RBE549 Project2 FaceSwap

Haoying Zhou
Department of Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA, 01609
Email: hzhou6@wpi.edu

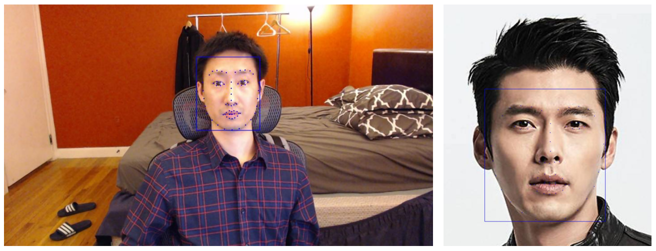
Zhentian Qian
Department of Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA, 01609
Email: zqian@wpi.edu

I. Phase 1: Traditional Approach

A. *Facial Landmarks detection*

The facial landmarks are detected using the dlib library. We first use a pretrained CNN based face detector offered by dlib. The CNN model is much more accurate than the HOG based model, but takes much more computational power to run.

64 facial landmarks are then detected in the bounding box detected by the face detector, as shown in Figure 1 for data 1 and Figure 2 for data 2.



(a) face 1 landmarks (b) face 2 landmarks

Fig. 1: Output of dlib for facial landmarks detection for data 1. Blue landmarks are overlayed on the input image.

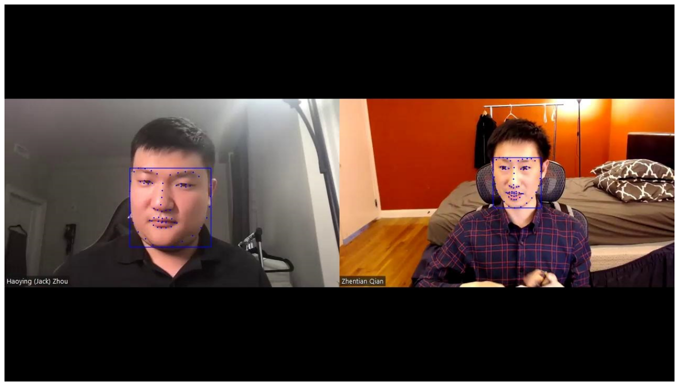
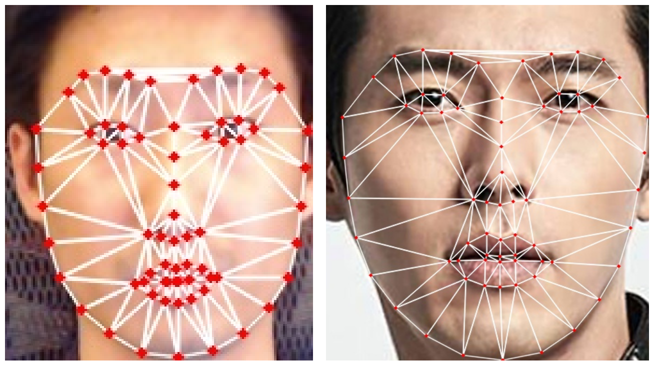


Fig. 2: Output of dlib for facial landmarks detection for data 2. Blue landmarks are overlayed on the input image.

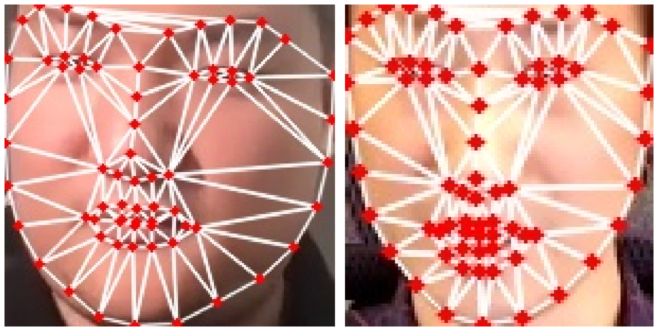
B. *Face Warping using Triangulation and Thin Plate Spline*

Delaunay Triangulation are performed on the detected face regions, with the 68 facial landmarks as the vertices, as shown in Figure 3 for data 1 and Figure 4 for data 2.



