Project 2: FaceSwap

(Using 1 Late Day)

Rohitkrishna Nambiar (115507944)
University of Maryland
College Park, Maryland 20740
rohit517@umd.edu

Rohith Jayarajan (115458437)
University of Maryland
College Park, Maryland 20740
rohith23@umd.edu

***Abstract*—In this project we explore the application of FaceSwap using different methods such as Triangulation and Thin Plate Spline (TPS) fitting. We use dlib and PRNet for detecting facial landmarks. Different blending methods are studied to provide a seamless face swapping. We then compare the outputs of the traditional approach with the deep learning based approach and present outputs of different test scenarios.**

I. Introduction

II. Face Warping using Triangulation

The traditional method of swapping spaces is discussed in this section. A block diagram of the process is shown in diagram 1.

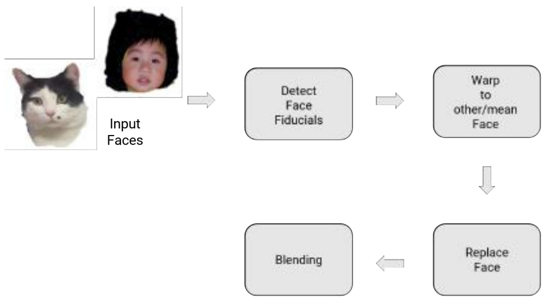


Fig. 1. Overview of the face swapping pipeline.

A. Facial Landmarks detection

Given a face we find the landmarks or important points on the face so as to achieve a one-to-one correspondence between the facial landmarks across different faces. Using facial landmarks reduces the computational complexity of using all the points on the face. Using all the points on the face will give us better results. To detect the landmarks in a given face, the *dlib* library in OpenCV[1] is used. This uses a model pre-trained on many human faces with a ResNet architecture. The output of the landmarks given by *dlib* is a set of 68 landmark points, describing the eyes, eyebrows, nose, lips and jawline of the human in the image. The output of the facial landmarks given by *dlib* can be seen in figure 2

Once we have obtained the facial landmarks, we make the approximation that the 3D face can be approximated using 2D shapes formed by the landmarks. This can be done by triangulating the facial landmarks and then performing an affine transformation between corresponding triangles in two images.

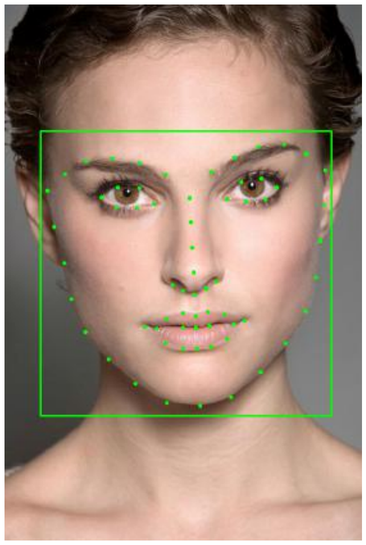


Fig. 2. Facial landmarks detected by *dlib*.

Triangulation involves forming a triaingular mesh over the 2D iamage and the best and efficient way to do so is by using *Delaunay Triangulation* which can be obtained by drawing the dual of the Vornoi diagram. Delaunay Triangulation can be constructed in $\mathcal{O}(n \log n)$ time. Delaunay Triangulation tries the maximize the smallest angle in each triangle.

Also to ensure one-to-one correspondences between the triangulation in both the images, the Delaunay Triangulation is constructed on one image and used to replicate for the other image. The function *cv2.getTriangleList()* in *cv2.Subdiv2D* class of OpenCV is used to implement the Delaunay Triangulation. We use inverse warping so that there is no loss of information while swapping the pixel information of the faces. We warp face in image A (the source image) to B (the destination image).

The output of the Delaunay Triangulation given by *cv2.getTriangleList()* can be seen in figure 3.

The following steps are followed to implement the swapping of face A to B.

1) ***Step 1:*** For each triangle in B, we compute the Barycentric coordinates using equation 1.

