Project 2: Face Swap!

Wenyan Li
School of Electrical and
Computer Engineering
Email: wenyanli@umd.edu

***Abstract*—This project mainly focuses on implementing an end-to-end pipeline for face replacement first in two images, then in a short video. In order to do so, we first make use of the provided codes for finding our intereseted feature points (landmarks) of the individual face, then by applying triangulation and thin plate spline method independently, we can swap the faces and blend them as seamless as possible.**

I. Introduction

In this project, we implement face replacement by using Triangulation and Thin Plate Spline (TPS) respecitively. Considering each method, works have been done in the following aspects:
In face warping using triangulation,

1) Correpondences at key feature points are obtained by using face detector
2) Triangular mesh over the feature points are defined in both images
3) By using triangle correspondence, each triangle is warped separately from source to targe image.

In TPS,

1) Parameters of the TPS are estimated using information of the feature points
2) Applying the TPS parameters to transform pixels from source to the target image.

II. Face Warping using Triangulation

In this part, first, feature points(eyes, edges of the cheek, etc) are detected in the face by using provided face detector. We can obtain corresponding key feature points in two images in this step. Then a triangular mesh over the points are defined in both of the images, and the triangular meshes should be the same in the two images, which enables later step of transformation of the image pixels. I didn't check the correspondence of the triangles at the first time, and my swapping result turns out to be a mess. It's convenient to compute the triangulation by using the MATLAB delaunay function, so I simply applied the function twice, however the triangulation does not match in this case, as you can see in Fig. 2.
Corners should be assured to be the same for each triangle if wanting a successful pixel transformation and a normal swaped face.

Pixels in the target image that should be replaced later are computed by calculating the points lie in the convex hull defined by the feature points, as shown in Fig. 1.

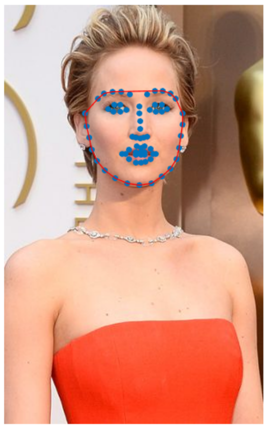


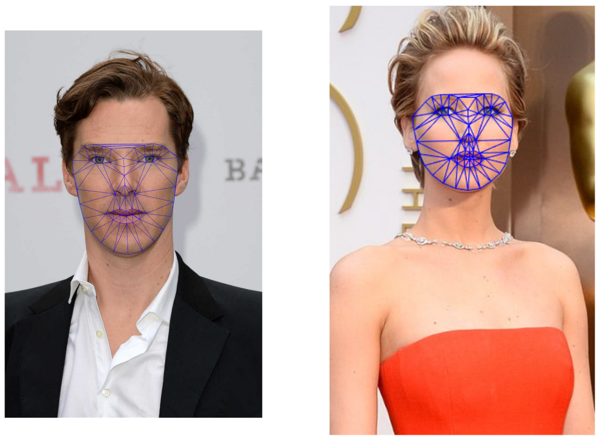
Fig. 1: Convex Hull in the target image

Each pixel in the convex hull is replaced by the corresponding pixel in the source image calculated using the barycentric equation, and in this case, interpolation is not needed since all the pixels on the face are transformed. The barycentric equation is shown in Eq.1. Hence, by corresponds the pixels in the source image with the pixels in the target, we can simply copy pixel value from the source to the target to achieve face swap. However, due to the differences in the color, brightness and saturation of the two images, a simple replacement gives us weird result, and we want to make the generated swaped face photo to be as natrual as possible, blending is needed to be done, which is discussed later.

$$\begin{bmatrix} a_x & b_x & c_x \\ a_y & b_y & c_y \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{1}$$

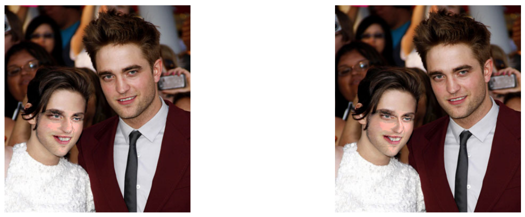
III. Face Warping using Thin Plate Spline

Face Warping using thin plate spline method can also be achieved by following the tutorial, i.e. first calculating the TPS parameters with the related feature pts on the two faces, then by applying the calculated parameters in the TPS formular, we can map all of the face pixels in the convex hull of the source image into the target image. Some features in TPS result seems to be highlighted, such the nose becomes even more



