RBE 549: Project 2 - Face Swap

Chinmay Kate
M.S. Robotics Engineering
Worcester Polytechnic Institute (WPI)
Worcester, MA 01609
Email: cskate@wpi.edu

Mandeep Singh
M.S. Robotics Engineering
Worcester Polytechnic Institute (WPI)
Worcester, MA 01609
Email: msingh2@wpi.edu

***Abstract*—In this project we implement computer vision methods to do face swapping between two images or two faces in a single image. It is similar to like using Snapchat face filters. This problem looks simple at first but because human face is a 3d projection, so just replacing one face with the other will not look realistic. We will be using some advanced computer vision pipelines which uses Delaunay triangulation and Thin Plate Splines which tackle this problem. And in later section, we will see how this same problem can be tackled with the help of deep learning.**

***Index Terms*—Face swapping**

I. Phase 1 : Traditional Computer vision pipelines

A. Introduction

In Phase 1 of homework, our aim is to divide the 3d face area into multiple small areas and then try to warp those similar sections in corresponding images to get a realistic output. To, implement this we first extract face fiducial landmarks using already trained model in dlib library. Then we can use either delaunay triangulation warping method or Thin Plate Spline warping method to warp and replace the faces. And at the end we have to implement blending between the swapped face and original image to maintain a seamless texture. Fig 1 can be referred for the overview of the pipeline followed.

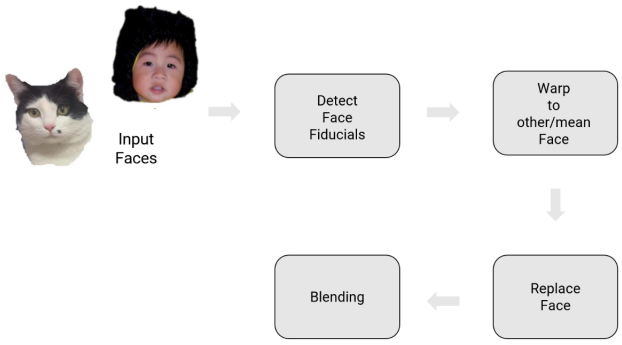

Fig. 1: Overview of the Face Swap pipeline.

B. Detecting face landmarks

To start with the pipeline of face swap, we have to detect the faces in an image and their corresponding facial landmark features. For this, we use already trained models of the dlib library. For our pipeline we have used a model which predicts 68 landmarks depicting a human face. The important thing to note is the order of these landmarks is same for all the detected faces, meaning if the 1st feature is giving us the position of the eye centre in one face then it will be the same in any other face as well. Using these landmarks we can apply warping methods to do the process of face swapping. Fig. 2 can be referred for the detected facial markers on the given image.

Fig. 2: Detected facial landmarks

C. Face Warping and Swapping

After we have detected the facial landmarks, we have to somehow implement a way to warp one face with respect to the other face in the image and then swap them. We have two methods which we are implement and are discussed in detail below.

1) Using Delaunay Triangulation: Faces can be warped between two images using delaunay triangulation on the obtained facial landmarks. In delaunay triangulation method, triangles are made using the facial landmark points as the corners of the triangles. Then the assumption is made that all the triangles within a single triangle lies in same plane so that we can do affine transformation between two such triangles of different faces. Delaunay triangulation or duals of the voronoi diagram are made such that they try to maximize the smallest angle of the triangle. Delaunay triangles are shown in fig 3. Below are the steps to be followed after getting the triangles from the facial landmarks to warp and swap the faces:

- The first thing to make sure is that the triangulation must match between the two faces so that we will warp similar part of the faces.To do this, first find delaunay triangle

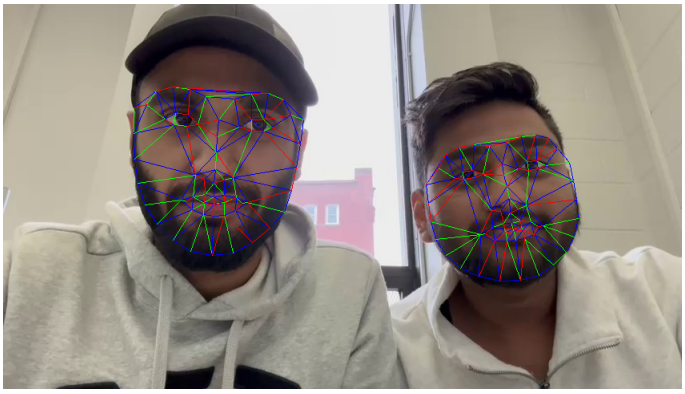

Fig. 3: Delaunay triangles

list using opencv library on the first face. Then mark the indices of the landmark list from which all the triangles are made. Using the same indices from the landmark list of the second face we will form our triangle list for the second face.