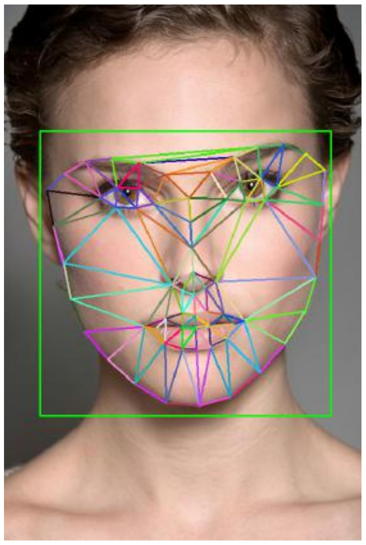


Fig. 3. Delaunay Triangulation given by *cv2.getTriangleList()*.

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

Here the barycentric coordinates are given by $[\alpha\ \beta\ \gamma]^T$. To compute this we take the inverse of the 3×3 matrix. This is done for each triangle using the following equation

$$\begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \mathcal{B}_{\triangle}^{-1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2}$$

Using the values of α, β, γ the point is inside the triangle if $\alpha \in [0,1]$, $\beta \in [0,1]$, $\gamma \in [0,1]$ and $\alpha + \beta + \gamma \in [0,1]$.

2) ***Step 2:***
Using the barycentric coordinates obtained in the previous step, we compute the pixel locations in image A using

$$\begin{bmatrix} x_A \\ y_A \\ z_A \end{bmatrix} = \mathcal{A}_{\triangle} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} \tag{3}$$

where

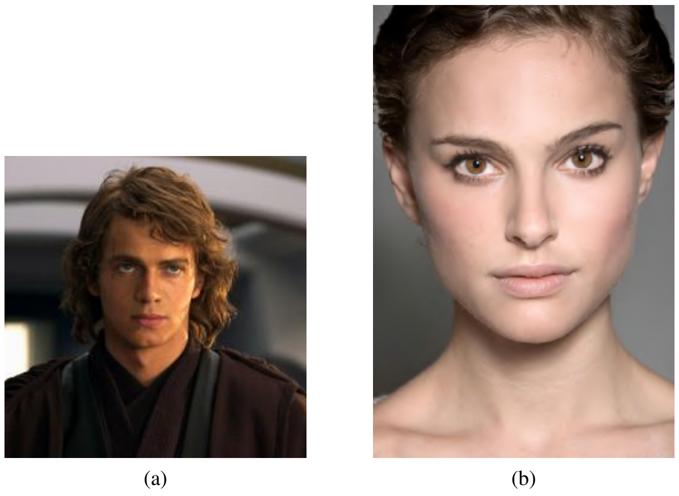
$$\mathcal{A}_{\triangle} = \begin{bmatrix} \mathcal{A}_{a,x} & \mathcal{A}_{b,x} & \mathcal{A}_{c,x} \\ \mathcal{A}_{a,y} & \mathcal{A}_{b,y} & \mathcal{A}_{c,y} \\ 1 & 1 & 1 \end{bmatrix} \tag{4}$$

On obtaining $[x_A\ y_A\ z_A]^T$ we convert it to homogeneous coordinates as follows

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

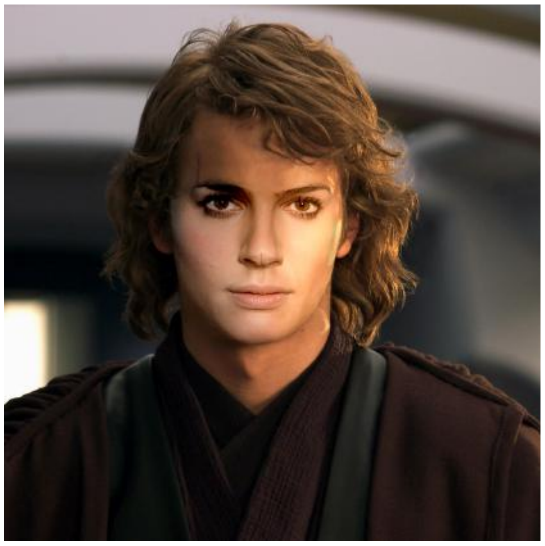
3) ***Step 3:***
In this step, we copy back the pixel value at location (x_A, y_A) from image A back to image B.

Using the above steps, we perform face swapping for the images in 5.



(a) (b)

Fig. 4. Face swap using Triangulation input images



(a)

Fig. 5. Face swap using Triangulation output

III. Face Warping using Thin Plate Spline

The human face is very complex and also has a smooth shape. Using the triangulation method to perform face swapping is certainly not the best way to do so as it assumes the transformation is affine in each triangle. A better solution to face swapping is by using Thin Plate Splines (TPS)[2] which can be used to model shapes that are complex.

