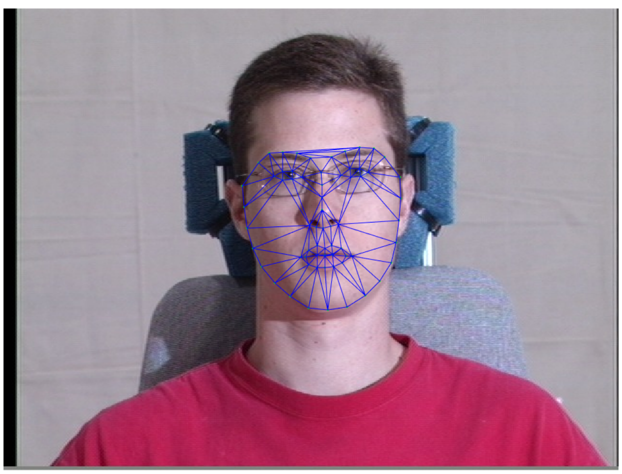
CMSC 733 Project 2

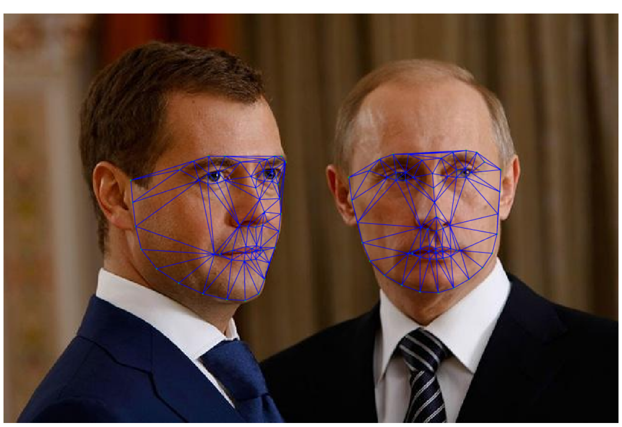
Wei-An Lin
The Department of Electrical and Computer Engineering
University of Maryland
Email: walin@terpmail.umd.edu

I. Triangularization

After extracting landmarks from the target face, we use 'delaunay' to triangularize the set of points. We define the target region to be transformed as the convex hull formed by all the landmarks. To reduce boundary artifact, we apply 'imerode' to slightly shrink the size of the chosen region. Example result for triangularization is shown in Fig. 1.



(a)



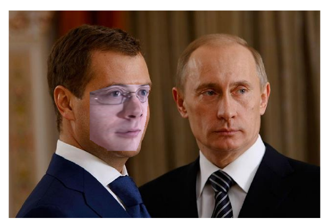
(b)

Fig. 1: Sample results for triangulariztion.

For each point in the target region, we decide which triangle it belongs to and compute the corresponding barycentric coordinate. The same triangularization is defined on the source face image with different landmark points. The inverse transformed Cartesian coordinates can be obtained easily. Results can be seen from Fig. 2.

(a)

(b)



(c)

(d)

Fig. 2: Sample results for face swap using triangularization.

II. Thin-plate Spline

Denote the image coordinates of the source image and the target image as (u, v) and (x, y) respectively. We aim at fitting two splines:

$$u = a_{u,1} + a_{u,x}x + a_{u,y}y + \sum_{i=1}^{p} w_{u,i} U(\|(x_i, y_i) - (x, y)\|), \tag{1}$$

$$v = a_{v,1} + a_{v,x}x + a_{v,y}y + \sum_{i=1}^{p} w_{v,i} U(\|(x_i, y_i) - (x, y)\|). \tag{2}$$

As previously mentioned, we choose the target region as the shrinked convex hull defined by the detected landmarks. The result is shown in Fig. 3.

(a)

(b)

(c)

(d)

Fig. 3: Sample results for face swap using TPS.

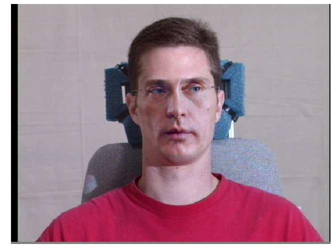
III. Blending

To blend two regions come from different domains, we use the Poisson Editing technique (The code is provided by https://www.mathworks.com/matlabcentral/fileexchange fast-seamless-image-cloning-by-modified-poisson-equation.). The results from both triangularization and TPS are shown in Fig. 4 and Fig. 5. Based on the figures, we observe that triangularization keeps relative distance pretty well. For example, the glasses in Fig. 4c have less distortion then the one in Fig. 5c. On the other hand, TPS produces better curved surfaces. The left side of the nose in 5a has better curve than the one in Fig. 4a. In terms of speed, TPS is faster in our case since triangularization requires creating triangles and pixel assignment followed by change of basis transforms, while TPS requires only basis transforms.

IV. Motion Filtering

For motion filtering, we interpolate three frames between two original consecutive frames, which is one form of low-pass filtering. Interpolation smoothed out the motion, but doesn't seem to be

(a)

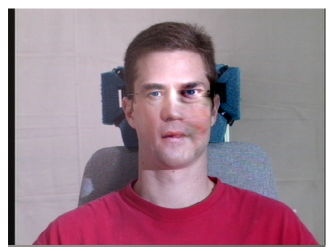


(b)

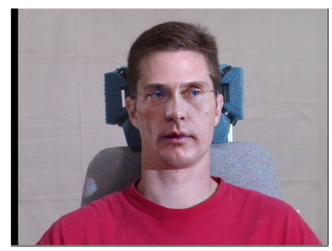
(c)

(d)

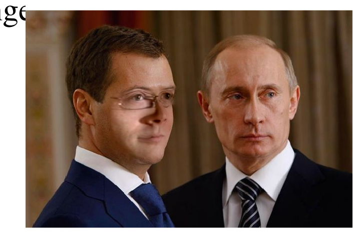
Fig. 4: Blending results for triangularization.



(a)



(b)



(c)

(d)

Fig. 5: Blending results for TPS.

stabilized. There are still some abnormal distortion in the expression. It is possible that for each pixel location in the face region, larger temporal duration should be considered while averaging.

V. Face Swap for Video and Results for the TestSet

To swap faces in videos, some problem arises. The most severe one occurred in this project is that the face detector may not be robust enough, and may fail for some frames. In this case, we simply put the original frame and perform motion filtering. For example, some face images in Test1.mp4 are not detected, even though the face is extremely frontal and clear. Test3.mp4 contains frames taken in dark environment, which fails the face detector. Fig. 6 shows some sample results for the chosen videos. It can be observed that blending in Fig. 6a is not good enough even though the chins are successfully swapped. It may result from not sufficient erosion in the face region. In Fig. 6b, we can see the expression is swapped nicely. Fig. 7 shows some sample results for the test set.

(a)

(b)

Fig. 6: Sample results for face swap in video.

(a)

(b)

(c)

Fig. 7: Sample results for face swap in video.