(a) face 1 (Zhentian Qian) triangulation (b) face 2 (Hyun Bin) triangulation

Fig. 3: Triangulation on two faces we want to swap for data 1.

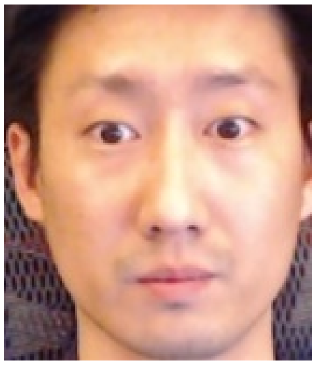


(a) face 1 (Haoying Zhou) triangulation (b) face 2 (Zhentian Qian) triangulation

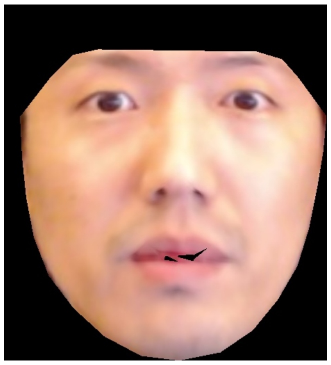
Fig. 4: Triangulation on two faces we want to swap for data 2.

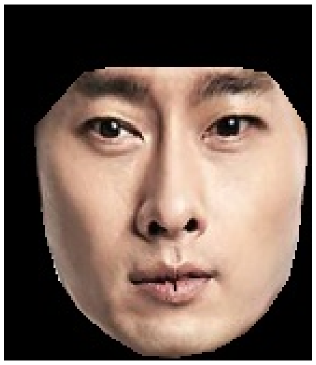
The images warped using Triangulation and Thin Plate Splines are shown in Figure 5 for data 1 and Figure 6 for data 2. Note that for data2, we are not swapping the two faces in each frame. Rather, there are reference faces for face 1 and face 2, taken from one frame in the video. All warping is performed with respect to the reference faces.

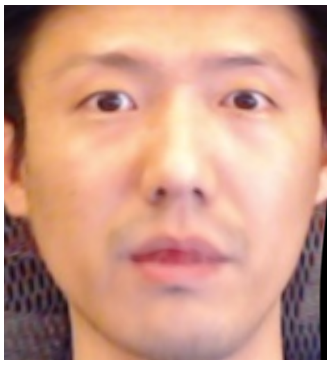
Comparing the warped images using Triangulation and Thin Plate Splines. We can see that, for triangulation, there are some black artifacts in the warped images (note the black region between the lips in the middle row of Figure 5 and Figure 6). For Thin Plate Splines, no such artifacts are spotted. This is because the affine transformation calculated based on triangulation would fail if the three vertices constituting the triangle lie closely on a line, leaving a black region in the warped image. As for Thin Plate Splines, since all facial landmarks are utilized in this process, and also regularization term is added, the calculation is always successfully and no black artifacts are rendered.


(a) face 1 (Zhentian Qian)

(b) face 2 (Hyun Bin)


(c) face 1 warped to face 2 using Triangulation


(d) face 2 warped to face 1 using Triangulation


(e) face 1 warped to face 2 using TPS

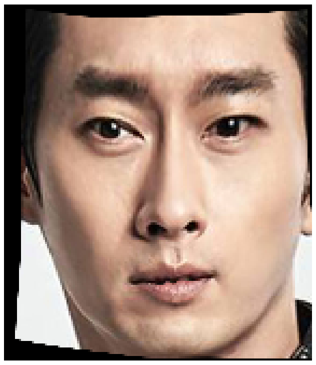
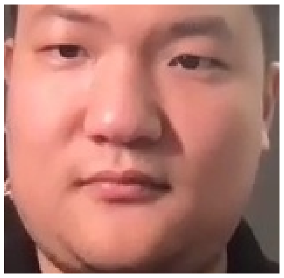
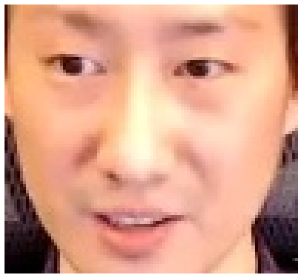
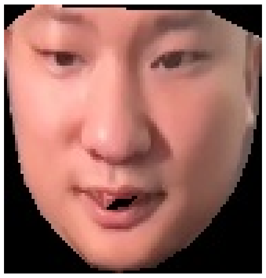
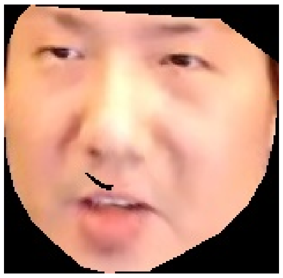