To perform face swapping we want to compute a Thin Plate Splines (TPS) that maps from the feature points in the Target Image (B) to the corresponding feature points in Source Image (A).

The equation for the Thin Plate Spline is given by equation 6.

$$f(x,y) = a_! + a_x x + a_y y + \sum_{i=1}^{p} w_i U(\|(x_i, y_i) - (x,y)\|_{L_1}) \tag{6}$$

where,

$$U(r) = r^2 log(r^2)$$

We perform an inverse warping here instead of forward warping so that there are no holes/loss of information in the process. This is done by finding parameters of a Thin Plate Spline which maps from B to A. The following steps are performed to find the parameters of the Thin Plate Spline equation

1) ***Step 1:*** The solution of the Thin Plate Spline model requires solving equation 7.

$$\begin{bmatrix} K & P \\ P^T & 0 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_p \\ a_x \\ a_y \\ a_1 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_p \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{7}$$

where
$K_{i,j} = U(\|(x_i, y_i) - (x,y)\|_{L_1}), v_i = f(x_i, y_i) and\, the$ ith row of the matrix P is $(x_i, y_i, 1)$.
The matrix K is of size $p \times p$ and matrix P is of size $p \times 3$. For a stable solution, the solution of 7 is given by 8

$$\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_p \\ a_x \\ a_y \\ a_1 \end{bmatrix} = \left(\begin{bmatrix} K & P \\ P^T & 0 \end{bmatrix} + \lambda I(p+3, p+3) \right)^{-1} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_p \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{8}$$

where $\lambda \approx 0$ and I is an identity matrix so that the matrix can be invertible.

2) ***Step 2:*** After the parameters of the Thin Plate Spline have been computed, all pixels in image B are transformed by the TPS model. After this transformation has been done, the pixel value is read back from image A directly. The position of the pixels in A is generated by solving the TPS equation twice, once for the x coordinates and once for the y coordinates.

IV. Replace Face

The process of replacing faces is fairly simple. All the pixels from face in image A are warped and fitted to the face in image B and pixels are replaced. But just doing this will not give us a convincing and seamless face swapping as changes in color, lighting and edges will produce unwanted artifacts.

A replaced face using the pipelines discussed without blending is shown in figure 6

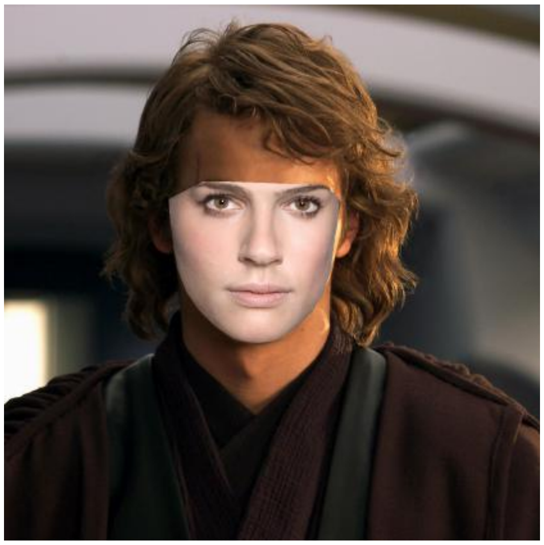


Fig. 6. Output of face replacement. The difference in color, lighting and edges give unwanted artifacts.

V. Blending

To reduce and to eliminate the effects of artifacts to a good extent, we use a variant of Poisson blending to blend the warped face onto the target face. This is performed using OpencCV's inbuilt function *cv2.seamlessClone(src, dst, mask, center, flags)* where *src* is the source image, *dst* is the destination image, *mask* is the warped destination image, *center* is the position where the mask is to be placed on the source image and *flags* specify the type of cloning.

A replace face with blending using the pipelines discussed is shown in figure 7.

