3D Vision – Assignment 2

魏家博 (Chia-Po Wei)

Department of Electrical Engineering

National Sun Yat-sen University

- Put all the *.ipynb, *.npy, *.jpg in the same folder.
- $\mathbf{m}$ is a matrix of shape (2,68), and each column of $\mathbf{m}$ represents a facial landmark in $\mathbb{R}^2$. $\mathbf{m}$ is the variable `face2d` in the code template.
- Use `cv2.imread` to read the image, emma_watson.jpg.
- Convert the image from BGR to RGB.
- [TODO1] Display the 2D landmarks and the face image as in Figure 1.
- Hints
  - Use `plt.imshow()` to display the face image.
  - Use `plt.plot()` to draw the 2D landmarks.

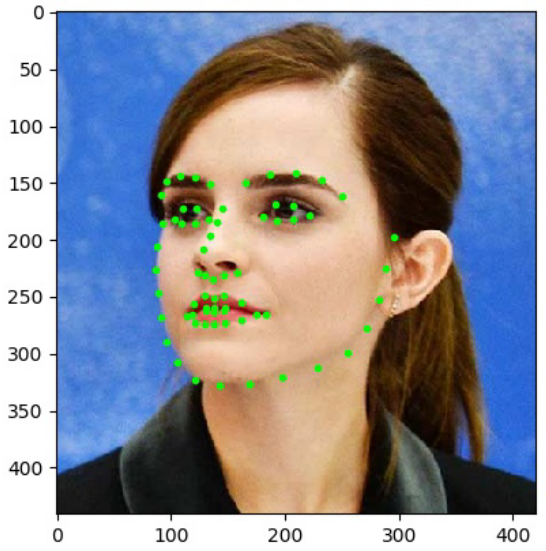

Figure 1: 2D Facial Landmarks

- $\mathbf{M}$ is a matrix of shape (3,68), and each column of $\mathbf{M}$ represents a facial landmark in $\mathbb{R}^3$. $\mathbf{M}$ is the variable `face3d` in the code template.

- [TODO 2] Pose Estimation

- Step 2.1: Construct the pseudo intrinsic matrix $\mathbf{K}$ as below

$$\mathbf{K} = \begin{bmatrix} f & 0 & cx \\ 0 & f & cy \\ 0 & 0 & 1 \end{bmatrix}$$

  where f is the max of `[height, width]`, cx = `width/2`, and cy = `height/2`.

  Hint: `height, width = img.shape[:2],` where `img` is obtained from `cv2.imread()`.

- Step 2.2: Use below to obtain the extrinsic matrix.

```
_, rotvec, transvec = cv2.solvePnP(face3d.T, face2d.T, K, None)
```

  You need `cv2.Rodrigues()` to get the rotation matrix from `rotvec`.

- Step 2.3: Compute Euler angles from the rotation matrix. Suppose R is a rotation matrix denoted by

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \ell = \sqrt{r_{11}^2 + r_{21}^2}$$

Use below to obtain the pitch, yaw, and roll angles from R. (Import math to use atan2.)

If $\ell \geq 10^{-6}$, pitch = atan2(r_{32}, r_{33}), yaw = atan2($-r_{31}, \ell$), roll = atan2(r_{21}, r_{11}).

Otherwise, pitch = -atan2(r_{23}, r_{22}), yaw = atan2($-r_{31}, \ell$), roll = 0.

Print the pitch, yaw, and roll angles in degrees. (You need to convert from radians to degrees, because atan2 returns values in radians.)

- [TODO 3] Compute the reprojected 2D landmarks, `face2d_repr`.
- Let `face2d` be the observed 2D landmarks given in TODO 1, and `face3d` be the 3D landmarks in TODO2.
- Use perspective projection to project `face3d` to `face2d_repr` as done in TODO 3 of Assignment 1, in which K and [R, t] are obtained from TODO 2 of this assignment.
- Note that you already have the intrinsic matrix K and the extrinsic matrix [R, t] from TODO 2, where R is the rotation matrix, and t is the translation vector (`transvec`).
- Draw `face2d` (green) and `face2d_repr` (red) as in Figure 2.

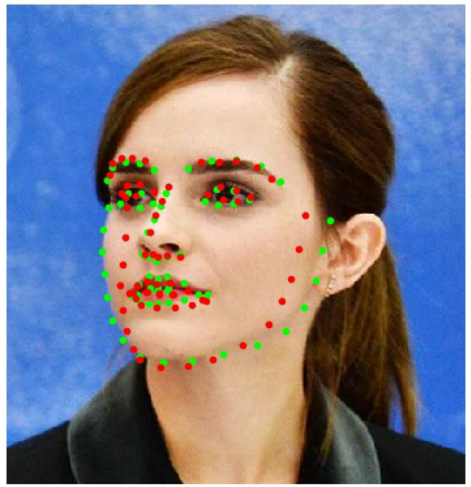

Figure 2

- [TODO 4] Print the reprojection error, which is defined by

$$\frac{1}{n}\sum_{i=1}^{n}\sqrt{(x_i-\hat{x}_i)^2+(y_i-\hat{y}_i)^2}$$

  - where (x_i, y_i) denotes the ith landmark of `face2d`, and $(\hat{x}_i, \hat{y}_i)$ denotes the ith landmark of `face2d_repr`, and n is the number of landmarks.

