Project 2 FaceSwap
CMSC733

Akanksha Patel
Master of Engineering in Robotics
University of Maryland
Email: apatel44@umd.edu

USING 1 LATE DAY

Sri Manika Makam
Master of Engineering in Robotics
University of Maryland
Email: smakam@umd.edu

***Abstract*—In this project we implement an end-to-end pipeline to swap faces. We achieve the task using two approaches: (a) classical computer vision and (b) deep learning approach. The results of the two approaches are presented and compared.**

I. Phase 1: Traditional Approach

In this section, we'll discuss the traditional approach to swap faces. We have tested our pipeline in two scenarios: (a) video with a single face swapped with an image and (b) video containing two faces, which are swapped with each other. This is achieved by following the steps below:

1) Facial Landmarks detection
2) Face Warping
   a) Triangulation
   b) Thin Plate Spline
3) Blending
4) Motion Filtering

A detailed explanation of these steps is given in the following sections.

A. Facial Landmark Detection

The first step in the pipeline is to detect facial fiducials. We obtain these landmarks using the 'dlib' library. We use a pre-trained model to obtain the location (x, y) of 68 key points on the detected face. It always returns the points in the same order, thus giving us correspondence between landmarks of both the faces we want swap.

B. Face Warping

Next step is to warp one face over the other. This can be achieved in one of the two following ways:

1) Triangulation: We ideally need to warp the faces in 3D, but we don't have any 3D information. So, we can make some assumption about the 2D image to approximate 3D information of the face. One simple way is to triangulate using the facial landmarks, obtained in previous step, as corners and

Fig. 1. Landmarks on destination and source images

then make the assumption that in each triangle the content is planar (forms a plane in 3D) and hence the warping between the the triangles in two images is affine. Triangulating or forming a triangular mesh over the 2D image is simple but we want to triangulate such that it's fast and has an efficient triangulation. One such method is the Delaunay Triangulation and can be constructed in O(nlogn). It is obtained by drawing the dual of the Voronoi diagram, i.e., connecting each two neighboring sites in the Voronoi diagram. This method tries to maximize the smallest angle in each triangle.

We have used the getTriangleList() function in cv2.Subdiv2D class of OpenCV to implement Delaunay Triangulation. When we obtained the triangles on source and destination images using the above mentioned function, we came across cases where the obtained triangles were not in correspondence with each other though there facial landmarks were in correspondence. So, we performed Delaunay Triangulation on destination image, kept track of the 3 landmarks used in forming each triangle and formed triangles on source image using the corresponding landmarks. Thus, we obtained the correspondence between triangles in both the images.

Then, for each triangle in the destination, we find points lying inside the triangle and their corresponding Barycentric coordinates (α, β, γ). Now, for every point (x,y) in each triangle in the destination, we find the corresponding point in source image using the barycentric equation. We do this for every triangle formed in the destination and get the correspondence between all points in both the faces. Now, to get the pixel values at all the obtained points on the source face, we used

scipy.interpolate.interp2d as the pixel values obtained will not necessarily be integer values. Then, we copy the obtained pixel values onto the destination face.

Fig. 2. Delaunay Triangulation on destination and source images

2) Thin Plate Splines: Warping using triangulation is achieved using affine transformations between the corresponding triangles. However, the geometry of a human face cannot be efficiently captured using affine transformations. Therefore, we use another method to perform warping.

Thin Plate Splines (TPS) is a spline based data-interpolation and smoothing technique that can be used to model arbitrary complex shapes. In two dimensions, TPS results in a surface that passes through the control points and results in minimum bending. Mathematically it can be represented in the following form:

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

where x_i, y_i are the control points, $a_1, a_2, a_3 and w_i$ are the parameters of the TPS and $U(r) = r^2 \log r^2$.

We calculate the TPS coefficients using equation 2 where $K_{ij} = U(||(x_i,y_i) - (x_j,y_j)||_1)$, P is a vector of all the detected features in the destination image, $P_i = (x_i, y_i, 1)$, $I(p+3, p+3)$ is an identity matrix and λ is a regularization coefficient (for this implementation, $\lambda = 0.000001$).