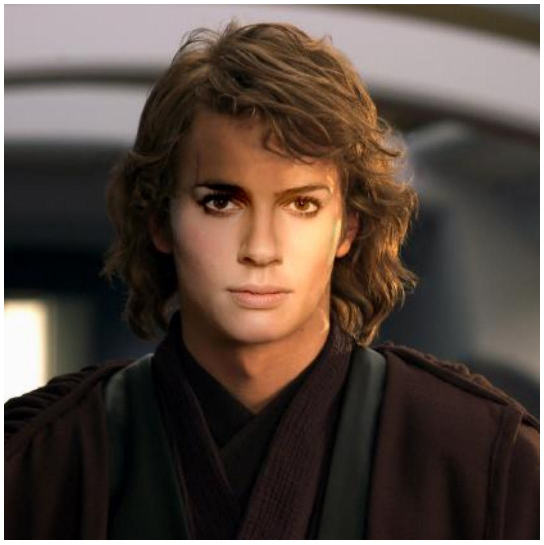


Fig. 7. Output of face swapping using TPS with poisson blending.

VI. Deep Learning approach

In this approach, we use an off-the-shelf deep learning model called PRNet[3] for fiducial detections. PRNet gives the same number of fiducial landmarks as dlib along with other artifacts such as depth and meshing. The architecture is given in fig. 8

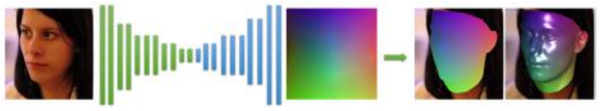


Fig. 8. PRNet architecture

Some of the outputs given by the PRNet can be visualized below (The images are channel reversed ie. in BGR).

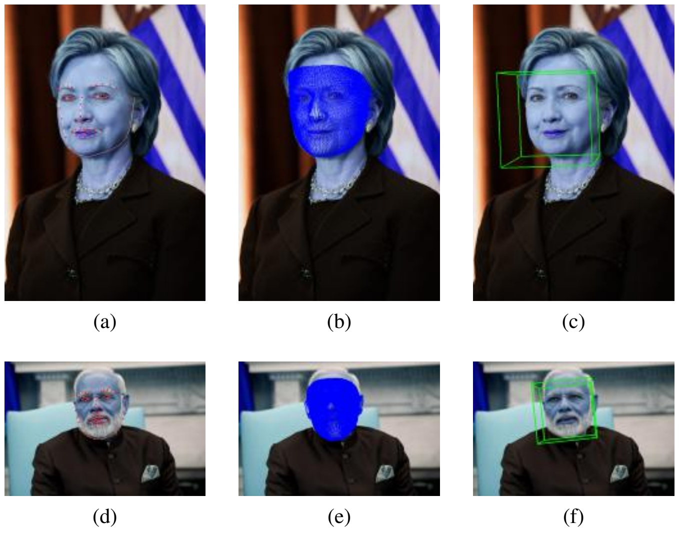


Fig. 9. PRNet outputs

Using the same two images we used in triangulation, we run face swapping with blending using the deep learning model. The output is shown in fig. 10.

VII. Experiments

To test our implementation, we considered two scenarios where we swap a face in the video with an image and the other where we swap two faces in a video. We created our two test sets with images shown in Fig.11

A. Traditional Approach

The face swap outputs of our test sets from triangulation and thin plate spline methods is shown in Fig.12, Fig.13, Fig.14, Fig.15.

B. Deep Learning Approach Output

We used PRNet to get the fiducial landmarks and used TPS to swap the faces. We also used the swapping implementation given in the PRNet code to compare our results. We notice that PRNet gives a better output in blending as it blends the forehead as well. This can be seen in Fig.16 and Fig.17.

