CMSC733: Project 2 - FaceSwap

Hossein Souri
Center for Automation Research, UMIACS
University of Maryland, College Park
Email: hsouri@umiacs.umd.edu

I. Introduction

The aim of this project is to implement and end-to-end pipe line to swap faces in a given video. To tackle this problem we will use two method; traditional and the deep learning approach. We will explain each part with more detail in the following sections.

II. Phase 1: Traditional Approach

In this section, we will take advantage of a traditional method to do the face swap. In the traditional approach w do facial landmark detection, face warping, replacing, and blending. An overview of the traditional method is shown in figure 1.

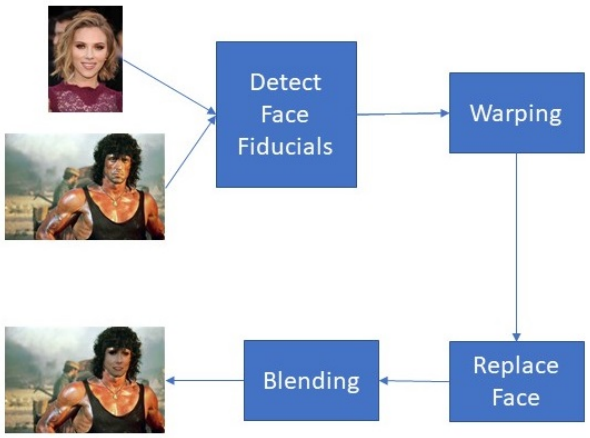

Fig. 1: Overview of the face replacement pipeline

A. Facial Landmarks detection

The first step in the traditional method is to find extract landmarks features. One of the major reasons to use facial landmarks instead of using all the points on the face is to reduce computational complexity. To extract the landmarks in the face, we use dlib library in OpenCV whcih is basically a ResNet model trained on human faces. In the figures 2-3 you can see examples of the landmark detection. The dlib gives 68 feature points. Our code is

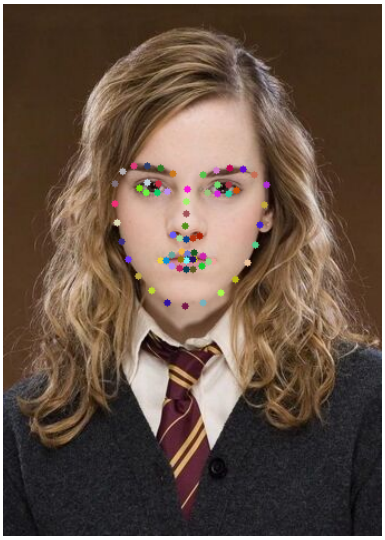

Fig. 2: Facial landmarks detected in Hermione's face

Fig. 3: Facial landmarks detected in Ron's facee

B. Warping using Triangulation

After detecting the landmarks in the faces, the next step is to ideally warp the daces in 3D. Even though we don't have the 3D information, we can use 2D image information

to approximate 3D information of the face. In the triangulation method we use the facial landmarks as corners and we assume that in each triangle the content is planar and hence the warping between the triangles is affine. We do so using Delaunay Triangulation. This algorithm tries to maximize the smallest angle in each triangle, and from this, we can obtain the same triangulation between the source and the target face image. The function cv2.getTriangleList() in cv2.Subdiv2D class of OpenCV is used to implement the Delaunay Triangulation [1]. We use inverse warping so that there is no loss of information while swapping the pixel information of the faces. In the figures 4-6 you can see examples of the triangulation method.

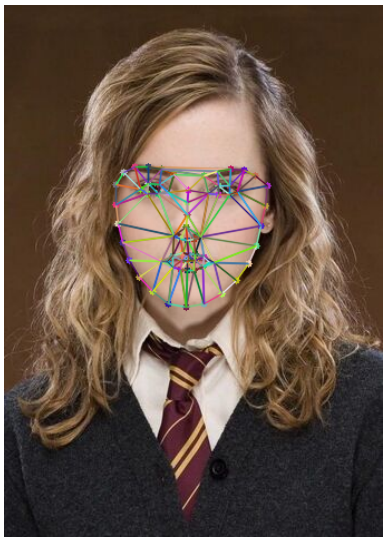

Fig. 4: Triangulation in Hermione's face

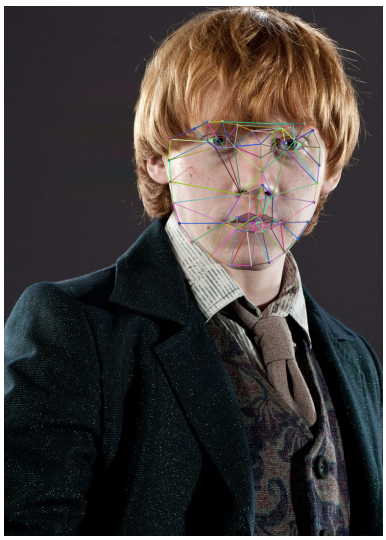

Fig. 5: Triangulation in Ron's facee

Warping is performed as explained in the following steps:

