Style Transfer for Headshot Portraits

Haard Panchal - 201501153
Saurabh Ravindranath - 201501159
Eashwar Subramanian - 201501163

Abstract

The aim of the project is implement Style Transfer of Headshot portraits using the process of image morphing. We first explore the process of morphing images, by implementing the warping and cross dissolve processes. We then use the above, to implement Style Transfer of Headshot portraits from an example picture to the input by generating correspondance points automatically, Morph the example and combine them.

1 Introduction

Headshot portraits are a popular subject in photography. Professional photographers spend a great amount of time and effort to edit headshot photos and achieve a compelling style. Different styles will elicit different moods. A high-contrast, black-and-white portrait may convey gravity while a bright, colorful portrait will evoke a lighter atmosphere. However, the editing process to create such renditions requires advanced skills because features such as the eyes, the eye-brows, the skin, the mouth, and the hair all require specific treatment. Further, the tolerance for errors is low, and one bad adjustment can quickly turn a great headshot into an uncanny image. To add to the difficulty, many compelling looks require maintaining a visually pleasing appearance while applying extreme adjustments. Producing such renditions requires advanced editing skills beyond the abilities of most casual photographers. This observation motivates the development of the algorithm: to introduce a technique to transfer the visual style of an example portrait made by an artist onto another headshot. Users provide an input portrait photo and an example stylized portrait, and the algorithm processes the input to give it same visual look as the example. The output headshot that we seek to achieve is the input subject, but as if taken under the same lighting and retouched in the same way as the example.

2 Image Morphing

2.1 Warping

To achieve warping, we perform the following steps:

1. Generate correspondence points between the two images.
2. Use the in-built Delaunay triangulation function in matlab to obtain the triangulation of the corresponding points.
3. We then interpolate the points in the two images as a function of time, t ($t \in [0, 1]$), using the following formula :

   $$(1 - t) \times p_1 + t \times p_0$$

   where p0 and p1 are the corresponding points in the two images.

In order to obtain the positions of the remaining points in the images, we use the triangulation that we have computed, we perform the following steps:

1. We first determine which triangle the given point belongs in.
2. We then compute the barycentric-coordinates of the point using the following expression :

   $$x = p_1 \times a + p_2 \times b + p_3 \times c$$

   where x is the required point, a, b, c are the vertices of the triangle and p_1, p_2, p_3 are the coefficients. Here, $p_1 + p_2 + p_3 = 1$.
3. As there are 3 equations and 3 variables, the coordinates of the point in the new image can be computed.

2.2 Computing The Image Color

We compute the color of the different images across time t using the cross-dissolve method.

1. We first warp the 2 images.
2. We then compute the image at time t using the following expression :

   $$Image_t = (1 - t) \times Image_1 + t \times Image_2$$

Here, Image1 and Image2 are the input images to the program.

2.3 Extension - Application in Animation

We have extended the image morphing idea so that given 2 images of the object in different locations, the program generates the intermediate frames necessary so that an animation is generated of the object in motion, from its position in image A to its position in image B.

This is done by choosing a set of points such that the image is contained within it, in both the images. So, as these points get interpolated over time, the positions of the triangles change uniformly, so that the position of the entire object gets changed along with it, due to the cross-dissolve algorithm used.

In order to obtain the results for this method, the object we have chosen is a hand, which is in position A in image 1 and position B in image 2. Using our program, we marked points so that the hand is contained within the points.

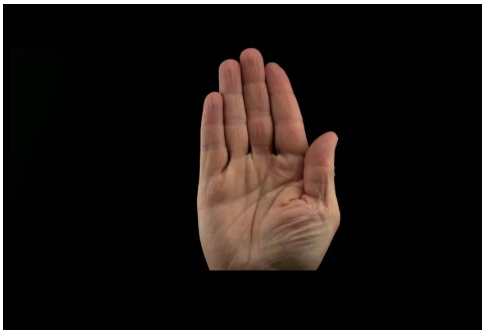

Figure 1: Input 1

These are the 2 images given as input to the program. The output of the program is the following gif:
https://drive.google.com/open?id=0B9D4yQ7vx6qVZWN6elhHdTRJSTA

2.4 Results

1. We have a demonstration of the working of the program, as it morphs 8 different images sequentially, showing the evolution of the Ford Mustang models over time. We used 20 different input points to achieve

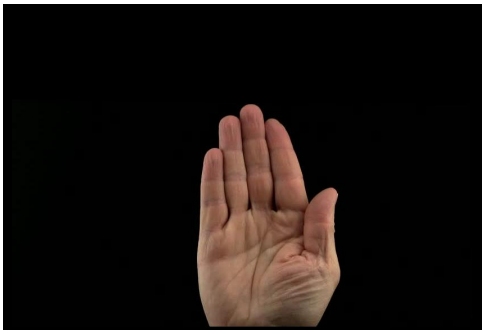

Figure 2: Input 2

our result, where images were generated at an interval of 0.025 sec:
https://drive.google.com/open?id=0B9D4yQ7vx6qVY2FLT2NfQWliaEE

2. We have a demonstration of the program morphing -
https://drive.google.com/file/d/0B48hZ3rGe8E0c2ZoaWJzY0M0UkE/view?usp=sharing

3 Style Transfer For Headshot Portraits

3.1 Correspondance Points Generation

To obtain correspondences between the input and reference images, we take a coarse-to-fine approach, using a series of off-the-shelf tools. We detect the facial landmarks using a template [Saragih et al . 2009]. This gives us 66 facial landmarks as well as a triangulated face template. We then morph the example image to match the features of the input image using the morphing program that we have implemented.