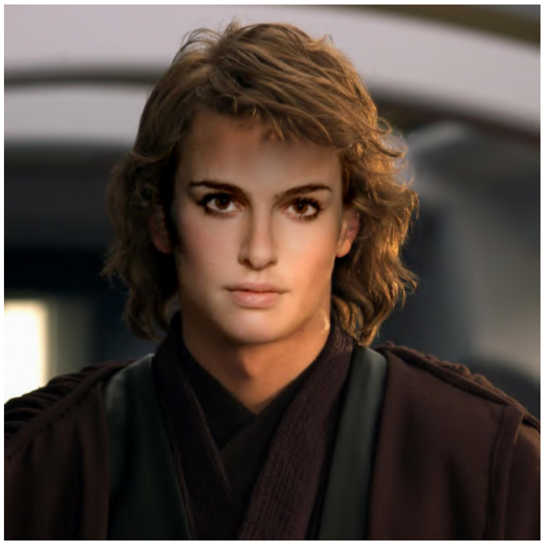


Fig. 10. Face swapping using PRNet

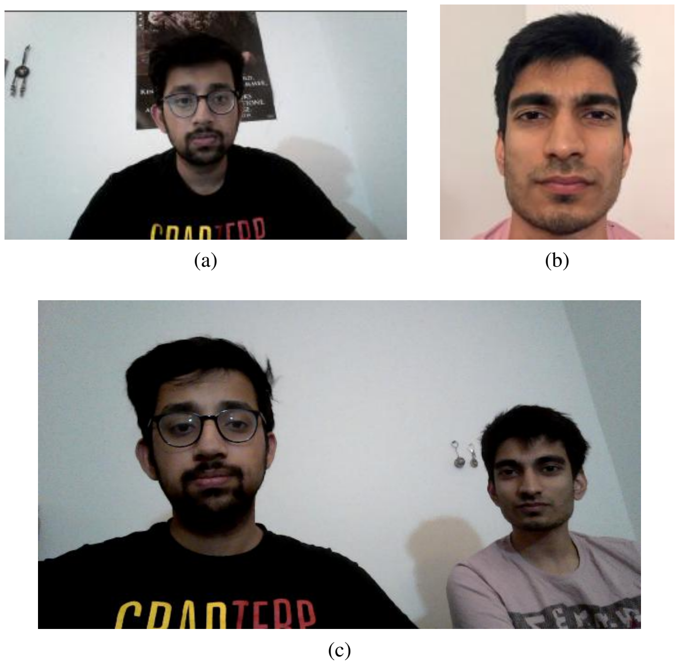


Fig. 11. Test inputs. a) Swap face in video with image. b) Swap faces in video.

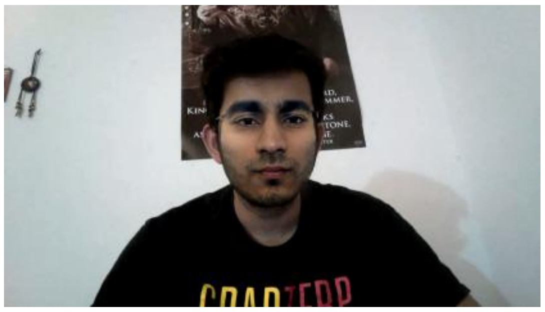


Fig. 12. Face swap in video with image using Triangulation method.

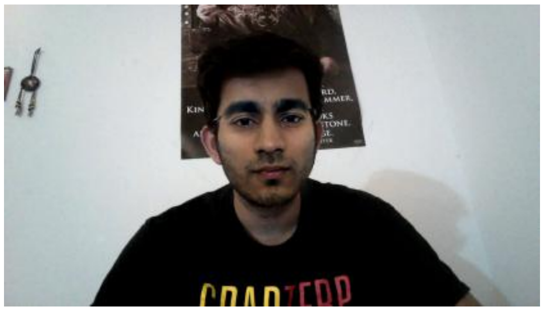


Fig. 13. Face swap in video with image using Thin Plate Spline.

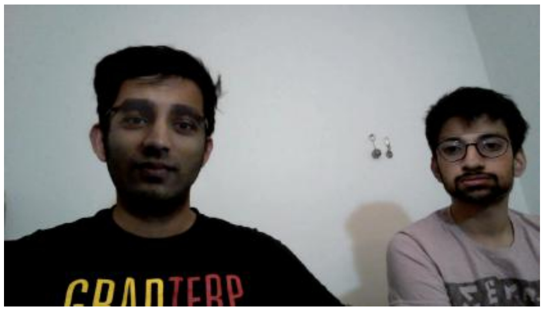


Fig. 14. Face swap in video with image using Triangulation method.

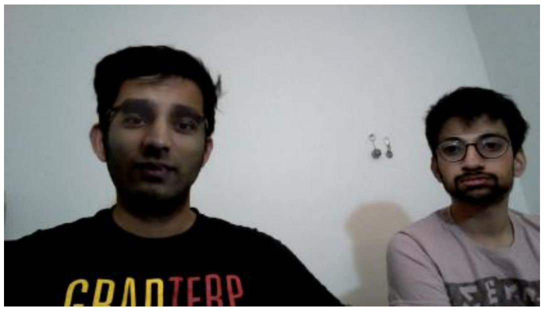


Fig. 15. Face swap in video with image using Thin Plate Spline.

