 is used directly from the former work, what we need to prepare is just the input pictures.

The real faces are from the dataset '300W' or the internet. The style faces of game characters are from the internet. Both of the input pictures need to be clipped or resized into the same resolution.

Also, after the face landmark detection, we need to add several points by hand to fix the algorithm of face mapping. This will be described in the next section.

4. Methods

As mentioned in the introduction, our work can be divided into two parts. The first is face landmark detection and feature mapping. The second is to transfer the style. What's more, we improve the Patch Matching Algorithm to suit our issue and reduce the run time.

4.1. Face Landmark Detection

Because we found there are some mismatches and twists of facial features in their work, we want to improve the effects. According to the work[5] mentioned in Section 2, we take their ideas. Before the style transfer, we can process the input faces. If the facial features of style face is changed first according to the related positions in the real face, the transfer operation can be more smooth and avoid deformation.

To realize this facial feature mapping, the first to do is to detect the landmarks in the faces. We have test several methods and templates, like the subspace constrained mean-shifts, the tool from Dlib and so on. Finally, we choose the method provided by Dlib with API for Python because of the convenience. It can mark 68 landmarks on the face, including eyebrows, eyes, mouth, nose and cheek contours. The recognition effects are satisfactory both in real faces and style faces, and the results will shown in Section 5.

4.2. Facial Feature Mapping

After the landmark detection, we will apply the facial feature mapping to the faces. That is to tune the face contour, feature sizes, positions in a monolithic way. Here we refer to a template on the internet(Face Morph Using OpenCV — C++ / Python), the main algorithm is Delaunay Triangulation and Affine Transform. Because we directly

Figure 3. Example of Visual Attribute Transfer

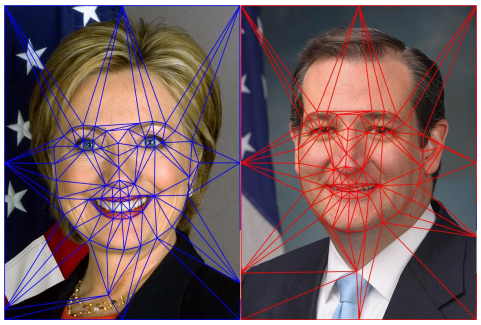

Figure 4. Delaunay triangulation example

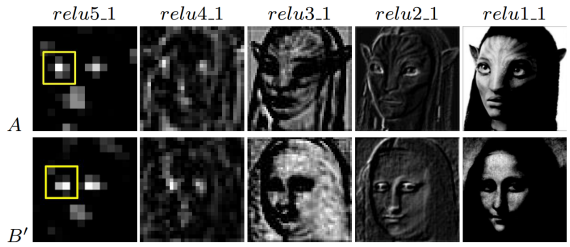

Figure 5. Input images' feature maps of each layer

use those algorithms in the library, so here detailed principles of them will not be described.

Given a set of points in a plane, a triangulation is to find the subdivision of the plane into triangles, with the points as vertices. In a Delaunay triangulation, triangles are chosen such that no point is inside the circumcircle of any triangle. Any discussion on Delaunay triangulation has to include Voronoi diagrams because they are mathematical dual. For those points, a Voronoi diagram partitions the space such that the boundary lines are equidistant from neighboring points. If we connect the points in neighboring Voronoi regions, a Delaunay triangulation is formed.

For a whole picture, only those 68 landmarks on the face are not enough for the Delaunay triangulation. So we need to add a few more points by hand to complete the nodes on the picture so that the triangles can cover the whole picture. Four corners and four middle points of the edges are added first. Then a point on the jaw and a point on the left ear are added. Also, if exists, points of shoulders will be added.

In all, there are 80 points on a single picture, and 149 triangles can be formed according to the algorithms with the same structure. Every triangle in both pictures contains similar part of the pace, for example in the Fig.4, one of the two triangles inside the left eye of both faces contains the left upper part and the other contains the right bottom part.