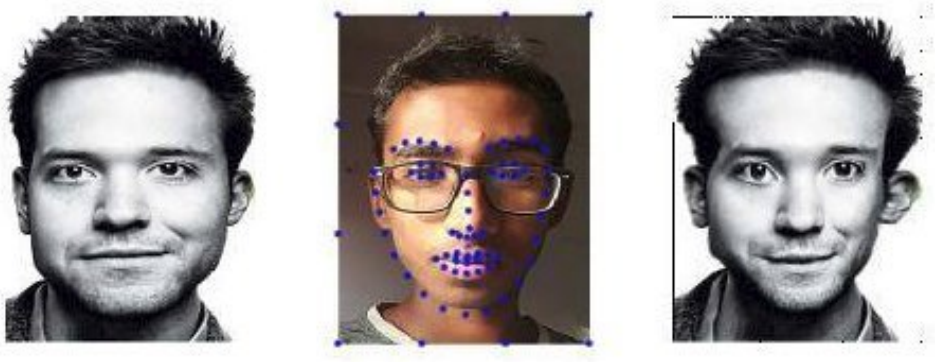

Figure 3: Correspondance Points Generation and Morphing 1

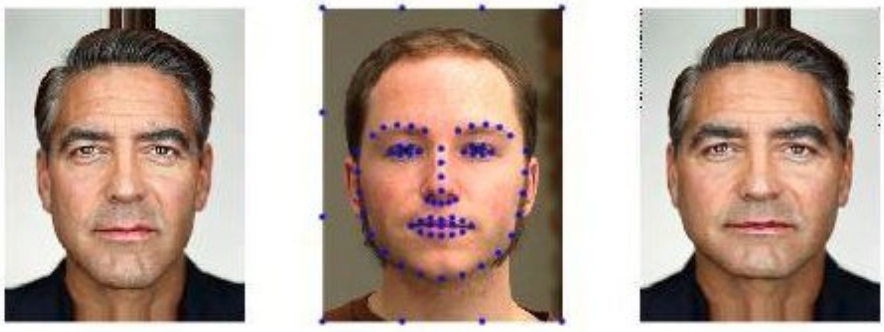

Figure 4: Correspondance Points Generation and Morphing 2

3.2 Pyramid Based Energy Transfer

We begin by constructing a Laplacian pyramid of the image, without down sampling as we proceed through the levels:

$$L_l[I] = \begin{cases} I - I \circledast G(x) & \text{if } l = 0 \\ I \circledast G(x^l) - I \circledast G(x^{l+1}) & \text{if } l \geq 1 \end{cases}$$

After experimentation, we have found $x = 1.6$ to be the optimal x value. The last layer of the pyramid having n layers is defined as:

$$L_{lmax}[I] = I \circledast G(x^{lmax})$$

The energy of an image at a certain level in the Laplacian pyramid is defined as the square of the Laplacian at that level convolved with a Gaussian filter:

$$S_l[I] = L_l^2[I] \circledast G(x^{l+1})$$

At each level, once the energies of both the images are calculated, the gain between the energy maps of the morphed and the images are computed as following:

$$gain_l = \sqrt{\frac{S_l[E]}{S_l[I] + 10^{-4}}}$$

The final ourput image is calculated by multiplying the Laplacian of that image with the gain computed. The final image is then recombined using the Laplacian layers.

$$L_l[out] = L_l[content] \times gain$$

As there can be issues with the gain maps, we limit their values, so that the values are between 0.9 and 2.8

$$RobustGain = max(min(gain, 2.8), 0.9)$$

In order to reconstruct the final output image from the Laplacian Pyramid, we simply add all the elements of the pyramid, including the residual, so that the intermediate terms are cancelled and we are left with only the image.

We use the CIE-Lab color space, as it approximates human perception, and work on each channel independently for each of the steps above.

3.3 Results and Conclusions

3.3.1 Results

Figure 5: Result - 1

Figure 6: Result - 2

Figure 7: Result - 3

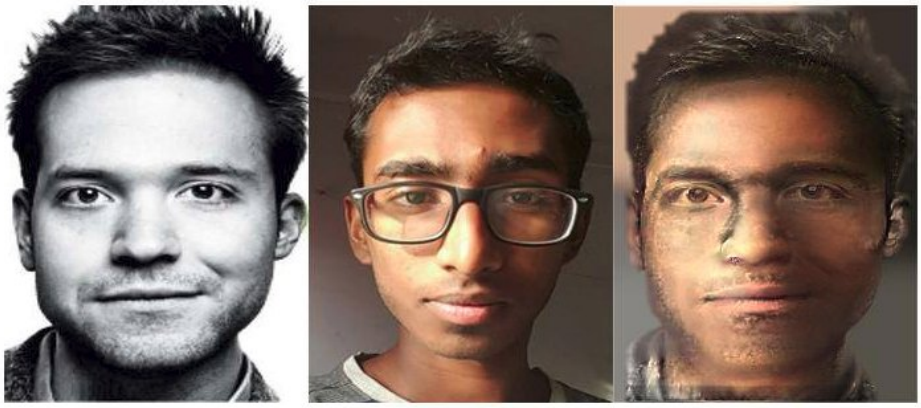

Figure 8: Negative Result - The glasses are present in the ouput image

4 References

1. YiChang Shih, Sylvain Paris, Connelly Barnes, William T. Freeman, and Frdo Durand - Style Transfer for Headshot Portraits

2. Ce Liu, Jenny Yuen, Antonio Torralba - SIFT Flow: Dense Correspondence across Scenes and its Applications IEEE (2011)