FaceSwap

USING 1 LATE DAY

Ajith Kumar Jayamoorthy
MS in Robotics Engineering
Worcester Polytechnic Institute
ajayamoorthy@wpi.edu

Shiva Kumar Tekumatla
MS in Robotics Engineering
Worcester Polytechnic Institute
stekumatla@wpi.edu

***Abstract*—In this project, we presented three different ways for swapping faces. The first two methods are traditional computer vision methods and the third one is deep-learning method. As part of traditional methods, we presented the detailed explanation of face warping using triangulation and thin plate spline (TPS). For triangulation, we used Delaunay Triangulation method, which is the dual of Voronoi triangulation. In case of deep learning we used the pre-trained MobileNet-V1 to provide landmarks, based on which we implemented face-swap. We used photos of different celebrities and this project members as the data sources and presented the results of face-swap among different inputs. We also considered certain inputs where the results can have artifacts.**

I. Phase 1: Traditional Approach

A traditional method that can be used for swapping faces is shown in figure 1.

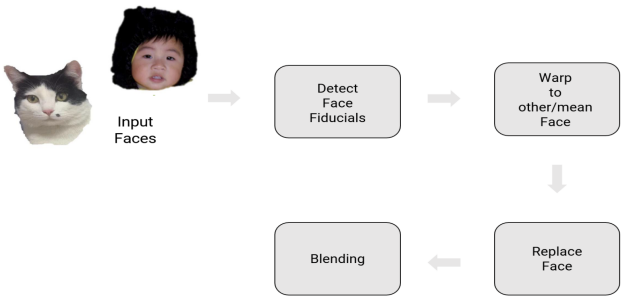

Fig. 1. Overview of face replacement pipeline

In this approach, initially the face fiducials are detected for each input image. Once the fiducials are detected, the faces are warped using either triangulation method or TPS method. After warping the faces are replaced, and then blended. The input images considered in this work are shown in the figures 2, and 3.

A. Facial Landmarks Detection

The first step in the traditional approach is to find facial landmarks. These landmarks are the important points on the face. We can find the one-to-one correspondence between the facial landmarks. This is same as detecting corners in the panorama project. Usage of facial landmarks also reduces the computational complexity. For obtaining the facial landmarks, we used dlib library that is built into OpenCV and python.

Fig. 2. Input Image 1

Fig. 3. Input Image 2

The outputs for landmarks detection of each input images are shown in figures 4 , and 5.

B. Face Warping using Triangulation

We need to use the landmarks obtained above to warp the faces in 3D. But we do not have any 3D information. Hence can we make some assumption about the 2D image to approximate 3D information of the face. One simple way is to triangulate using the facial landmarks as corners and then make the assumption that in each triangle the content is planar and hence the warping between the the triangles in

Fig. 4. Facial Landmarks detection on input image 1

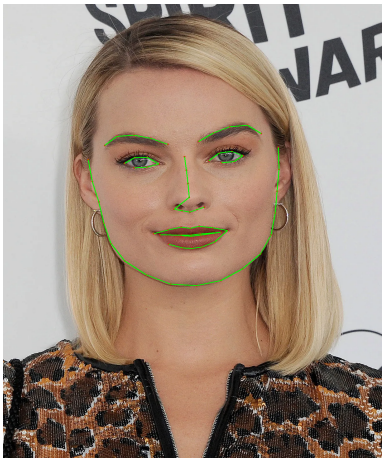

Fig. 5. Facial Landmarks detection on input image 2

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. We can use Delaunay Triangulation for this. Delaunay is an efficient method and can be performed in O(nlogn) time. We want the triangulation to be consistent with the image boundary such that texture regions won't fade into the background while warping. Delaunay Triangulation tries the maximize the smallest angle in each triangle. Output after triangulation for each input image is shown by figures 6, and 7.