- To compute all of the following steps, we will first extract individual triangles from the source and destination faces . This is done by taking the rectangular region from the image which includes the full triangle and then applying a mask of the shape of the triangle on the rectangular region to get only the triangle region points from the image.
- For each triangle in the destination face, we compute the barycentric coordinates. Barycentric coordinates are nothing but a way to express a pixel position in terms of triangle vertices it is in. We do this for all the pixel positions within a triangle.

$$\begin{bmatrix} \mathcal{B}_{a,x} & \mathcal{B}_{b,x} & \mathcal{B}_{c,x} \\ \mathcal{B}_{a,y} & \mathcal{B}_{b,y} & \mathcal{B}_{c,y} \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Fig. 4: Computing barycentric coordinates.

In the above figure, α, β and γ are the barycentric coordinates for position (x,y) within the triangle whose vertices are (Bax,Bay), (Bbx,Bby) and (Bcx,Bcy).

- Repeat the above step of calculating barycentric coordinates for all the triangles in the destination face.
- Now, we will calculate the the corresponding pixel position in source face which will be equivalent to pixel position in destination face. We use our calculated barycentric coordinates for this purpose as per the below equation.

$$\begin{bmatrix} x_{\mathcal{A}} \\ y_{\mathcal{A}} \\ z_{\mathcal{A}} \end{bmatrix} = \mathcal{A}_{\Delta} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix}$$

Fig. 5: Computing pixel positions in source image.

We know the barycentric coordinates, triangle vertices in source face and we compute the pixel positions (Xa, Ya). Repeat this for all the triangles.

- Now, we have the equivalent pixel positions in both faces. We just have to copy back the pixel value from a pixel position in source triangle to the corresponding pixel position in destination triangle. But the computed coordinates of the source image will be floating point data, so we have to do interpolation to get the desired intensity value. We use bilinear interpolation for this purpose.
- Once we interpolate and copy all these intensity values from source face triangles to destination face triangles, our warping and swapping part is complete. Notice that we did inverse warping process as we first calculated the pixel positions in the source image and then copied that intensity value in the target pixel position. Results after warping can be seen in fig (shown only for 1st face).

2) Using Thin Plate Splines: The results of warping by Delaunay triangulation may not be great because we assume that all the points in a triangle lies in the same plane which is not true as human face has a complex structure. TO tackle this problem, one more algorithm can be used called Thin plate splines which can warp more complex shapes by fitting a spline equation to it. Interesting part is in Thin plate spline there are components of both, affine warp as well as warp using spline equation.
Following steps are used to warp faces using Thin Plate Splines method:

- A thin plate spline has the following form.

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

Fig. 6: Thin Plate spline model.

Where the first three terms are affine part of the transformation and rest is the spline formulation part. We have a, ax, ay and Wi as the unknowns of this equation.

- To calculate the above unknowns we solve have to solve the following equation:

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

Fig. 7: Equation to compute TPS parameters.

Here we have taken, v1, v2,....vn as the x,y landmarks points of the destination face. The λI matrix is added

so that inverse of the corresponding matrix is always computable and thus give a stable solution.
This equation has to be solved twice, once for x coordinates and secondly for y coordinates. But, by proper vectorization both x and y coordinates parameters are computed in single pass only.
Note that U(r) has a log function in its equation and we know log is not defined at 0. So, input U(0) = 0 to avoid any errors in code.

(a)
Original image

(b)
Results after using Delaunay triangle warping method

(c)
Results after using Thin Plate Spline warping method

Fig. 8: Face warping and swapping

- Input the solution of Wi, a, ax, ay and az in the TPS model first for individual x coordinates and then y coordinates of the source face. We will get warped x and warped y position of the destination face. Now, equate intensity values at these warped positions equal to the corresponding intensity values in source face.
- Apply mixed blur on the output to smoothen out the output in case there are any missing pixel values.

D. Blending

After warping and swapping one face to another, the result will not look natural because of color variations among the two faces. To avoid this, we implement poission blending. We have used opencv function of seamless clone to implement blending operation. The results before and after blending can be seen in Fig.

(a)

(b)

Fig. 9: Results before (a) and after (b) blending

E. Results Analysis

There are some observations regarding both the methods used for warping and swapping the faces which are discussed in below points:

- We observed that Facial landmarks using dlib library works only if the faces are straight and facing the camera. Because of this sometimes only one face is detected in image.