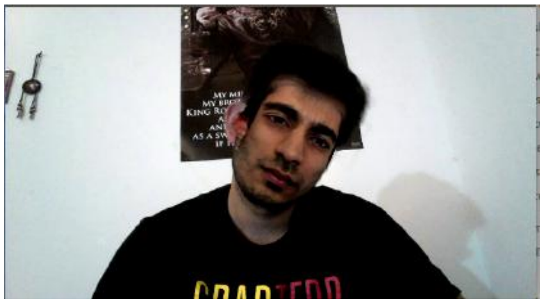


Fig. 16. Face swap in video with image using PRNet.

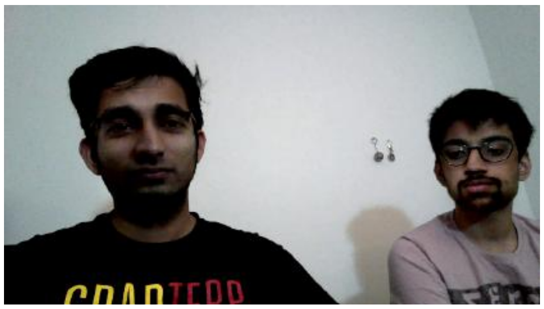


Fig. 17. Face swap within video using PRNet.

VIII. Results

1) The least effective method for face swap application was the triangulation method. In the triangulation method, the 3D face is approximated using 2D shapes formed by the landmarks. This was done by triangulating the facial landmarks and then performing an affine transformation between corresponding triangles in two images. This use of a triangular 2D mesh over the face image wasn't successful in capturing all the information of the complex structure involved. Its run time complexity was $\mathcal{O}(n \log n)$ which is better than that of thin plate spline method.
2) The thin plate splines method was a significant improvement over the triangulation method. Using this enabled to effectively capture the information of the complex structure of the face. Outputs of the warp from the thin plate spline methods were better than triangulation method as expected. Its run time complexity however was $\mathcal{O}(n^3)$ due to the matrix inversion involved which is worse than triangulation method.
3) PRNet gives dense meshes, depth outputs and good results even when the face is facing at an angle not parallel to the camera. This helps for use in other application of face morphology. Also since it gives points on the forehead, the blending output is more visually pleasing.

IX. Conclusion

Thus we have implemented Triangulation and Thin Plate Spline methods for face warping, studied different blending methods and compared traditional method of implementation with deep learning based methods.

Acknowledgment

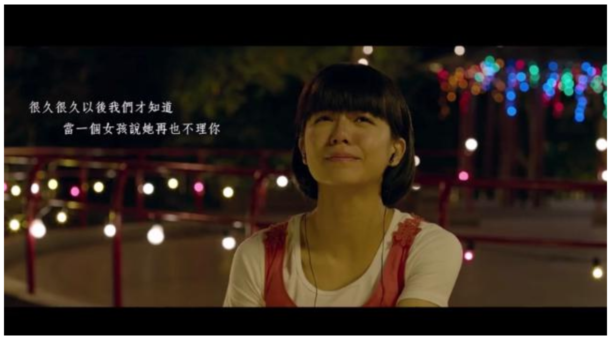
We would like to thank Prof. Yiannis and the TA's Nitin and Chahat for their help with the project.

References

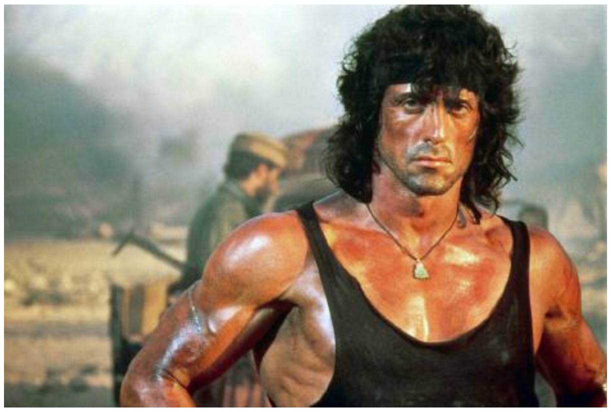
[1] Itseez, "Open source computer vision library," https://github.com/itseez/opencv, 2015.