These calculated coefficients are then used to transform the pixel values of the destination image to the corresponding value in source image using equation 1. We then replace the pixels in the destination image with the corresponding source image.

$$\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_p \\ a_x \\ a_y \\ a_1 \end{bmatrix} = \left(\begin{bmatrix} K & P \\ P^T & O \end{bmatrix} + \lambda I(p+3, p+3) \right)^{-1} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_p \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2}$$

C. Blending

Simply replacing the pixels from the source image over the destination image produces an unnatural output as there are sudden illumination and color changes. A better way is to perform Poisson Blending of the two images. We have used OpenCV seamlesssClone to implement Poission Blending.

D. Motion Filtering

While testing this pipeline on a video, we have noticed that the facial landmarks are not identified in every frame. Therefore, there are frames in between that will not have faces swapped. We can overcome this problem by using Kalman Filter.

Kalman Filter is a filtering technique which recursive filter to estimate the internal state of a linear dynamic system. Our state a (272 x 1) vector that contains the position (x_i, y_i) and velocity (v_{xi}, v_{yi}) of the 68 detected features. The measurements is a (136 x2) vector that contains the position (x_i, y_i) of the facial landmarks. The process noise is a given by $10^{-3} * I(272)$ and measurement noise is given by $10^{-2} * I(136)$, where $I(c)$ represent an identity matrix of size (c x c).

E. Results and Discussion

We implemented the two methods of face swapping on two data sets created by us. Data1 is video having one face and it is swapped with face of Selena Gomez. Data2 has two faces and they are swapped with each other.

The Triangulation and TPS outputs obtained on Data Set-1, Data Set-2, Test Set-1, Test Set-2 and Test Set-3 are shown in figures 3 to 7 respectively.

The observations of the outputs obtained from TestSet are discussed below.

1) **TestSet1:** The result of both the methods on the first test set are equally good. The first test set has relatively less motion and good lightening conditions and thus the facial landmarks are detected in majority of the frames. The few frames in which the landmarks are not detected, kalman filter correctly estimates the position of the landmarks producing a continuous output. However, the facial landmarks are not being detected in few initial frames of the video and so face swap cannot be observed. The resultant video is still not very smooth and contains small vibrations in the location of landmarks, that are visible as small jitters in the video.
2) **TestSet2:** The second test set has a lot of motion and the people are not always facing towards the camera. This makes it difficult to detect the features. Therefore, there are quite a lot of frames where the facial features are not detected (especially when the front view of the face isn't available). Kalman Filter is able to estimate the position

