Face Swap

Saurabh Palande
Masters in Robotics
University of Maryland, College Park
UID: 118133959
Email: spalande@umd.edu

***Abstract*—The aim of this project is to implement an end-to-end pipeline to swap faces in a video just like Snapchat's face swap filter or any face swap website. This is a fairly complicated procedure and variants of the approach implemented in this project have been used in many movies. The first approach is the traditional approach which consists of 2 methods: 1. Using Delaunay Triangulation and Barycentric coordinates 2.Using Thin Plate Spline. The second approach is the Deep learning approach to obtain face fiducials/full 3D mesh and then perform face replacement.**

I. Traditional Approach

The traditional approach consist of 4 important steps which are detecting face fiducials, Warping to other face, replacing the face and then performing blending. The complete pipeline is mentioned in Figure 1.

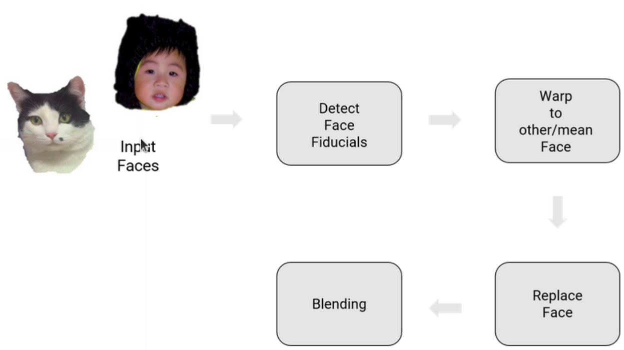

Fig. 1: Face swap pipeline

The two traditional methods of face warping used in this project are:

1) Face warping using Delaunay Triangulation
2) Face warping using Thin Plate Spline

A. Detecting Facial Landmarks

The first step in the traditional approach is to find facial landmarks (important points on the face) so that there is one-to-one correspondence between the facial landmakrs. One of the major reasons to use facial landmarks instead of using all the points on the face is to reduce computational complexity though better results can be obtained using all the points (dense flow) or using a meshgrid. For detecting facial landmarks dlib library built into OpenCV and Python is used. The detected facial landmarks of the target and source face is shown in the Figure 2 and Figure 3 respectively.

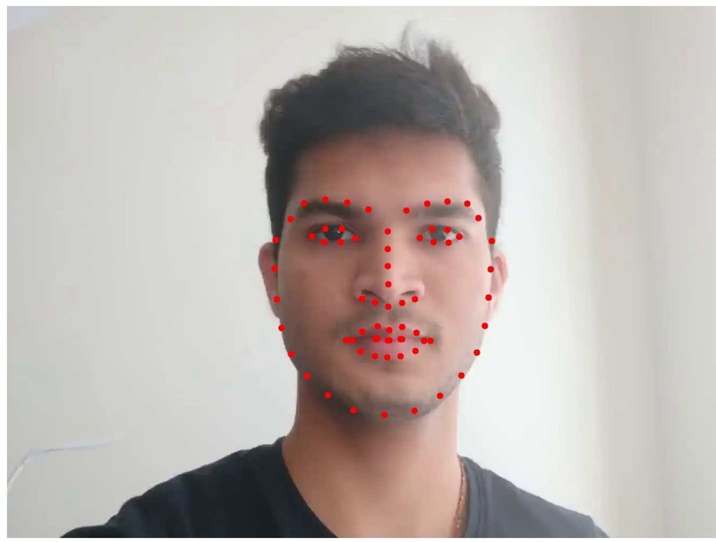

Fig. 2: Facial landmarks of Target face

Fig. 3: Facial landmarks of Source face

B. Face Warping

Now that the facial landmarks are detected, the next step is to warp the target and source faces using these landmarks. The two methods used in this project are:

1) Using Delaunay Triangulation: To swap the faces we need to warp the faces in 3D, however we don't have 3D information. So we make some assumption about the 2D image to approximate 3D information of the face by triangulation using the facial landmarks as corners and assuming that in each triangle the content is planar and hence the warping between the triangles in two images is affine. Delaunay Triangulation is used as it tries to maximize the smallest angle in each triangle. Since we use dlib to find the facial landmarks, there is correspondence between the facial landmarks and hence correspondence between the triangles. The generated delaunay triangles of the source and the target image is shown in Figure and figure respectively.

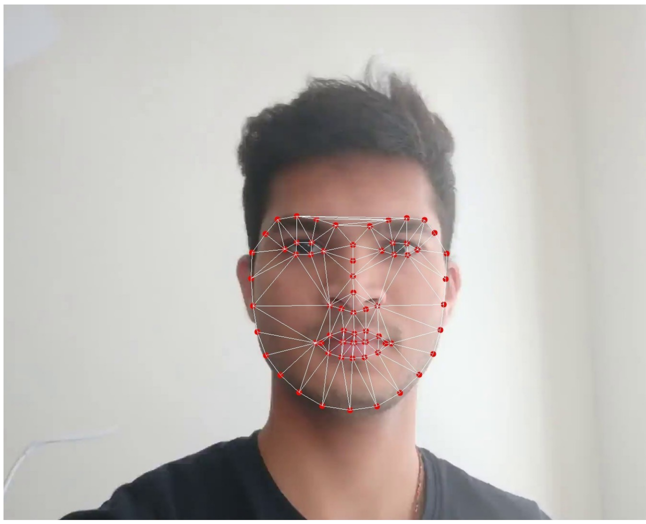

Fig. 4: Delaunay triangle of Target face

Fig. 5: Delaunay triangle of Source face

To warp one face to another the following steps are followed:

1) For each triangle in the target/destination face B, compute the Barycentric coordinate.

$$\begin{bmatrix} B_{ax} & B_{bx} & B_{cx} \\ B_{ay} & B_{by} & B_{cy} \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{1}$$

Given a point (x,y,1), the barycentric coordinates can be found as

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

The point (x,y,1) lies inside the triangle if α, β, γ lie in [0,1] and $\alpha + \beta + \gamma$ lies in (0,1].

2) Now that we have the barycentric coordinate, we can find the corresponding pixel position on the source image.

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

where

$$A_{\Delta} = \begin{bmatrix} A_{ax} & A_{bx} & A_{cx} \\ A_{ay} & A_{by} & A_{cy} \\ 1 & 1 & 1 \end{bmatrix} \tag{4}$$

3) After obtaining $\begin{bmatrix} x_A & y_A & z_A \end{bmatrix}^T$, we need to convert them to homogeneous coordinates as follows

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

$$y_A = \frac{y_A}{z_A} \tag{6}$$

4) Copy the value of pixel at (x_A, y_A) to the target location. For this scipy.interpolate.interp2d is used.

2) Using Thin Plate Spline: Face warping using triangulation assumes that we are doing affine transformation on each triangle. This might not be the best way to do warping since the human face has a very complex and smooth shape. A better way to do the transformation is by using Thin Plate Splines (TPS) which can model arbitrarily complex shapes. We compute a TPS that maps from the feature points in B to the corresponding feature points in A . We define two splines, one for the x coordinate and one for the y having the following form:

$$f(x, y) = a_1 + (a_x)x + (a_y)y + \sum_{i=1}^{p} w_i U(||(x_i, y_i) - (x, y)||_1) \tag{7}$$

where $U(r) = r^2 log(r^2)$ We find the parameters of a Thin Plate Spline which will map from B to A. Warping using a TPS is performed in two steps which are as follows:

1) In the first step, we will estimate the parameters of the TPS. The solution of the TPS model requires solving the following equation:

$$\begin{bmatrix} K & P \\ P_T & 0 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ : \\ : \\ w_p \\ a_x \\ a_y \\ a_1 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ : \\ : \\ v_p \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{8}$$

where $K_{ij} = U(||(x_i, y_i)(x_j, y_j)||_1)$, $vi = f(x_i, y_i)$ and the ith row of P is $(x_i, y_i, 1)$. K is a matrix of size size p×p, and P is a matrix of size p×3. In order to have a stable solution we computed the solution by:

$$\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ \vdots \\ w_p \\ a_x \\ a_y \\ a_1 \end{bmatrix} = (\begin{bmatrix} K & P \\ P_T & 0 \end{bmatrix} + \lambda I(p+3, p+3))^{-1} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ \vdots \\ v_p \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (9)$$

where $I(p+3, p+3)$ is a p+3×p+3 identity matrix. λ 0 but is generally very close to zero. This step is performed twice, once for x co-ordinates and once for y co-ordinates.

2) In the second step, we use the estimated parameters of the TPS models (both x and y directions) and transform all pixels in image B by the TPS model. Then we read back the pixel value from image A directly. The position of the pixels in image A is generated by the TPS equation. The output images are shown in the figures below:

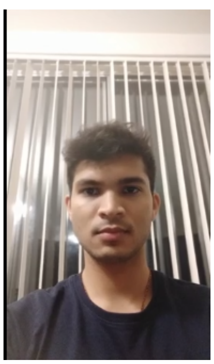

Fig. 6: Original image

Fig. 7: Swapped face by Delaunay

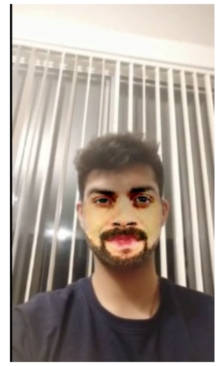

Fig. 8: Swapped face by TPS

Fig. 9: Original image

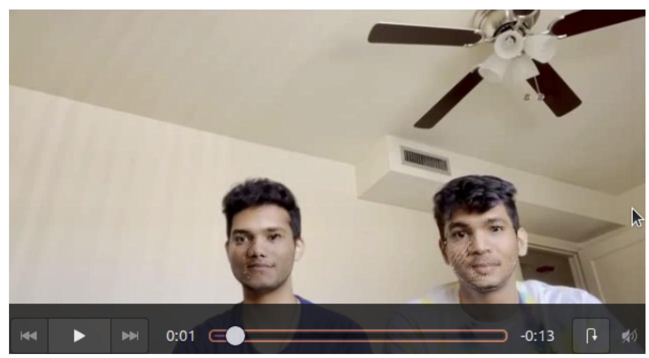

Fig. 10: Swapped face by Delaunay

Fig. 11: Swapped face by TPS

REFERENCES

[1]