Finally, we can map the facial features. First of all, we set a morphing parameter between 0 and 1, whose function is to define the degree of the twist. For example, 0 or 1 means one face structure will be totally changed to the other face. And 0.5 means face features of both faces will tend to locate at the average position between them. And we will get the related landmark points positions in the output picture.

Next, choose a triangle in the image which you want to change and the corresponding triangle in the output image and calculate the affine transform that maps the three corners of the triangle in the image to the three corners of the output. The other image is the same. The Affine Transform algorithm has already been integrated in the OpenCV library. Since the function in the library is designed for a rectangle, the method simply imports a mask to solve the problem.

So finally, the output picture will be generated piece by piece in the shape of triangles. The changed results will be shown in Section 5.

What's more, here we use a trick. We set the parameter as 0.2, which means the real face will only change a little but the style face will be similar to the real face. The effects of the trick will also be shown in Section 5.

4.3. Deep Image Analogy

We use deep image analogy to realize the style transfer. Assume that the input images called A and B', and the generated images called A' and B, which A' has the same structure as A and the same content as B', similar for the B to B'. In this strategy, we first compute the features of input images A and B' through pre-trained CNN —VGG19. Since the VGG19 network has 5 Relu layer, we generate 5 feature maps from L1 to L5, as shown in Fig.5.

After that, at each layer, we evaluate a forward Nearest

Neighbour Field (NNF) method and reverse NNF by mapping functions $\phi^L_{a\to b}$ and $\phi^L_{b\to a}$, respectively. $\phi^L_{a\to b}$ maps a point in feature map F^L_A to another in feature map F^L_B. Since the similarity of A, B and A', B', $\phi^L_{a\to b}$ also maps $F^L_{A'}$ to $F^L_{B'}$. $\phi^L_{b\to a}$ is similarly defined in the reverse direction. The function of $\phi^L_{a\to b}$is shown below.

$$\phi^L_{a\to b} = \underset{q}{argmax} \sum_{x\in N(p), y\in N(q)} (||\bar{F}^L_A(x) - \bar{F}^L_B(y)||^2 + ||\bar{F}^L_{A'}(x) - \bar{F}^L_{B'}(y)||^2) \quad (1)$$

where $N(p)$ is the patch around point p. We set the patch size to b e 3×3 when L=5,4,3 and 5×5 when L=2,1.

Then, we will use this function to do reconstruction and up-sampling steps, as shown in Fig.6. Those two steps aimed to generate the merged feature and the ϕ function in the previous layer. We use the following equation to generate the feature map of previous layer (L-1 layer):

$$F^{L-1}_{A'} = F^{L-1}_A \dot{W}^{L-1}_A + R^{L-1}_{B'} \dot{(}1 - W^{L-1}_A) \quad (2)$$

where the $R^{L-1}_{B'}$, which is randomized initially, can be solved by minimizing the following loss function:

$$\ell_{R^{L-1}_{B'}} = ||CNN^L_{L-1}(R^{L-1}_{B'}) - F^L_{B'}(\phi^L_{a\to b})|| \quad (3)$$

and the W^{L-1}_A is represented by:

$$W^{L-1}_A = \alpha_{L-1} M^{L-1}_A \quad (4)$$

where M^{L-1}_A is a function that specifies the magnitudes of neuron responses at layer L-1, in order to preserve content structures from A when they are present. We apply a sigmoid function to get M^{L-1}_A:

$$M^{L-1}_A(x) = \frac{1}{1 + exp(-\kappa \times (|F^{L-1}_A(x)|^2 - \tau))} \quad (5)$$

where $\kappa = 300, \tau = 0.05$ and $|F^{L-1}_A(x)|^2$ being normalized to [0, 1].

The specific derivation is mentioned in [4]. After we repeat those two steps five times, we can recover the input image of this net, in other words, the transformed images we want.

4.4. Patch Matching Algorithm

The basic patch matching algorithm is a randomized algorithm for quickly finding approximate nearest neighbor matches between image patches [1]. This approach is divided into 3 processes, shown in Fig.7. Assume that we want to find the best match of the object (described by the boxes in A) in B. At first, we will initialize a random patch offset for each cell. Then, check if the offset from neighboring patches give a better matching. If so, adopt neighbour's patch offset. Finally, choose the best patch in step two, and search for better patch offsets within a concentric radius around the current offset and halve the radius each time. This process end until radius less than or equal to 1.

