Classical and Deep Learning Approaches for Face Swapping

Kumar Gaurav
Department of Computer Science
University of Maryland College Park

Shreya Suresh
Department of Computer Science
University of Maryland College Park

I. Phase I

Face swap is a common computer vision application seen in many applications like Instagram, Snapchat etc. This project was to implement face swap in various scenarios, one of the scenario where face from an image is being swapped to face in the video, second is to swap two faces present in the video, itself. Third video, the face is exposed to various lightning conditions, in the dark. Therefore, this project was challenging in terms of understanding of the concept and its implementation.

A. Facial Detection

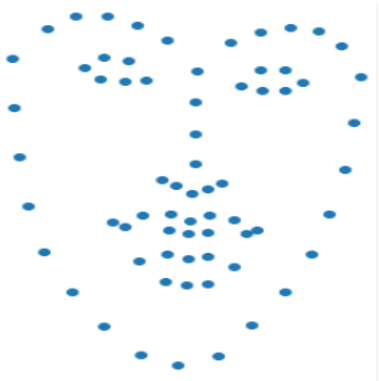

Fig. 1. Facial Markers identified by dlib

It starts with facial keypoint detection. Our first step would be to identify important facial features boundary of nose, ears, eyes, lips and chin. But we need not implement it from scratch. One interesting library already exists known as dlib. It is pre-trained on a large set of image data. It gives a set of 68 points inside a bounding box for a face.

B. Face Warping using Triangulation

Face is a 3D data, which needs to be swapped with another face/3D data. Since we have extracted keypoints, We can make triangles as only three points will be needed for affine transformation and each triangle can be approximated as 2D surface. Starting with the face which we want to copy as Face1 and face on which it needs to be placed as Face2. The algorithm is as follows:

- Face1 is divided in many triangles using Delaunay Triangulation of subdiv module form opencv.
- Each triangle is formed by three indexes out of 68 index of facial features from dlib.
- Store these triangles as separate data in the form of indexes, will be used later.
- We need to find barycentric coordinates of integral points in each triangle, using an equation given in project page.
- Finding integral points by checking each point, whether it is in triangle or not, is computationally expensive.
- A simple idea is to fix smallest rectangle, in which a triangle can fit.
- Now, the computation cost reduces from O(W[image]*H[image]) to O(Length of largest size of triangle * Height of triangle) for computation of integral points in each triangle.
- After computation of barycentric coordinates, match corresponding triangles in Face2 using triangle indexes of Face1. Otherwise, there will not be perfect triangle matching and will lead to bad correspondences.
- Lastly, by inverse warping facial pixels can be copied to another face.

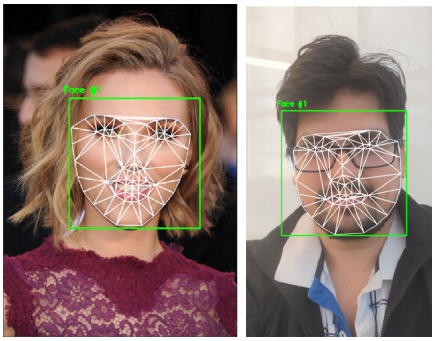

Fig. 2. Delaunay Triangulation Output

C. Face Warping using Thin Plate Spline

In this part, we using thin plate spline to perform warping as opposed to triangulation. It adopts the position of least net bending energy. Since this technique models arbitrarily complex shapes, it gives more *wiggle room* for the transformation of facial points. We followed steps mentioned on the course web page with few implementation modifications.

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

Fig. 3. Thin Plate Spline Equation

In the above equation, we had to add a small epsilon value before calculating U(r) since the difference of the same points yield 0. Shown below are the results for our custom data.

Fig. 4. Face warped according to TPS

D. Face Replacing and Blending

Now that we have the warped face to be copied, it is first directly placed on the target image.

Fig. 5. (Left) TPS Warping and (Right) Delaunay Triangulation

Since the skin tones of the two faces were different, the result is not visually appealing. Poisson blending was performed with the use of seamlessClone function. A mask was created for blending this new face. We noticed that by eroding the mask, we get better blending outputs.

