Project 2
Faceswap

Sakshi Kakde
M. Eng. Robotics
University of Maryland
College Park, MD, 20742
Email: sakshi@umd.edu

Gokul Hari
M. Eng. Robotics
University of Maryland
College Park, MD, 20742
Email: hgokul@umd.edu

***Abstract*—The report describes in brief our solutions for the project 2. The report is divided into two sections. First section explores the traditional approach for face swapping. Second section describes the implementation of a supervised and an unsupervised deep learning approach of estimating homography between synthetically generated data.**

I. Phase 1: Traditional Approach

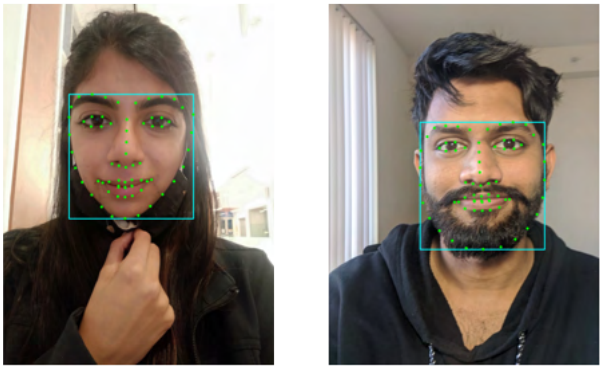


Fig. 1. Visualizing Face fiducial markers and bounding boxes

This section describes in brief the steps involved for swapping faces using traditional method. Refer figure 2 for an overview of the process.

A. Facial Landmarks Detection

To perform Face swap, we need to first detect a human face in a given image frame. In order to detect a human face, we utilise a Histogram of Oriented Gradients (hog) + Support Vector machine (svm) based face detector, that is built in the dlib library, for face detection [4], which is developed based on the research done by [5] and [6]. Having estimated the bounding box where the human face is located, we identify 68 face fiducial markers using the ensemble of regression trees algorithm that is built into dlib library along with the pretrained models. These 68 face fiducial markers/ key points , denoted as P_{face} that provide information about the facial structure. Moreover, the orderly arrangement of the 68 face markers is preserved across different detected faces. Having computed the face fiducial markers, we can obtain the human faces from the image frames to perform swapping. The detected facial bounding box and the face fiducial markers P_{face} are shown in figure 1

B. Face Warping using Triangulation

One of the methods to superimpose a face from the source to a destination face, is by performing Delauney triangulation of the face marker points P_{face}.

For a given set of discrete points P_{face} in a general position, Delaunay triangulation DT(P_{face}) is a triangulation such that no point in P_{face} is inside the circumcircle of any triangles in DT(P_{face}). Delaunay triangulations maximize the minimum angle of all the angles of the triangles in the triangulation [7].

For a given image of a face with it's respective face markers P_{face}, we obtain Delauney triangles, using the subdiv2d class in opencv and the getTriangleList function. We select only the triangles where all the three vertices belong within P_{face}, the 68 face markers. We optimized the selection process using a dictionary of 68 P_{face} and their corresponding indices, instead of running nested for loops. This set of triangles obtained is denoted as DT(P_{face}).

To maintain the order of triangle correspondences between two images, we first obtain and select the vertices of triangles using subdiv2D only for the destination face as DT(P_{face2}). We also note the indices, denoted as $idxs_{DT}$, of these vertices with respect to P_{faces2}. Since the order of correspondence of P_{faces1} and P_{faces2} is maintained, we pick the only the points in P_{faces1} according to the indices $idxs_{DT}$ to obtain the vertices of the Delaunay triangles of source face denoted as DT(P_{face1}). The Delaunay triangulated faces are shown in figure 3.

To perform the swapping, the facial image pixels inside every triangle in DT(P_{face1}) is warped to the corresponding triangle in DT(P_{face2}), to reshape the source face according to the facial structure of the destination face as shown in figure 5. We compute barycentric coordinates to ensure that the pixel coordinates to be warped lie inside the triangle, and perform inverse warping between the source triangle and a destination triangle to perform a hole-free warping.