From the above triangulation, we obtain the list of co-ordinates of these triangles for each face. This list consist of corresponding triangles in each face. Now for each set of triangles t1 from image 1 and t2 in image 2, we calculate the Barycentric coordinates for each of these triangles respectively. The theory behind the calculation of Barycentric coordinates is given in [1]. After this the corresponding warped co-ordinates of from face2 is calculated for face1 and vice versa. Then the pixel values at the given warped co-ordinates are swapped between the two faces. The resulting image from the process is as shown in figure 8

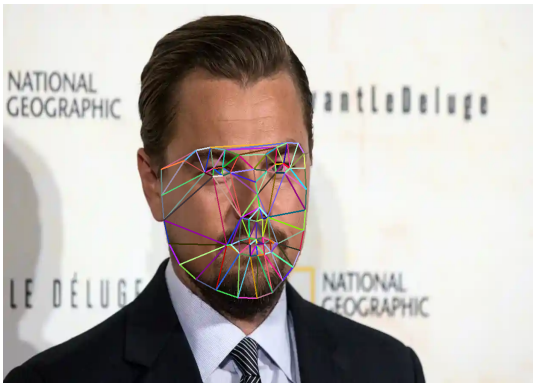

Fig. 6. Triangulation of landmarks on input image 1

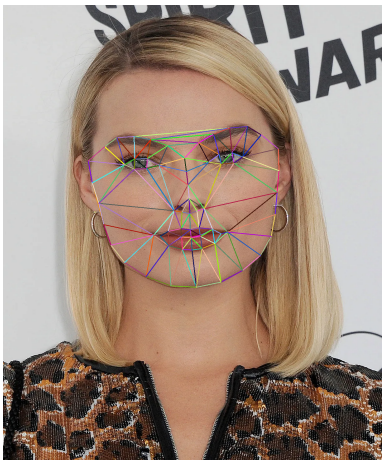

Fig. 7. Triangulation of landmarks on input image 2

Fig. 8. Output of Warping using Delaunay Triangulation

C. Face Warping using Thin Plate Spline

In triangulation, we are doing affine transformation on each trinagle.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. Now, we want to compute a TPS that maps from the feature points in to the corresponding feature points in . Note that we need two splines, one for the x coordinate and one for the y. Imagine a TPS to mathematically model beating a metal plate with a hammer. In the first step, we estimate the parameters of the TPS. In the second step, use these estimated parameters of the TPS models (both x and y directions), transform all pixels in image 2 by the TPS model. Now, read back the pixel value from image 1 directly. The position of the pixels in image 1 is generated by the TPS equation.

After that, all the pixels from face 1 are warped to face 2 and all the pixels are replaced. Of course, simply replacing pixels does not look natural. Selected facial region from the face 2 is shown figure 9. Warped result of face 2 to match face 1 is shown by figure 10. The final blended output is shown by 11.

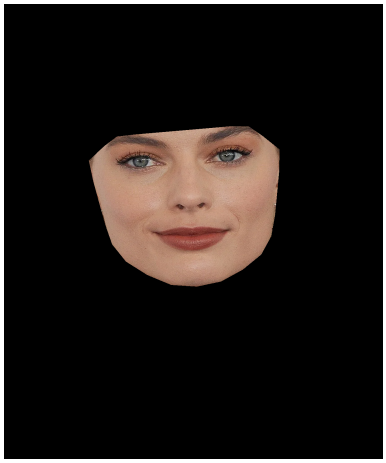

Fig. 9. Selected facial features from face 2

TPS performance is tried on many other inputs as well, and the results of all these are available in Output folder.

II. PHASE II - Deep Learning Approach

In this Phase we are going to use an existing network to get output for an image in 3D space and based on the input we would apply 3D TPS and from the warped points obtained as a output from the 3D TPS, we will work on projecting it into 2D space and then obtaining the corresponding pixel values to be replaced.

A. Data Generation

In the case of Data, we have to first consider two images with a face with full visibility. Then the aspect ratio of the two face images must be made the same as shown in figure 12 and figure 13. After cropping the image, the two faces are combined into one image to be considered as a input for the Neural Network as shown in figure 14.