(a) Triangular mesh in source (b) Triangular mesh in target

Fig. 2: Triangles that does not match



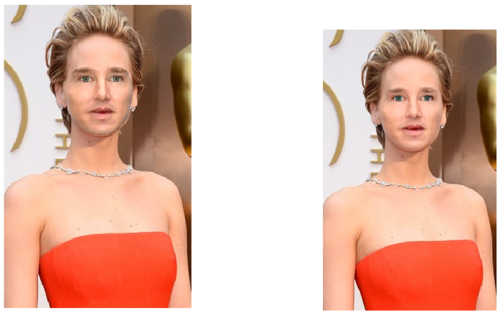
(a) Using triangular mesh (b) Using TPS

Fig. 3: Comparisons between triangulation and TPS

obvious and even a little bit skewed than in the result using triangulation. The comparison of using triangulation method and TPS is shown in the Fig. 3.

IV. Blending

Without blending, the direct result of face replacement is quite like a copy and paste result and if the two faces that are being swapped are of great difference, the result can be kind of scary. Hence, the method of cross dissolve is used for blending the faces. Mean value of the pixels are calculated for the target and the source image, and for the source image, a weight is assigned to the facial pixels so that the values of the pixels can match with the initial values of the pixels in the target image. However, though the color can be balanced for two faces to some extent using cross dissolve, the edge of the convex hull which contains the facial pixels can be still obvious. Thus, we want to smooth the edge before the face is replaced. By creating the mask for the face region and apply the gaussian filter on the warped face, a better blending result can be obtained, and the face becomes more smooth and natural. Here's the result of the face blending using source face of "Shelock" and Jennifer in Fig ̃refbc. Possion blending is also experimented in this step, however the result is not satisfying as the color of the face is not balanced as shown in Fig. 5.



(a) Simple cross dissolve (b) Using mask for blending

Fig. 4: Blending results

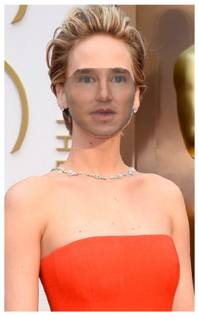


Fig. 5: Possion blending result

V. Face replacement in video

Theoretically, replacing face in video should be not a hard problem after successfully swap faces in a single frame. However, I got problems when dealing with videos. First problem is that the size of each frame in my video is kind of large, so it takes forever the first time I directly use each frame of the video and do a face swap with the source image. Then when I try to resize the target frames as target images and the source image and use them in the replacement, there were problems related with using tsearchn function(which should not happen since the face replacement does not require specific size of the input images).

A. Speed up!

However, since I don't have enough time to debug, I take a compromise method to speed up, i.e. instead of taking each frame of the video and replace the face with the source image, I took the images out from every two frames, and replaced the face with the face in the source image. This method does not influence the result much for videos that do not contain fast movement. However, if take images out from every three frames, the video becomes discontinuous.

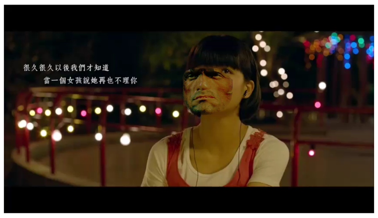


Fig. 6: Testset result of a single frame

B. Testset result

When dealing with the testset, there's problem that there's no face that can be detected from the video at first. Again, I suffered from error produced at the step of using tsearchn. So I only do a face replacement on a single frame. I admitted that the result of this part is poor.

VI. Conclusion

In this project, face replacement using the method of triangulation and TPS is succefully implemented for faces in images(frames). The results of using the two methods are compared and the encountered problems in the process are analysed and fixed. Meanwhile, different blending methods are experimented for a better replacement result. However, for the part of face replacement in video, there is much work remaining to be done (since my project for this part is not complete. I know that this should not be an excuse, but I do wish I could have more time on this project, however I had a midterm on the same day as the project deadline and another project due).

References

[1] Patrick Perez, Michel Gangnet, and Andrew Blake. Poisson image editing. *ACM Transactions on Graphics (TOG),* volume 22, pages 313-318. ACM, 2003.

[2] Masayuki Tanaka. Poisson Image Editing. Retrieved March 5, 2017, from https://www.mathworks.com/matlabcentral/fileexchange/37224-poisson-image-editing/content/PoissonEdiitng20151105/imgrad.m