[2] F. L. Bookstein, "Principal warps: Thin-plate splines and the decomposition of deformations," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 11, no. 6, pp. 567–585, 1989.

[3] Y. Feng, F. Wu, X. Shao, Y. Wang, and X. Zhou, "Joint 3d face reconstruction and dense alignment with position map regression network," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 534–551.



(a)



(b)

Fig. 18. Test set 1.

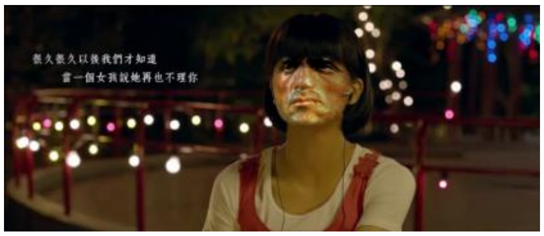


Fig. 19. Face swap in test video 1 with image using Triangulation method.

Appendix A
Test Set Outputs

Below are the results for the test examples.

A. *Test Set 1*

The face swap outputs of the given test sets from triangulation and thin plate spline methods is shown in Fig.19 and Fig.20.

The input image and video frame face sequence can be seen below in Fig. 18

B. *Test Set 2*

The input image and video frame face sequence can be seen below in Fig. 22. The face swap outputs of the given test sets from triangulation and thin plate spline methods is shown in Fig.23 and Fig.24.

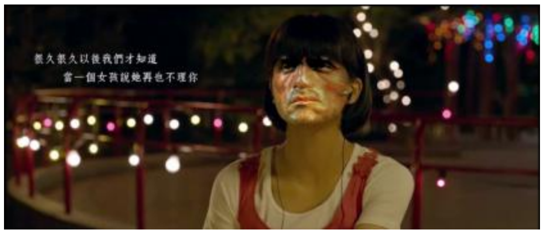


Fig. 20. Face swap in test video 1 with image using Thin Plate Spline.

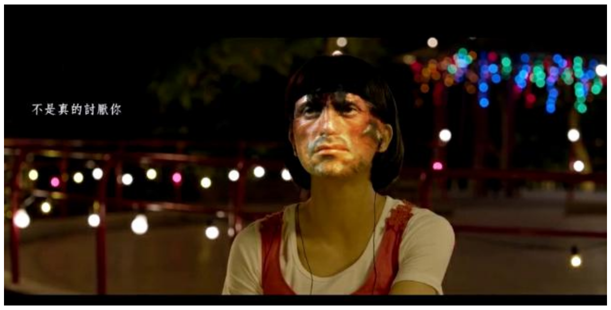
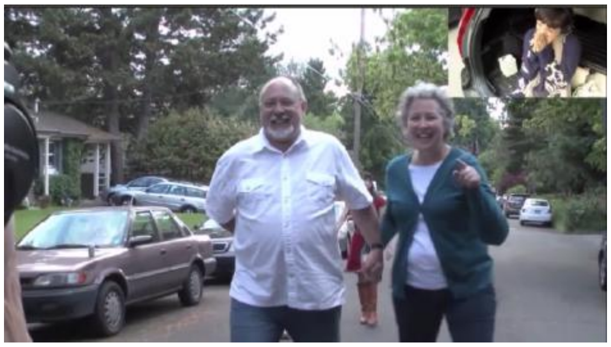


Fig. 21. Face swap using PRNet and TPS for test set 1.



(a)

Fig. 22. Test set 2.

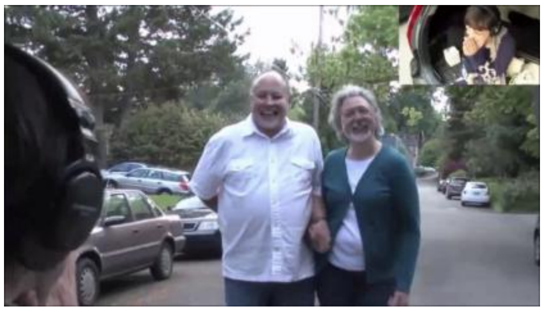


Fig. 23. Face swap in test video 2 with image using Triangulation method.