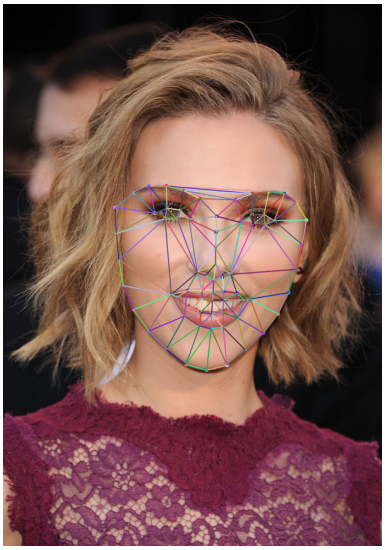

Fig. 6: Triangulation in Scarlett's facee

1) Step1: Computing Barycentric coordinates using the following equation:

$$\begin{bmatrix} B_{a,x} & B_{b,x} & B_{c,x} \\ B_{a,y} & B_{b,y} & B_{c,y} \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{1}$$

Here the barycentric coordinates are given by $[\alpha\ \beta\ \gamma]^T$. So, for each triangle we just need to get the inverse of the B_Δ:

$$\begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = B_\Delta^{-1} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2}$$

Now, given the values of $\alpha + \beta + \gamma$ we can say that a point x lies inside the triangle if $\alpha \in [0, 1]$, $\beta \in [0, 1]$, $\gamma \in [0, 1] and \in (0, 1]$.

2) Step2: Given the barycentric coordinates, we compute the pixel locations in image A using:

$$\begin{bmatrix} x_A \\ y_A \\ z_A \end{bmatrix} = A_\Delta \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} \tag{3}$$

Where A_Δ is given by:

$$A_\Delta = \begin{bmatrix} A_{a,x} & A_{b,x} & A_{c,x} \\ A_{a,y} & A_{b,y} & A_{c,y} \\ 1 & 1 & 1 \end{bmatrix} \tag{4}$$

After we obtain $[x_A\ y_A\ z_A]^T$, we need to convert the values to homogeneous coordinates as follows:

$$x_A = \frac{x_A}{z_A}\ and\ y_A = \frac{y_A}{z_A} \tag{5}$$

In the figures 7 you can see the warped face.

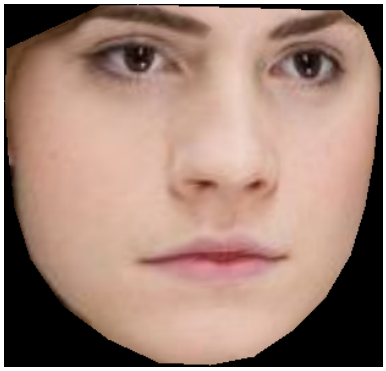

Fig. 7: Hermione's warped face using trangulation

III. Replace and Blending

The next step is replace the pixel value at location (x_A, y_A) from image A back to image B. Then we need to do the blending. For blending we use the method and codes in [2], that corrects color of the warped face according to the destination image. We use cv2.seamlessClone for this operation. In the figure 8 you can see the the output of face swapping using triangulation.

Fig. 8: Final output of face replacement using triangulation after blending

We can also do this face swapping on the frames from a video. In the figures 9-11 you can see two sample output of face swapping on test data using triangulation. Clearly the output in figure 11 is not very good. The reason is that the triangulation is not the best method for face swapping and also the faces in that video are of low resolution.

A. Warping using Thin Plate Spline

Since the human face has a very complex and smooth shape, triangulation is not the best method for warping. A better way to do the transformation is by using Thin Plate Splines (TPS) [3] which can model arbitrarily complex shapes. To perform

Fig. 9: Sample output of replacing Rambo's face on test video using triangulation

Fig. 10: Sample output of replacing Scarlett's face on test video using triangulation

Fig. 11: Sample output of replacing two face on test video using triangulation

face swapping we want to compute a Thin Plate Splines (TPS) that maps from the feature points in the Target Image (B) to the corresponding feature points in Source Image (A). A thin plate spline has the following form:

$$f(x,y) = a_1 + (a_x)x + (a_y)y + \sum_{i=1}^{p} w_i U(||(x_i,y_i) - (x,y)||_1) \tag{6}$$

Where:

$$U(r) = r2\log(r^2)$$

Like triangulation warping We perform an inverse warping here so there are no loss of information in the process. This is done by finding parameters of a Thin Plate Spline which maps from B to A.