(f) face 2 warped to face 1 using TPS

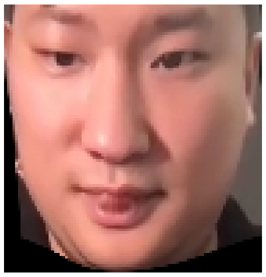
Fig. 5: Top row: Original images, Middle row (left to right): face 1 warped to face 2 and face 2 warped to face 1 using triangulation, Bottom row (left to right): face 1 warped to face 2 and face 2 warped to face 1 using Thin Plate Splines.


(a) face 1 (Haoying Zhou)


(b) face 2 (Zhentian Qian)


(c) face 1 warped to face 2 using Triangulation


(d) face 2 warped to face 1 using Triangulation


(e) face 1 warped to face 2 using TPS

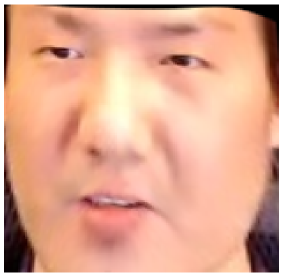

(f) face 2 warped to face 1 using TPS

Fig. 6: Top row: Original images, Middle row (left to right): face 1 warped to face 2 and face 2 warped to face 1 using triangulation, Bottom row (left to right): face 1 warped to face 2 and face 2 warped to face 1 using Thin Plate Splines.

C. *Replace Face*

Sample outputs of face replacement for data 1 and data 2 are shown in Figure 7 and Figure 8. Notice the difference in color and edges. The output is not a seamless blend.

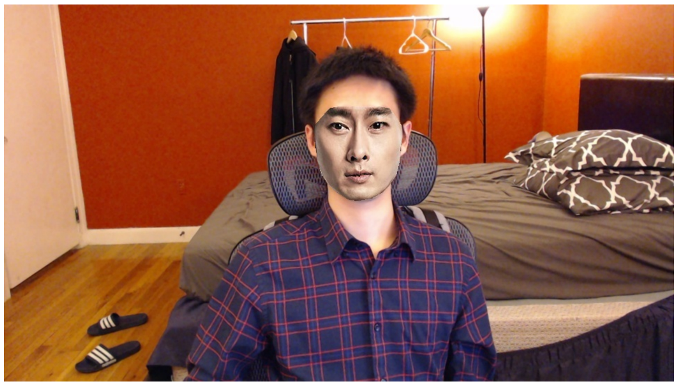


Fig. 7: Output of sample face replacement for data 1. Notice the difference in color and edges. The output is not a seamless blend.

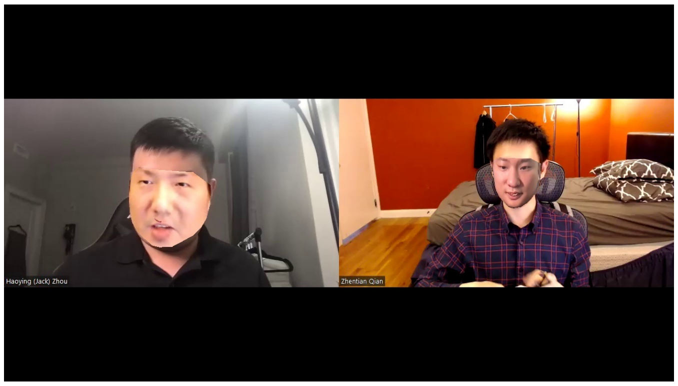


Fig. 8: Output of sample face replacement for data 2. Notice the difference in color and edges. The output is not a seamless blend.

D. Blending

We use the OpenCV seamless clone function to blend the warped face onto the target face, which is an implemetation of the possion blending [1]. The blending outputs for data 1 and data 2 using Triangulation and Thin Plate Splines are shown in Figure 9, Figure 10, Figure 11 and Figure 12.