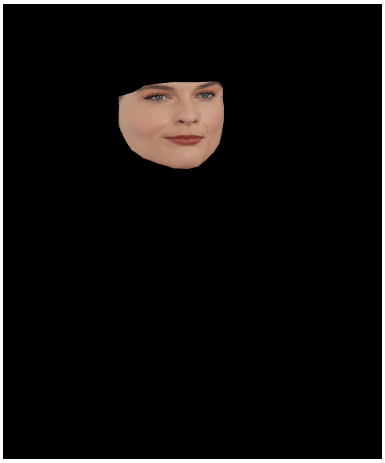

Fig. 10. Warped face 2 to match face 1

Fig. 11. Final Blended Image

Fig. 12. Resized image of face 1

B. Network Architecture and Output

The network used here is a pre-trained model of the MobileNet-V1 structure. The Architecture is as shown in the table 15

Fig. 13. Resized image of face 2

Fig. 14. Final Input Image for the neural network

Type / Stride	Filter Shape	Input Size
Conv / s2	3 × 3 × 3 × 32	224 × 224 × 3
Conv dw / s1	3 × 3 × 32 dw	112 × 112 × 32
Conv / s1	1 × 1 × 32 × 64	112 × 112 × 32
Conv dw / s2	3 × 3 × 64 dw	112 × 112 × 64
Conv / s1	1 × 1 × 64 × 128	56 × 56 × 64
Conv dw / s1	3 × 3 × 128 dw	56 × 56 × 128
Conv / s1	1 × 1 × 128 × 128	56 × 56 × 128
Conv dw / s2	3 × 3 × 128 dw	56 × 56 × 128
Conv / s1	1 × 1 × 128 × 256	28 × 28 × 128
Conv dw / s1	3 × 3 × 256 dw	28 × 28 × 256
Conv / s1	1 × 1 × 256 × 256	28 × 28 × 256
Conv dw / s2	3 × 3 × 256 dw	28 × 28 × 256
Conv / s1	1 × 1 × 256 × 512	14 × 14 × 256
5× Conv dw / s1	3 × 3 × 512 dw	14 × 14 × 512
5× Conv / s1	1 × 1 × 512 × 512	14 × 14 × 512
Conv dw / s2	3 × 3 × 512 dw	14 × 14 × 512
Conv / s1	1 × 1 × 512 × 1024	7 × 7 × 512
Conv dw / s2	3 × 3 × 1024 dw	7 × 7 × 1024
Conv / s1	1 × 1 × 1024 × 1024	7 × 7 × 1024
Avg Pool / s1	Pool 7 × 7	7 × 7 × 1024
FC / s1	1024 × 1000	1 × 1 × 1024
Softmax / s1	Classifier	1 × 1 × 1000

Fig. 15. MobileNet-V1 Architecture

The network was run using the information from reference [2].The output of the Network provides with a 3D point cloud file, an object file with pixel values, a text file with the landmark co-ordinates for each face, Image depth and pose estimation. Using the landmark coordinates we can implement the 3D TPS and obtained the relationship function between the two faces in 3D space.

C. 3D Thin Plate Spline

In case of the 3D spline modification, the implementation would be such that the equation for the Thin Plate Spline can be extended to the 3rd dimension by adding the z component information from the network.

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

The 3D landmark obtained from the Network is as shown in figure 16. After obtaining the 3D TPS, We can extract the

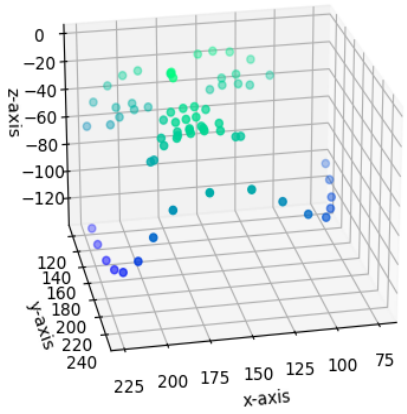

Fig. 16. 3D landmark for Face1

respective pixel positions from the obj file using the TPS. Later the pixel can be swapped and then the image can be project on the x-y 2D plane, with the z as the normal to the image. This could possibly given an swapped face output.

D. Conclusion

From the above results it can be observed that the TPS works much better than the Delaunay Triangulation. Using the Delaunay Triangulation method, the image is not seamlessly integrated. There are some rough spots and the interp2d [3] function had issues with the interpolation of missing values as well. In case of the TPS, the integration is seamless as can be observed from the output. In case of the 3D swapping of faces, the 3D TPS was not working as expected. From my observation, the implementation has some issue which needs to be resolved as a future improvement.

REFERENCES

[1] https://rbe549.github.io/fall2022/proj/p2/
[2] https://github.com/cleardusk/3DDFA
[3] https://het.as.utexas.edu/HET/Software/Scipy/generated/scipy.interpolate.interp2d.html