1) Step1: In the first step, we will estimate the parameters of the TPS by solving the following equation:

$$\begin{bmatrix} K & P \\ P^T & 0 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_p \\ a_x \\ a_y \\ a_1 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_p \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{7}$$

where $K_{ij} = U(||(x_i,y_i) - (x,y)||_1)$. $v_i = f(x_i,y_i)$ and the i^{th} row of the matrix P is $(x_i.y_i,1)$. K is a matrix of size $p \times p$ and matrix P is of size $p \times 3$. In order to have a stable solution we need to compute the solution by:

$$\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_p \\ a_x \\ a_y \\ a_1 \end{bmatrix} = (\begin{bmatrix} K & P \\ P^T & 0 \end{bmatrix} + \lambda I(p+3m, p+3))^{-1} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_p \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{8}$$

Where λ is a small value and I is the identity matrix.

2) Step 2:: After the parameters of the Thin Plate Spline have been computed, all pixels in image B are transformed by the TPS model. Aftr this transformation has been done, the pixel value is read back from image A directly. The position of the pixels in A is generated by solving the TPS equation. In the figures 12 you can see the warped face before blending.

We then replace and blend the warped face just like previous section. In the figure 13 you can see the the output of face swapping using TPS.

We can also do this face swapping on the frames from a video.In the figures 14-16 you can see two sample output of face swapping on test data using TPS.

Fig. 12: Swaped face using TPS before blending

Fig. 13: Final output of face replacement using TPS after blending

Fig. 14: Sample output of replacing Rambo's face on test video using TPS

Fig. 15: Sample output of replacing Scarlett's face on video data using TPS

Fig. 16: Sample output of replacing two faces on test video using tps

IV. Traditional Methods Analysis

After testing on both triangulation and TPS method, we find TPS to have better warping and better results. As we discussed in previous section, since the human face is is very complex triangulation is not efficient for face warping and TPS does a better job on human faces. We should note that triangulation is better than TPS in terms of time complexity. We found TPS to be slower than triangulation.

V. Phase 2: Deep Learning Approach

For this phase, a pretrained off-the-shelf model called PRNet [4] is used to obtain face fiducials using deep learning. The approach used here implements a supervised encoder-decoder model to obtain the full 3D mesh of the face. The architecture is given in figure 17

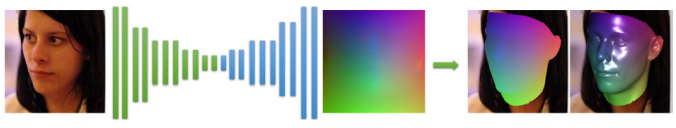

Fig. 17: The architecture of PRNet. The Green rectangles represent the residual blocks, and the blue ones represent the transposed convolutional layers.

In the figure 18 you can see the the output of face swapping using PRNet.

Fig. 18: Final output of face replacement using PRNet

We can also do this face swapping on the frames from a video.In the figures 19-21 you can see two sample output of face swapping on test data using PRNet.

Fig. 19: Sample output of replacing Rambo's face on test video using PRNet

In the figures 22 you can see the output of face swapping in a single video captured by our cell phone.

VI. Results of the test set

After experimenting on test set, we can fine that all methods fail to detect a face if the face alignment is not proper. Also, Triangulation method have more failure cases than other methods. As we see in figure 11 and 16, triangulation and TPS fail in detecting two faces in a low resolution Test2.mp4. PRNet method has shown good results compare to other methods.

References

[1] https://github.com/italojs/facial-landmarks recognition, "Facial landmarks recognition."
[2] https://github.com/wuhuikai/FaceSwap, "Face swap."

Fig. 20: Sample output of replacing Scarlett's face on video data using PRNet

Fig. 21: Sample output of replacing two faces on test video using PRNet

[3] F. L. Bookstein, "Principal warps: Thin-plate splines and the decomposition of deformations," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 11, no. 6, pp. 567–585, 1989.

[4] Y. Feng, F. Wu, X. Shao, Y. Wang, and X. Zhou, "Joint 3d face reconstruction and dense alignment with position map regression network," 2018.

Fig. 22: Output of replacing two faces in a single frame capture by a camera