- Because we are processing on every frame of the video, the warping results are not very consistent and thus results in video stability issues.
- Results of TPS are better than Delaunay triangles as the spline shape is calculated at a particular point using all other landmarks whereas in delaunay triangles triangle warping doesn't give best results because our face is not exactly a plane in those individual triangles.
- There can be some distortions in the resultant swapped image. So, we have used Mixed blur to even out those distortions. Any other type of blur like Gaussian blur can also be implemented.

II. Phase 2 : Deep Learning Method

A. Introduction

In this Phase, Off the shelf model from "Towards fast, accurate and stable 3D Dense face alignment" proposed by Jianzhu Guo et al is implemented. Here we use novel regression framework named 3DDFA- V2 to obtain output as 3D face fiducials, 3D depth and mesh. Further these full mesh are used to swap the faces and blend as per the classical method presented above.

B. 3DMM

3D Morphable model has been proved be more efficient compared dense vertices regression as they countered the problem of checkerboard artifacts due to deconvolution operators and has beautiful resolution images. Here small set of 3DMM parameters are regressed which have low dimensionality and low redundancy. 3DMM parameters influence the reconstructed 3D face differently, making regression tough and need to dynamically re-weight each parameters according to their importance in training. MobileNet is used as we regress small no of parameters.

C. Pipeline

This architecture consists of 4 parts lightweight backbone like MobileNet for predicting 3DMM parameters, Meta joint optimization technique (WPDC and VDC dynamically chooses best parameters ahead and converges faster), the landmark regression regularization(further elevating the facial lankmarks predicting accuracy) and short video synthesis to improve stability in 2D face alignment are used in training.

D. 3DMM Training

Here we define 3D face mesh, 3D face is reconstructed and projected onto the image plane with scale Orthographical projection.

$$V_{2d}(\mathbf{p}) = f * \mathbf{Pr} * \mathbf{R} * (\overline{S} + \mathbf{A}_{id}\alpha_{id} + \mathbf{A}_{\mathbf{exp}}\alpha_{\mathbf{id}} + t_{2d})$$

This is the Projection function generating 2D projection of model. Parameters of 3DMM are factor, Transition matrix containing Rotation and translation, pitch, yaw, roll. Entire architecture is MobileNet and output is feed to Landmark regression and meta joint optimization.

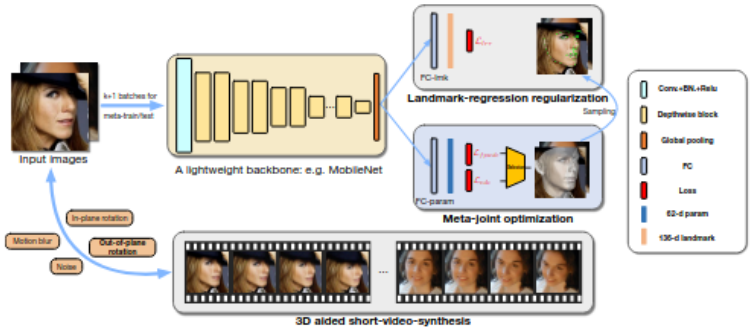

Fig. 10: Pipeline of 3DDFA-V2 Framework.

E. Meta-joint Optimization

Here we have 2 cost functions Vextex distance cost(VDC) and Weighted Parameter distance cost(WPDC). VDC optimizes by minimizing vertex distance between fitted 3D face and ground truth. WPDC assigns different weights to parameter. In training process model looks ahead k-steps with above cost functions then selects better one between cost functions based on errors and converges faster than vanilla joint optimization.

F. Landmark regression regularization

In 3D face reconstruction, the 2D sparse landmarks after projecting are usually used as and extra regularization to facilitate the parameter regression. Here we regress landmarks using L2 loss function. We use 68 facial 2D landmarks to regress.

G. 3D aided short video synthesis

In videos stability is critical issue. Here stability means changing of the 3D reconstructed across adjacent frames should be consistent with true face moving in fine grain level. To counter this 3D aided short synthetic video in mini-batch is generated. Common pattern like noise, motion blur, in-plane rotation, out-plane face movement can be modelled. We apply these transformation and short 3D video is generated and used to improve video stability.

Fig. 11: Ouputs from facial landmarks.

Fig. 12: Mesh Output.

Fig. 13: 3D face reconstructed.

H. Conclusion

Author of the above research paper have implemented the methodology described above. We used their code to generate facial landmarks, mesh and 3d depth information for our data. We tried implementing the face swap and blending the results but couldn't come up with reasonable results.

I. Test results

(a) (b)

original images

(c) (d)

Warped images c) Delaunay d) TPS

Fig. 14: Face warping and swapping