C. Face Warping using Thin Plate Spline

In this method, we want to get smooth functions that map the pixel co-ordinates for the source image to the destination

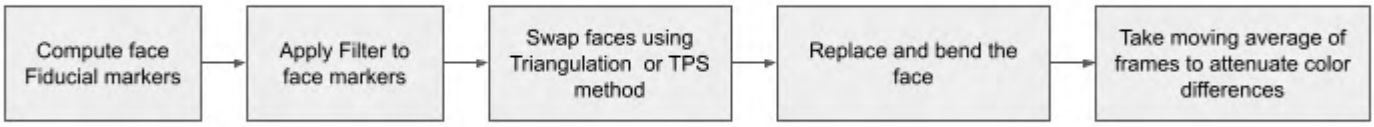


Fig. 2. Block diagram for face swap using traditional approach

Fig. 3. Visualizing Delaunay triangulation

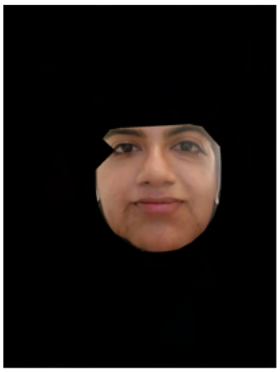


Fig. 4. Source Face restructured according to Destination Face - Triangulation method

image. These functions are given by:

$$f_{x'}(x,y) = a_1 + a_x x + a_y y + \sum_{i=1}^{N} w_i U(\|(xi,yi) - (x,y)\|$$

$$f_{y'}(x,y) = a_1 + a_x x + a_y y + \sum_{i=1}^{N} w_i U(\|(xi,yi) - (x,y)\|$$

Where, x and y are the pixel co-ordinates from the source image, x_i and y_i are the control points(face features), $f_{x'}(x,y)$ represents the x co-ordinate warped point, i.e x' and $f_{y'}(x,y)$ represents the y co-ordinate warped point, i.e y'.
Also, $U(r) = r^2 log(r)$
We need to find the terms a_1, a_x, a_y and w_i

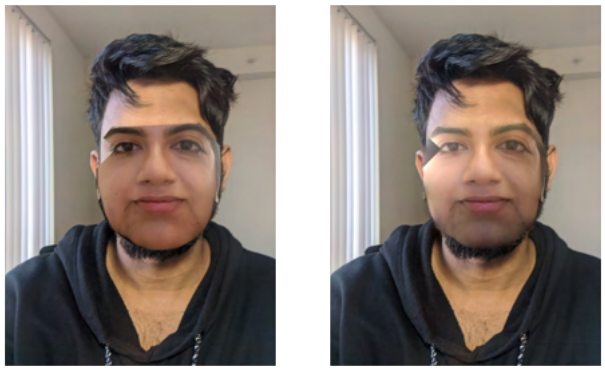


Fig. 5. Left: Face Swap without blending; Right: FaceSwap with poisson blending

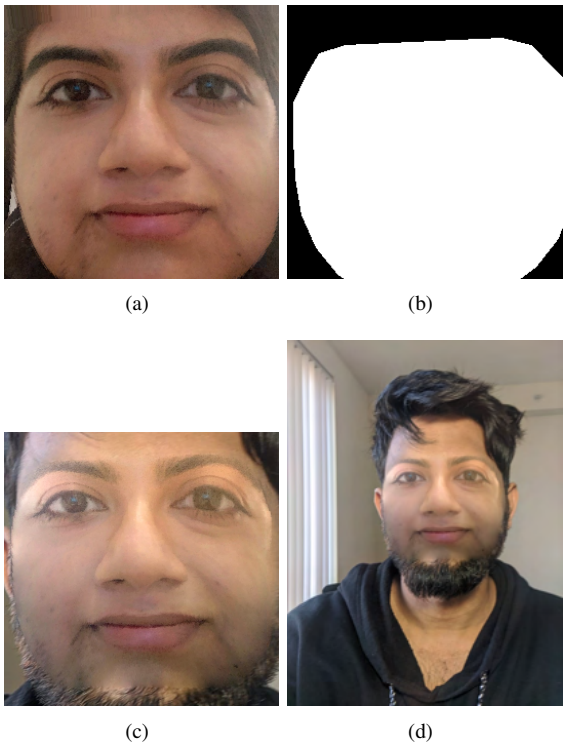
where $i \in [0, N)$, with N being the number of feature points. [3]. We have total N + 3 unknowns for each equation, i.e for getting x' and y', which can be solved using the method described in [3]. The steps followed for warping image using TPS are:

1) Using the facial features obtained from section I-A, get the bounding boxes and crop both faces.
2) Depending on the bounding box location, shift the landmarks location as well.
3) Using these shifted point, use the above method to get the parameters for both the functions $f_{x'}(x,y)$ and $f_{y'}(x,y)$.
4) Warp the cropped source image using the estimated Thin-Plate function.
5) Using the location of bounding box from step 1, paste the warped cropped source image on the destination image.
6) Blend the images as discussed in section I-D.

The result of warped image using TPS is shown in figure 6.