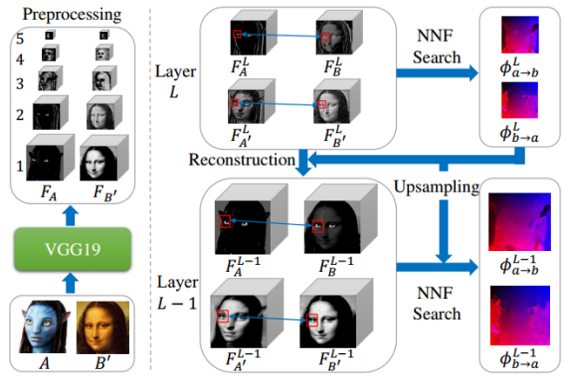

Figure 6. Pipeline of Deep Image Analogy

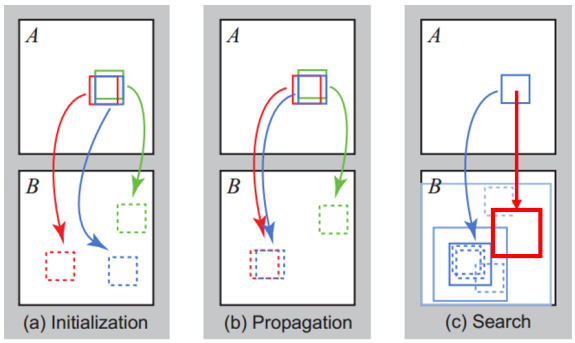

Figure 7. Patch Matching Pipeline.

However, since the random search's time complexity is $O(N^2)$, accompanied with the increase of image size, it is hard to adapt its computing demand. Therefore, we suppose to utilize landmark in order to reduce the run time.

As mentioned in Section 4.1, for each source image and target image, we can generate the face landmark. Therefore, for the corresponding features of human face, we can obtain the landmarks and the corresponding regions. That is to say, for each small patch, we only need to do sliding window search in those regions, which will improve our computing efficiency. Section 5.3 provides the improved run time data.

5. Experiments and Results

As mentioned above, the principle and methods are described in Section4. So in this section, results are represented mainly, as well as some simple explanations.

5.1. Face Landmark Detection

According to Section 4.1, the effects of the face landmark detection are shown in Fig.8,9.

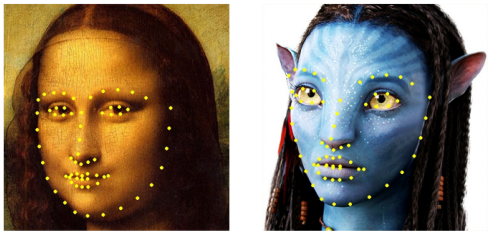

Figure 8. Face landmark detection examples 1.

Figure 9. Face landmark detection examples 2.

Figure 10. Face feature mapping example

Figure 11. Face feature mapping with parameter 0.2

5.2. Facial Feature Mapping

According to Section 4.2, the face mapping example of the classical input pairs: Mona Lisa and Avatar is shown in Fig.10. We can see that Avatar has the same face structure and facial feature position with Mona Lisa. And vise versa.

The face mapping using the trick parameter 0.2 is shown in Fig.11. We can see that the Mona Lisa's face does not changed a lot, while the Avatar's face in the structure of Mona Lisa has more smooth features.

Figure 12. The results by the origin algorithm

Figure 13. The results by our improved algorithm

5.3. Run Time

Because in the step of facial feature mapping, the source face and the style face will be aligned much more perfectly than the pictures without feature mapping. We can improve the step of patch matching algorithm a lot. The time complexity of our algorithm has been decreased a lot.

In the original work 'Deep Image Analogy', a Pytorch version deep network algorithm was realized with GPU (no type mentioned), and in this case they claim that the run time is about 30 minutes.