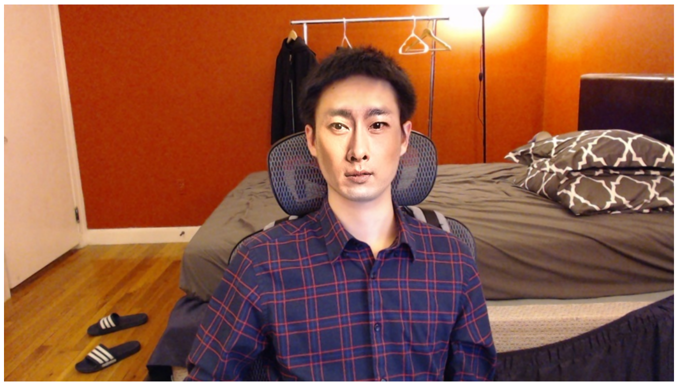


Fig. 9: Output of sample face replacement for data 1 after blending using Triangulation.

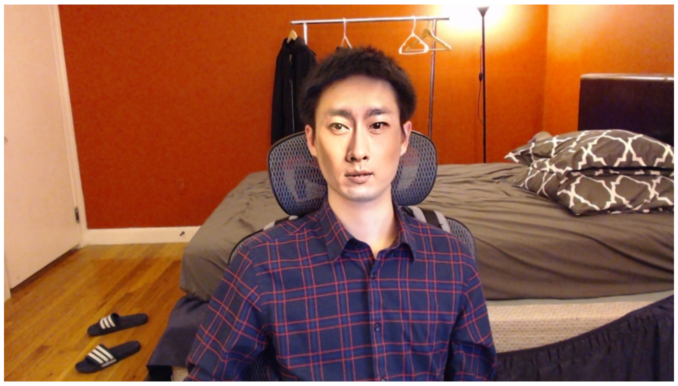


Fig. 10: Output of sample face replacement for data 1 after blending using Thin Plate Splines.

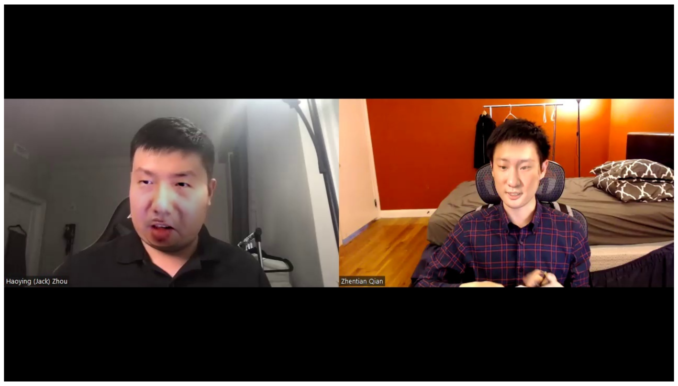


Fig. 11: Output of sample face replacement for data 2 after blending using Triangulation.

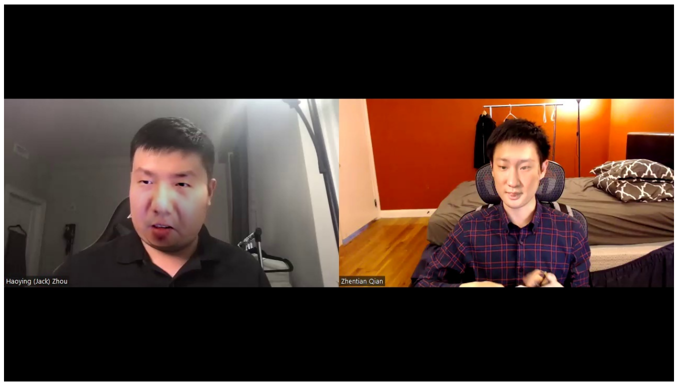


Fig. 12: Output of sample face replacement for data 2 after blending using Thin Plate Splines.