D. Blending

The source face, once reshaped using one of the above warping methods, were blended in the location of the destination image frame using Poisson blending. To perform Poisson blending, we need compute a mask of the face in destination image, where the region of interest (ROI) of the face has white pixels and the rest of the frame is black. We compute the



(a) (b) (c) (d)

Fig. 6. Warped image using TPS(a), binary face mask(b), face swapped on cropped image(c), final face swapped output(b)

convex Hull of P_{face} and fill in the ROI with white pixels to achieve this.

E. Motion Filtering

In this section, we will discuss in brief the approaches we implemented to smoothen out the face swap results for videos.

1) Undetected face box
The face detector provided in the dlib library needs a face bounding box as an input. Sometimes, there is no bounding box detected. To handle this error, we keep a track of the current detected bounding box and the difference between the current and past bounding box. If there is no detection, we predict the current bounding box using the past and the previous difference in positions of bounding boxes.
2) Moving Average Filter:
We implemented a moving average filter for the face features points obtained from section I-A. For a chosen window size N, the filter will take the average of the latest N values. Refer figure 7 for outputs after applying each filter.
3) Check on δx, δy:
We observed that at times, the location of feature points jumped by a value higher than expected. To avoid these jumped, we calculated the difference in current x, y, and previous x, y and capped the values higher than a particular threshold. We then updated the current position as $x+\delta x$ and $y+\delta y$. However, the motion was still noisy. Further, we implemented a moving average filter on δx and δy values to smoothen out the transition. Refer figure 7 for the filtered results.

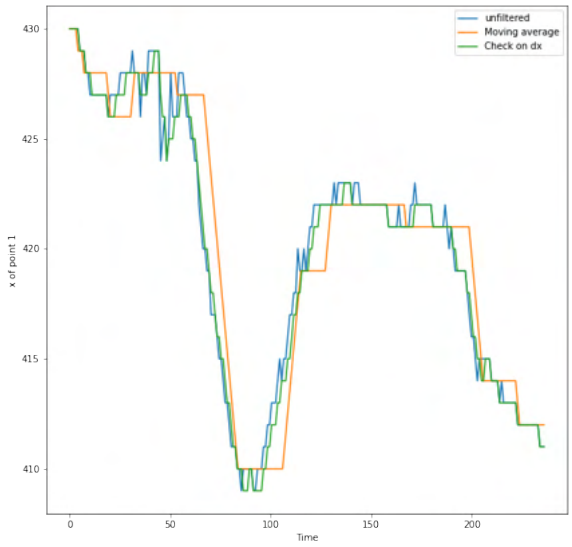


Fig. 7. x values of point 1 of detected face marks

4) Frame average
Though the boundary of the face was comparatively stable, there are color fluctuations within the warped face, especially with the TPS method. To make these changes less prominent, we implemented another moving average filter to take an average of the last N frames.

F. Analysis

1) Warping method: For an image, the thin plate spline method performed better than the triangulation method. However, for videos, we think that results from the triangulation method are better as they are more stable as compared to the results from the thin-plate spline. When we warp the frames from a video using the TPS method, it can be imagined as reshaping an image using a mold-like method. So, even if the boundaries for the detected face move by a bit, the internal structure of the whole face can get affected. This is not the case with the triangulation method, since we warp each triangle separately. We think this is the reason for getting a sort of a 'wavy' motion within the warped images obtained from the TPS method. We tried stabilizing the detected face features as discussed in section I-E, but the 'wavy' motion persisted with the TPS method.

2) Motion filtering: The motion filtering as discussed in section I-E helped us obtain better results for the videos with lesser movements. For the Test3 video, we applied just the moving average filter. Since the motion is very fast, if we apply a check on δx, δy values, we get wrong predicted points.

Putting a check on δx, δy is equivalent to putting a check on V_x, V_y and as the motion is not in a single direction, this approach will not work. An ideal solution might be to use an EKF, but due to limited time, we could not give it a try.

3) No face detected: We observed that sometimes, no face is detected in the videos. In such cases, we used the method 1 from section I-E. However, for the Test2 video, sometimes a wrong face box is predicted and we get wrong warped images.

II. Phase 2: Deep Learning Approach

In this section, we implement a deep learning network proposed by Feng et al. [8], in a research work titled "Joint 3D Face Reconstruction and Dense Alignment with Position Map Regression Network". In this research, they compute a 2D representation of the face called UV position map, which records the 3D shape of a complete face in UV space, then train a simple Convolutional Neural Network to regress it from a single 2D image. We utilised code from github provided by the authors for our Face Swap pipeline.

The implementation also involved a dlib model to detect faces, but it was a CNN based detector with higher accuracy but slower processing time. In cases when no faces were detected by the face detector, we utilised the previous detected face poses, similar to the procedure suggested in traditional approach. We did not incorporate any motion filtering in this method.