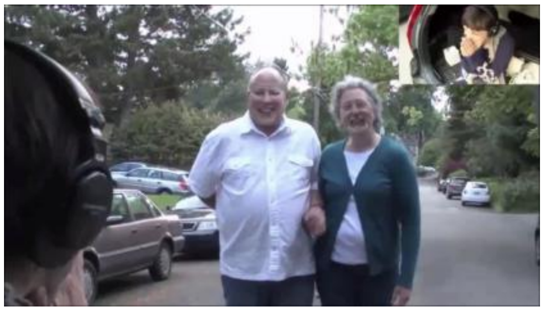


Fig. 24. Face swap in test video 2 with image using Thin Plate Spline.

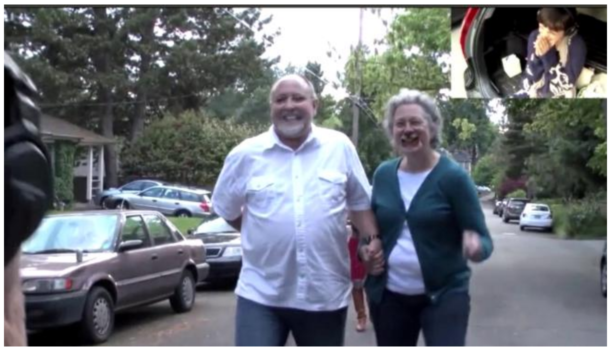


Fig. 25. Face swap using PRNet and TPS for test set 2.

C. Test Set 3

The input image and video frame face sequence can be seen below in Fig. 26. The face swap outputs of the given test sets from triangulation and thin plate spline methods is shown in Fig.27 and Fig.28.

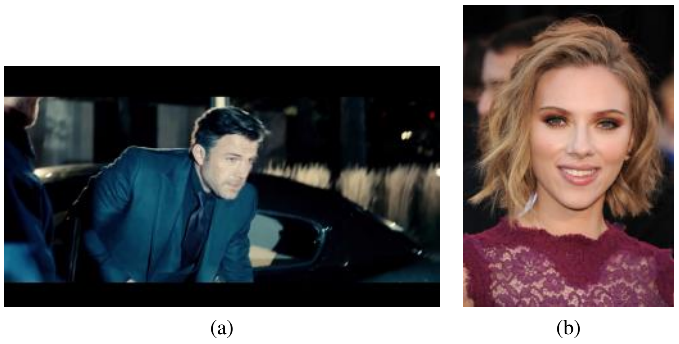


Fig. 26. Test set 2.

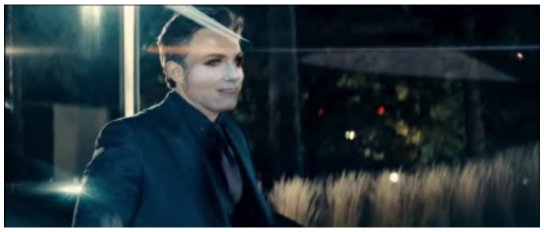


Fig. 27. Face swap in test video 3 with image using Triangulation method.

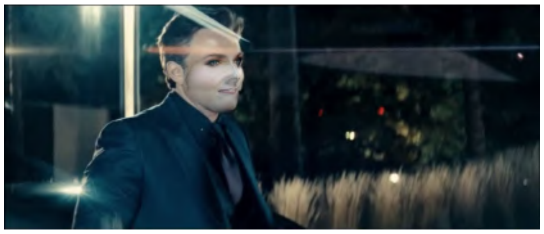


Fig. 28. Face swap in test video 3 with image using Thin Plate Spline.

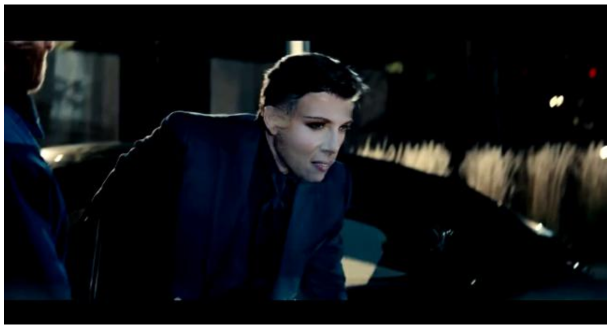


Fig. 29. Face swap using PRNet and TPS for test set 3.