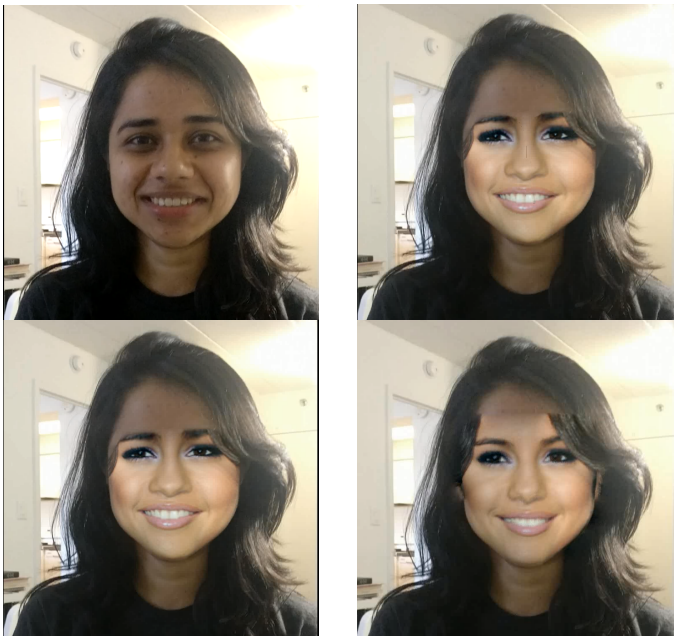

Fig. 3. Face Swapping using Triangulation, TPS and PRNet on Data Set 1

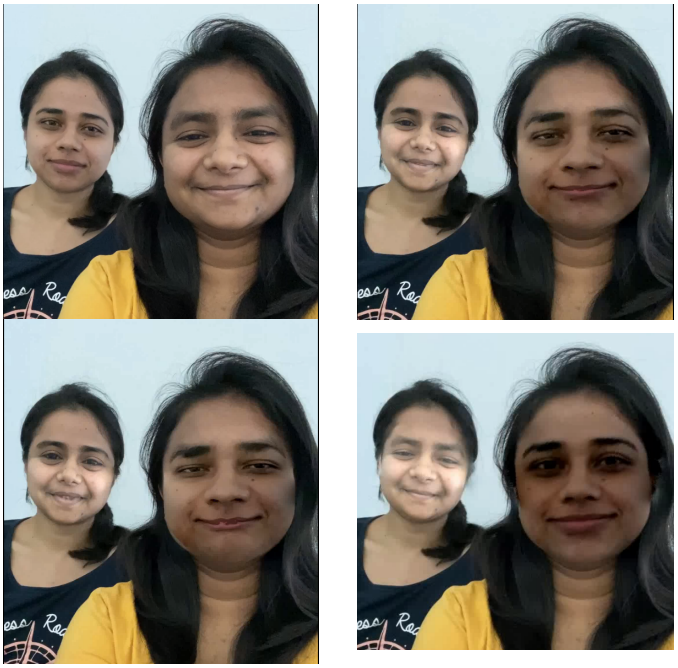

Fig. 4. Face Swapping using Triangulation, TPS and PRNet on Data Set 2

Fig. 5. Face Swapping using Triangulation, TPS and PRNet on Test Set 1

correctly if the features are missing for a short duration however it results in wrong estimates if the features are missing for a long duration. And this is what we see in this test set.

3) **TestSet3:** In the third test set, along with the motion and face alignment, we also have the problem of illumination changes. Here again, we face the same problem as faced in Test Set 2. The pipeline is not able to detect the facial landmarks in frames that are not well illuminated and kalman filter results in wrong estimates if the tracking is lost for a longer duration.

Fig. 6. Face Swapping using Triangulation, TPS and PRNet on Test Set 2

Fig. 7. Face Swapping using Triangulation, TPS and PRNet on Test Set 3

We can also observe random lines on the face in all the outputs obtained through Triangulation. This can be caused when we don't get all the points on the source face as we draw triangles on source with respect to the triangles on destination. So, we tried to do Delaunay triangulation on source and draw respective triangles on destination. This cannot work as we observed the triangles formed on destination to be overlayed on one another. Hence, we could not solve this problem.

II. Phase 2: Deep Learning Approach

In this section we have used the off the shelve pre-trained model discussed in [1] to perform face swap.

For the scenario where a single face in the video is to be swapped with an image, for each frame of the video we call the texture_editing function of PRNet with the frame and image as the inputs. The resultant images are converted to the video.

For the scenario where there are multiple images in the video, we split the frame into two images in such a way that the two biggest face detected are on two different images. These images are then pass as a parameter to the texture_editing function. We call the function twice to swap both the faces. The resultant outputs are then merged to get the frames for the output video.

A. Result and Discussion

The PRNet outputs obtained on Data Set-1, Data Set-2, Test Set-1, Test Set-2 and Test Set-3 are shown in figures 3 to 7 respectively.

In the Deep Learning approach, we get better results as compared to either of the traditional approaches. This can be majorly observed in TestSet3 output. The face swap is very smooth is because the PRnet uses full 3D mesh grid of the source face mask and then warps on the destination. Since the traditional approaches use dlib facial landmarks, not all the features of the face are represented in the warp. We can also observe that PRNet fails to detect the face in very few frames, which is not the case with traditional approaches.

REFERENCES

[1] Yao Feng, Fan Wu, Xiaohu Shao, Yanfeng Wang, and Xi Zhou. Joint 3d face reconstruction and dense alignment with position map regression network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 534–551, 2018.