According to our test, a similar process but with a 18-core CPU (Xeon W-2191B) costs over 45 minutes and the main part is the deconvolution step. After modifying the CPU-based NNF algorithm as mentioned in Section 4.4, the run time reduces to about 40 minutes which is about 10-15 percent faster in total.

5.4. Deep Image Analogy

As seen in Fig.12, we can easily noticed that in the faces transferred by the origin Deep Image Analogy algorithm, the eyes of are heavily of weird shape, and the mouth is not so natural. Compared to our result in Fig.13, thanks to the changed sizes and positions of five sense organs in the inputs, the final result is not so weird. As a conclusion, the facial feature mapping procedure can highly improve the result of the original style transfer algorithm.

As shown in Fig.14, we use different morphing parameters as mentioned in Section 4.2 to do the facial feature mapping step and then the style transfer. According to the result, the transferred with parameter 0.2 will give the best result as we predict in Section 4.2.

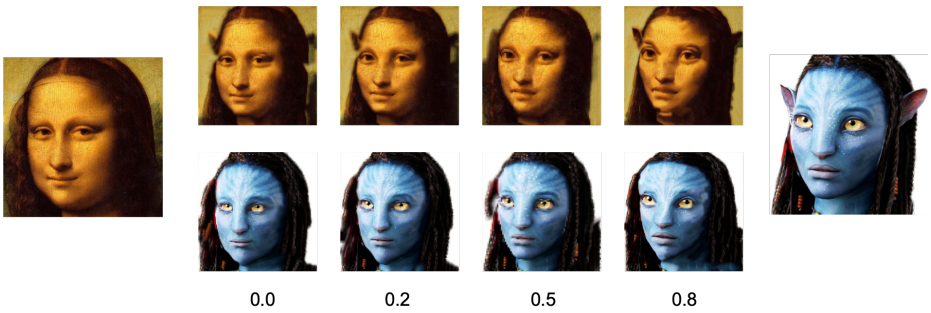

Figure 14. Different morphing parameters will give different style transferred results

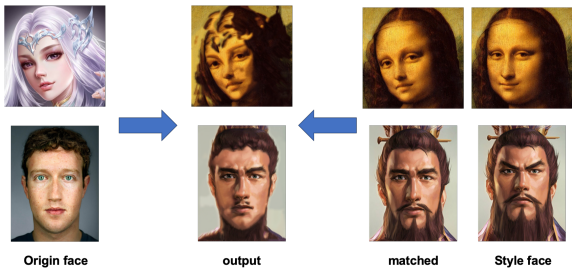

Figure 15. Better results

Figure 16. Bad results

5.5. Other Results

To test the serviceability or effects of our algorithm, we ran lots of distinct pictures with different sources and style image. Some results are displayed in Fig.15 and Fig.16. In Fig.15,there are two sets showing satisfactory effects of face style transfer.

While in Fig.16,we also get some bad results. We observe those results and try to find some reasons why those input pairs have worse effects. First is the complex structure, for example, the hair and the collars. In those cases, the style will be transferred indistinctly and seems in chaos. Second is the light and shadow. The edge between bright and dark in one picture may be recognized as a feature but the algorithm can not find a corresponding feature in the other input picture. And also the decorations will affect the style transfer, they may confuse the features.

6. Conclusion

In this work, we present an attempt of face style transfer. Based on a former research, we observe the imperfections in face issues. Then we propose two solution steps. One is to map the facial features first, and then an optimization is added to the origin algorithm. Although there are some problems, all works we do have proper results as we estimated and the final results of the face style transform are satisfactory.

Certainly, there still are some improvements can be completed. For example, applying the style transfer to a whole human body in the photo with the help of segmentation. Or further optimizing the NNF algorithm like adding some heuristic ranges to choose the most suitable feature.

References

[1] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. In *ACM Transactions on Graphics (ToG)*, volume 28, page 24. ACM, 2009.

[2] 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.

[3] 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.

[4] J. Liao, Y. Yao, L. Yuan, G. Hua, and S. B. Kang. Visual attribute transfer through deep image analogy. *arXiv preprint arXiv:1705.01088*, 2017.

[5] 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.