III. Results

We evaluated the performance of both the traditional methods and the Deep Learning methods. We also presented the results for swapping two faces in a single frame in Test2.mp4 from the test set and Data2.mp4 from our recorded videos. We have attached the output videos of all the test videos here

(a)

(b)

Fig. 8. Face swapped results for test1 using triangulation

(a)

(b)

Fig. 9. Face swapped results for test1 using TPS

(a)

(b)

Fig. 10. Face swapped results for test1 using PRNet

References

[1] https://www.youtube.com/watch?v=WDM_BGijXEc
[2] https://mathworld.wolfram.com/ThinPlateSpline.html
[3] https://khanhha.github.io/posts/ThinPlateSplinesWarping/thinplatespline
[4] http://dlib.net/imaging.htmlget_frontal_face_detector
[5] Histograms of Oriented Gradients for Human Detection by Navneet Dalal and Bill Triggs, CVPR 2005
[6] Object Detection with Discriminatively Trained Part Based Models by P. Felzenszwalb, R. Girshick, D. McAllester, D. Ramanan IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 32, No. 9, Sep. 2010
[7] https://en.wikipedia.org/wiki/Delaunay_triangulation
[8] Feng Y., Wu F., Shao X., Wang Y., Zhou X. (2018) Joint 3D Face Reconstruction and Dense Alignment with Position Map Regression Network. In: Ferrari V., Hebert M., Sminchisescu C., Weiss Y. (eds) Computer Vision – ECCV 2018. ECCV 2018. Lecture Notes in Computer Science, vol 11218. Springer, Cham. https://doi.org/10.1007/978-3-030-01264-9_33

(a)

(b)

Fig. 11. Face swapped results for test3 using triangulation

(a)

(b)

Fig. 12. Face swapped results for test2 using TPS

(a)

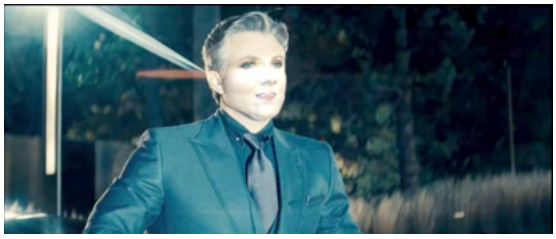
(b)

Fig. 13. Face swapped results for test2 using PRNet

(a)

(b)

Fig. 14. Face swapped results for test3 using triangulation



(a)

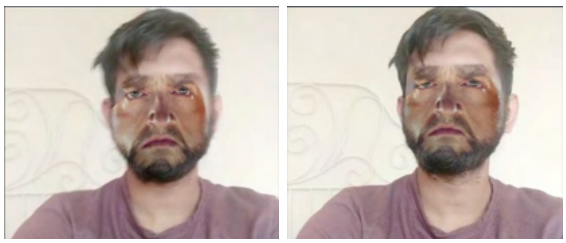
(b)

Fig. 15. Face swapped results for test3 using TPS

(a)

(b)

Fig. 16. Face swapped results for test3 using PRNet

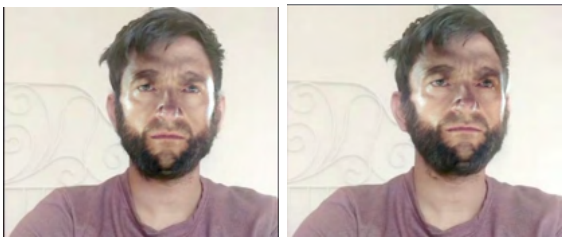


(a) (b)

Fig. 17. Face swapped results for Data1 using triangulation

(a) (b)

Fig. 18. Face swapped results for Data1 using TPS



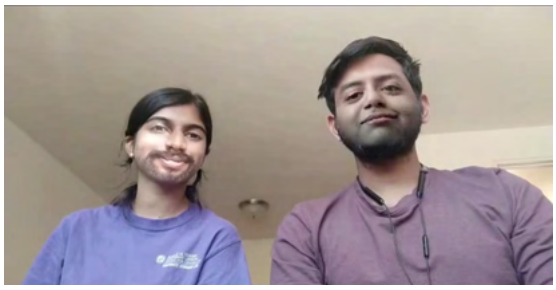
(a) (b)

Fig. 19. Face swapped results for Data1 using PRNet

(a)

(b)

Fig. 20. Face swapped results for Data2 using triangulation



(a)

(b)

Fig. 21. Face swapped results for Data2 using TPS

(a)

(b)

Fig. 22. Face swapped results for Data2 using PRNet