E. Motion Filtering

To reduce jittering of face landmarks, we implement a custom motion filter. This code is largely inspired by this project: https://github.com/mayankvik2/Stabilized-Facial-Landmark-Detection-in-Real-Time-Video. The main idea is to predict the current facial landmarks position using using the iterative Lucas-Kanade optical flow method with pyramids [2], and then treat the detected facial landmarks on the current frame using dlib as measurement and correct the prediction. The details of the motion filter can be found in algorithm 1.

```
Algorithm 1: Motion filter
Input: Previous filtered facial landmarks $f_{t-1}$, previous
    frame $I_{t-1}$, current frame $I_t$, current facial landmarks
    detection $m_t$, previous facial landmarks detection $m_{t-1}$.
Output: Filtered facial landmarks $f_t$;
    Predict the position of $f_{t-1}$ on the current frame $I_t$ using
    optical flow. The prediction is termed $p_t$;
    Calculate eye distance $ed$;
    $\sigma = ed^2/400$;
    $d = \|m_t - m_{t-1}\|$;
    $\alpha = \exp\{-d^2/\sigma\}$;
    $f_t = \alpha p_t + (1-\alpha)m_t$;
```

F. Failure Cases

One failure case is presented in Figure 13. We can see that the warped face is distorted. This is because the facial landmarks position is inaccurate on the jawline.

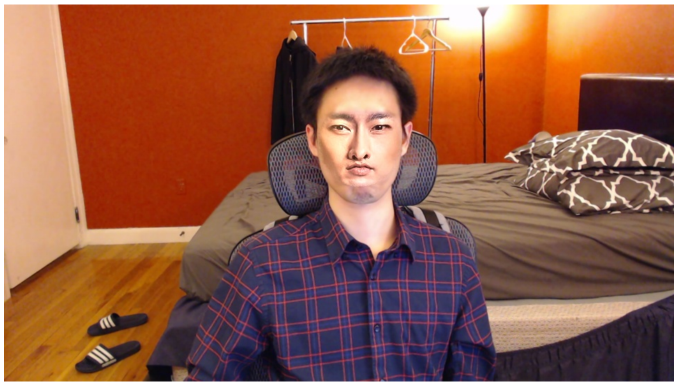


Fig. 13: Failure case

II. Phase 2: Deep Learning Approach

A. Facial Landmarks detection

Based on the problem description[3] and corresponding papers[4, 5], modifying the reference code with pre-trained model[6], we manage to find out the 68 points landmark for faces, shown as Figure 14.

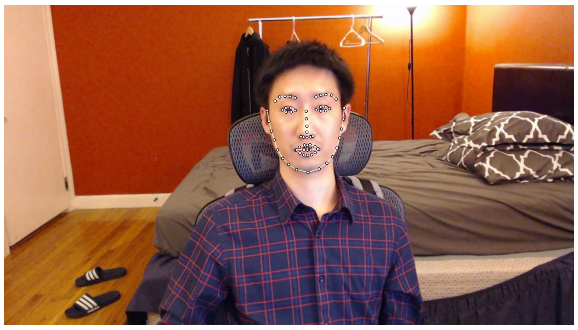
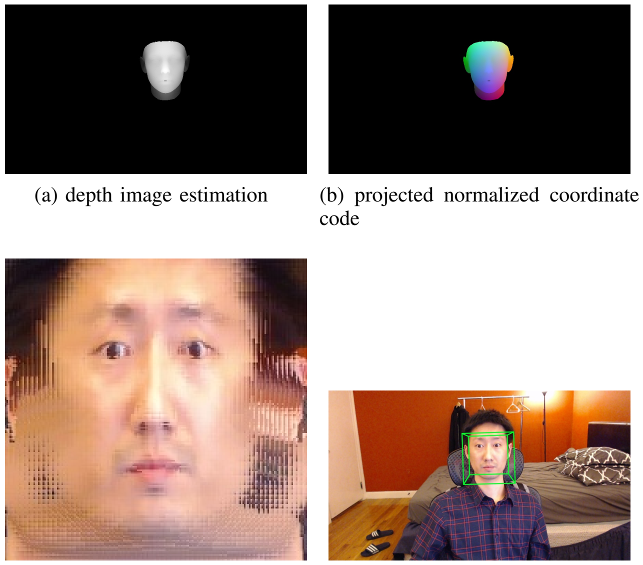


Fig. 14: Face landmark detected by deep learning method

Moreover, we can also obtain some other features such as the depth image estimation, projected normalized coordinate code (PNCC), pose adaptive feature(PAF) and pose estimation of the detected face, shown in Figure 15.