Fig. 6. (Left) Without eroding and (Right) with eroding

Eroding was causing the mask to be smaller than the face, which resulted in the edges being blended smoothly. The seamlessClone function blends strongest around the edges of the mask and gradually reduces the *power* of the background as you approach the centre.

One of the places we encountered a problem was while assigning the centre. It was initially incorrectly chosen as the centre of the image and the blending results were no better than the warping ones. We then corrected it by first creating a bounding box around the mask and then calculating it's centre.

E. Results and Analysis

The outputs for the custom set and test sets are presented below.

Fig. 7. Result using TPS for Set 1

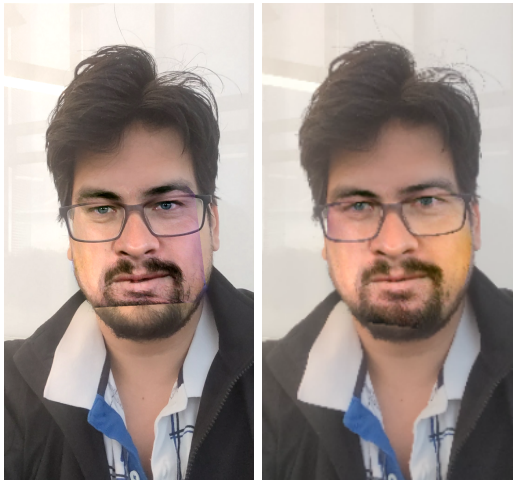

Fig. 8. Result using Triangulation for Set 1

Fig. 9. Result using TPS for Test Set 1

Fig. 10. Result using Triangulation for Test Set 1

Fig. 11. Result using TPS for Test Set 1

Fig. 12. Result using Triangulation for Test Set 1

Fig. 13. Result using TPS for Test Set 3

Fig. 14. Result using Triangulation for Test Set 3

Below mentioned are our observations on the test set provided.

- For Test2.mp4, the system did not identify the face in the video when the lighting was too dark. In an attempt to overcome this, we increased the brightness in frames where the system could not detect faces in the first attempt. This hack gave us better results but still has scope for improvement.
- Another observation was that while swapping faces, the result is better if both of them contain teeth or if both of them do not. Shown below are the results when one of the people is grinning and the other is not.

Fig. 15. Swapping where one person is grinning and the other is not

Fig. 16. Swapping where nobody is grinning

- In the video where we had to swap the face of the old man with the old woman, we noticed that it fails when they are looking at each other. This is because dlib is only a frontal face detector. It is unable to identify a face that is turned sideways.

We noticed flickering in the video when face detection does not happen in certain frames. A simple solution we used to tackle this was to write the previous frame (in which the face swapping was successful) onto the video. This improved the final video by a good amount.

II. Phase II

In this section, we used the code from Feng et al to obtain the full 3D mesh of the face and then inserted it in the pipeline we built for the traditional approach. Results are shown in the following sections, along with our analysis.

A. Results and Analysis

We observe that PRNet performs well for non-frontal faces but the classical approach using dlib performs better otherwise. It does not identify the features as well as dlib. Observations for the individual test sets are similar to that of the classical approach.

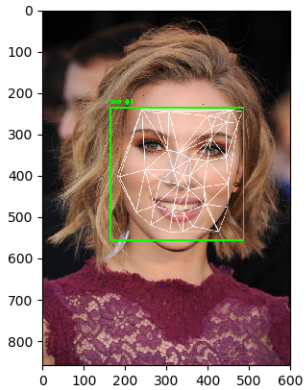

Fig. 17. Facial Markers identified by PRNet

The results for face swap using both the warping techniques for all the test videos are shown below.

Fig. 18. Result using TPS for PrNet Test Set 1

Fig. 20. Result using TPS for PrNet Test Set 1

Fig. 21. Result using Triangulation for PrNet Test Set 1

Fig. 22. Result using Triangulation for PrNet Test Set 3

REFERENCES

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

[2] Fred L. Bookstein, *Principal Warps: Thin-Plate Splines and the Decomposition of Deformations*, IEEE Transactions on Pattern and Machine Intelligence, Vol NO. 6, June 1996