- [TODO 5] Previously, we use all the 68 points of `face2d`. Next, consider using 51 points from `face2d` as shown in Figure 3 (the face contour is removed) to repeat TODO 2 ~ TODO 4. Note that you also need to remove the corresponding face contour in `face3d`.

- [Question 1] Which is better to perform pose estimation, using 68 points or using 51 points? Explain the reason for your choice.

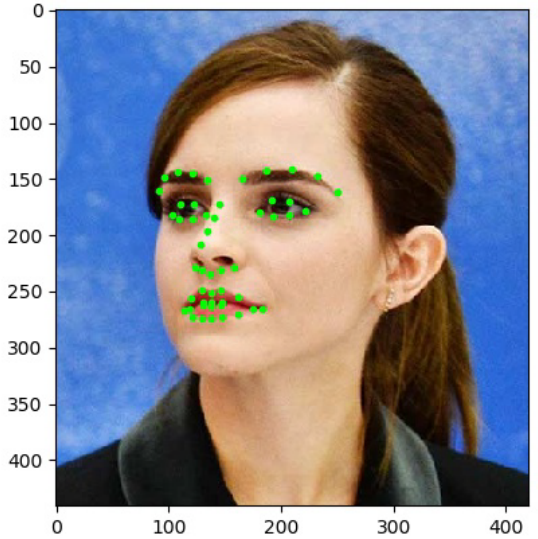

Figure 3

- [TODO 6] Augmented Reality
  - Step 5.1: Construct the eight vertices that contains `face3d` as in TODO 5 of Assignment1. The cube formed by the eight vertices is shown in Figure 4.
  - Step 5.2: Follow TODO 4 of Assignment 1 to project the eight vertices onto the 2D image plane, in which K and [R, t] are obtained from TODO 2 of this assignment.
  - Use `plt.plot()` to draw the edges of the cube and `face2d`, and the result is shown in Figure 5.

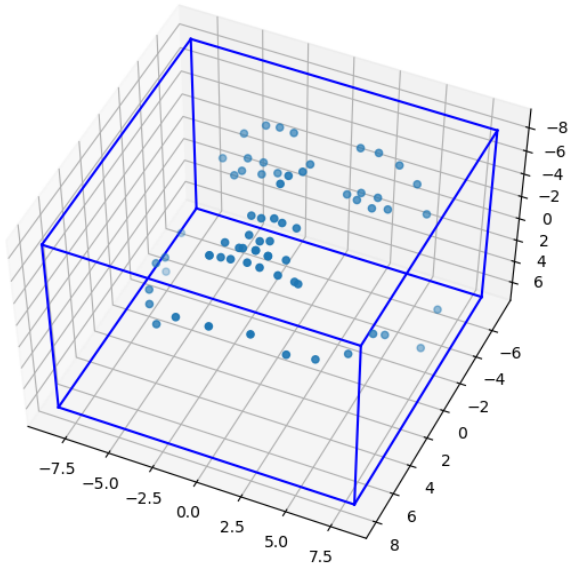

Figure 4

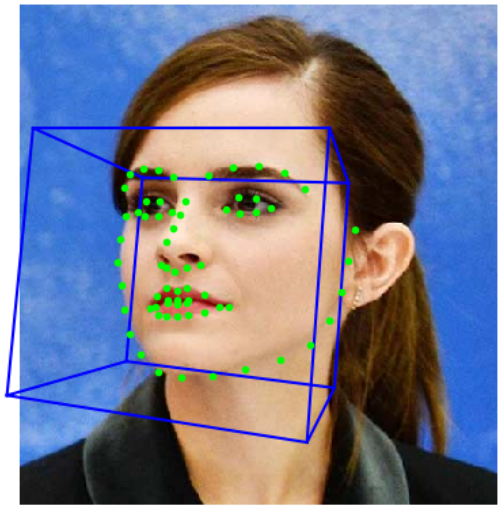

Figure 5

Grading

TODO 1: 5 points

TODO 2: 30 points

TODO 3: 20 points

TODO 4: 20 points

TODO 5: 10 points

Question 1: 5 points

TODO 6: 10 points

Remarks

1. In TODO 4, it is not required to use for loops to compute the reprojection error. In fact, it only requires numpy functions.

   If you use for loops, then you can only get 10 points (rather than 20 points) for this task even if your results are correct.