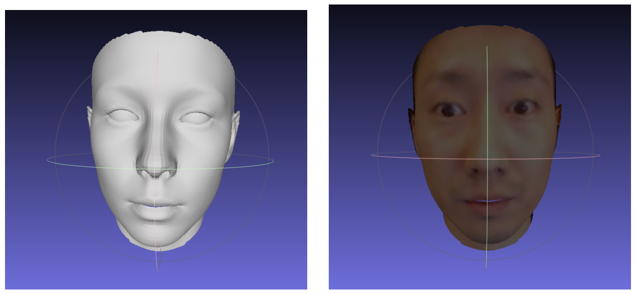


(a) depth image estimation (b) projected normalized coordinate code

(c) pose adaptive feature (d) pose estimation

Fig. 15: Features obtained from image using deep learning method

Meanwhile, we can reconstruct the 3D mesh of the face with its texture shown in Figure 16



(a) 3D mesh (b) 3D mesh with texture

Fig. 16: 3D mesh of the face with its texture

Furthermore, the deep learning method can also detect all faces if there are multiple faces in the image. An example is shown in Figure 17.

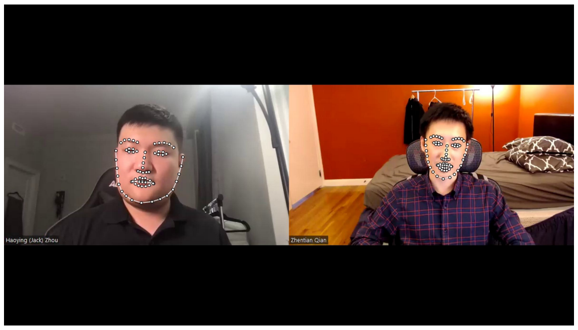
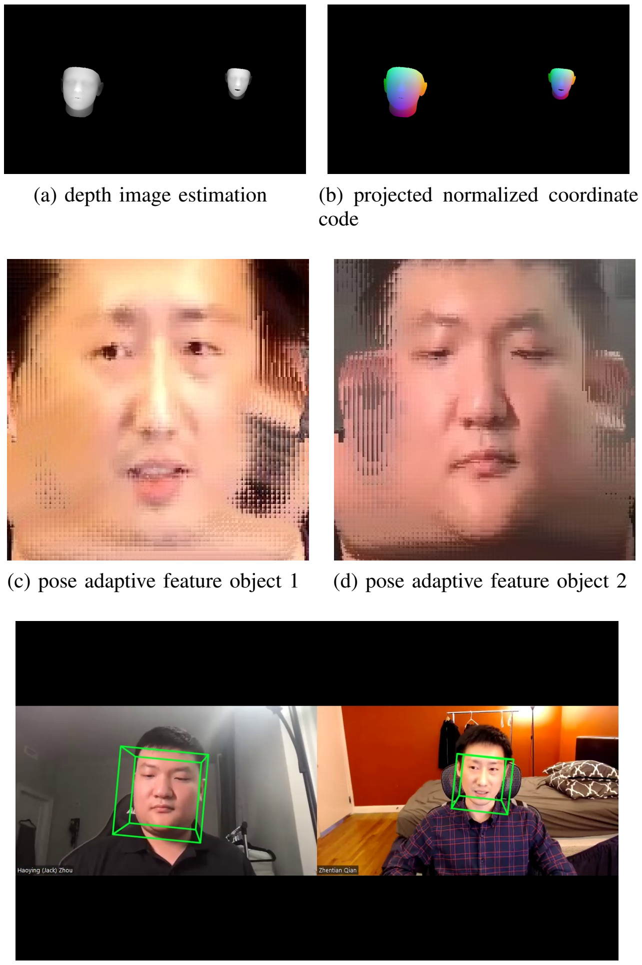


Fig. 17: Face landmark detected by deep learning method

Corresponding features are shown in Figure 18



(a) depth image estimation

(b) projected normalized coordinate code

(c) pose adaptive feature object 1

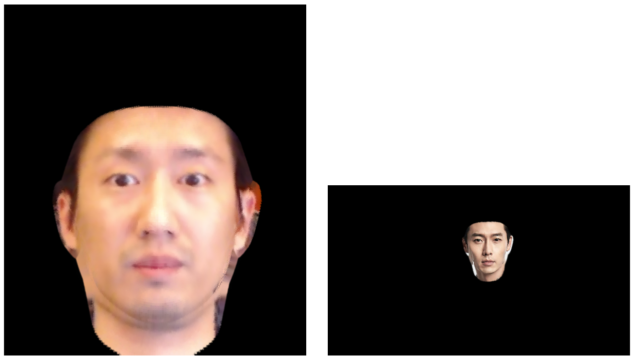
(d) pose adaptive feature object 2

(e) pose estimation

Fig. 18: Multiple face features obtained from image using deep learning method

B. Replacing face

Taking the advantage of the dense vertices predictions, we are able to obtain the 3D coordinates of the faces as well as its depth and color information, therefore, we can replace the vertices and their color information to swap faces. Using the same pair of faces in Figure 1, we can obtain the vertices after swapping shown in Figure 19.



(a) object1 with object2 face

(b) object2 with object1 face

Fig. 19: vertices after swapping faces

Knowing the 3D coordinates, using rendering algorithms or parallel projection, we are able to find the corresponding 2D pixel coordinate, therefore, replace the local RGB color values to swap the faces. Nevertheless, there are some issues such as non-convex area, distorted area and color difference. We are using `cv2.fillConvexPoly()`, `cv2.inpaint()` and Poisson blending to fix the problems.

Eventually, the results are shown in Figure 20 and Figure 21.

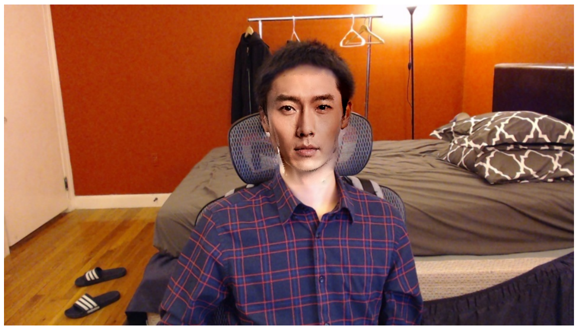


Fig. 20: Face swapping for Data1

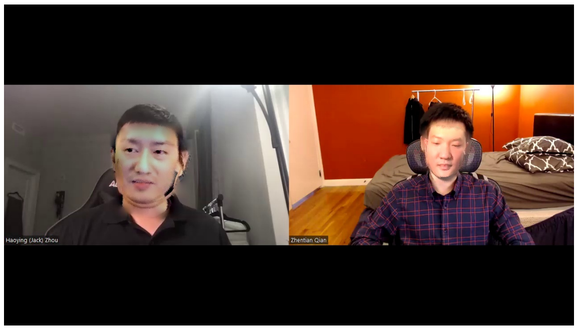


Fig. 21: Face swapping for Data2

In addition, the incorrect correspondence may lead to some mismatching. Also, the incorrect pose correspondence may let the images look torn.

In conclusion, deep learning are likely to achieve a better result compared to traditional approach.

References

[1] Patrick Pérez, Michel Gangnet, and Andrew Blake. "Poisson image editing". In: *ACM SIGGRAPH 2003 Papers*. 2003, pp. 313–318.

[2] Jean-Yves Bouguet et al. "Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm". In: *Intel corporation* 5.1-10 (2001), p. 4.

[3] Nitin J. Sanket, Lening Li, and Gejji Vaishnavi Vivek. *P2 Guidence*. URL: https://rbe549.github.io/fall2022/proj/p2/.

[4] Xiangyu Zhu et al. "Face alignment in full pose range: A 3d total solution". In: *IEEE transactions on pattern analysis and machine intelligence* (2017).

[5] Jianzhu Guo et al. "Towards Fast, Accurate and Stable 3D Dense Face Alignment". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2020.

[6] Jianzhu Guo, Xiangyu Zhu, and Zhen Lei. *3DDFA*. https://github.com/cleardusk/3DDFA. 2018.