3D Vision Assignment 1

魏家博 (Chia-Po Wei)

Department of Electrical Engineering

National Sun Yat-sen University

- $\mathbf{M}$ 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 1] Use `ax.scatter3D()` to plot $\mathbf{M}$ as below.

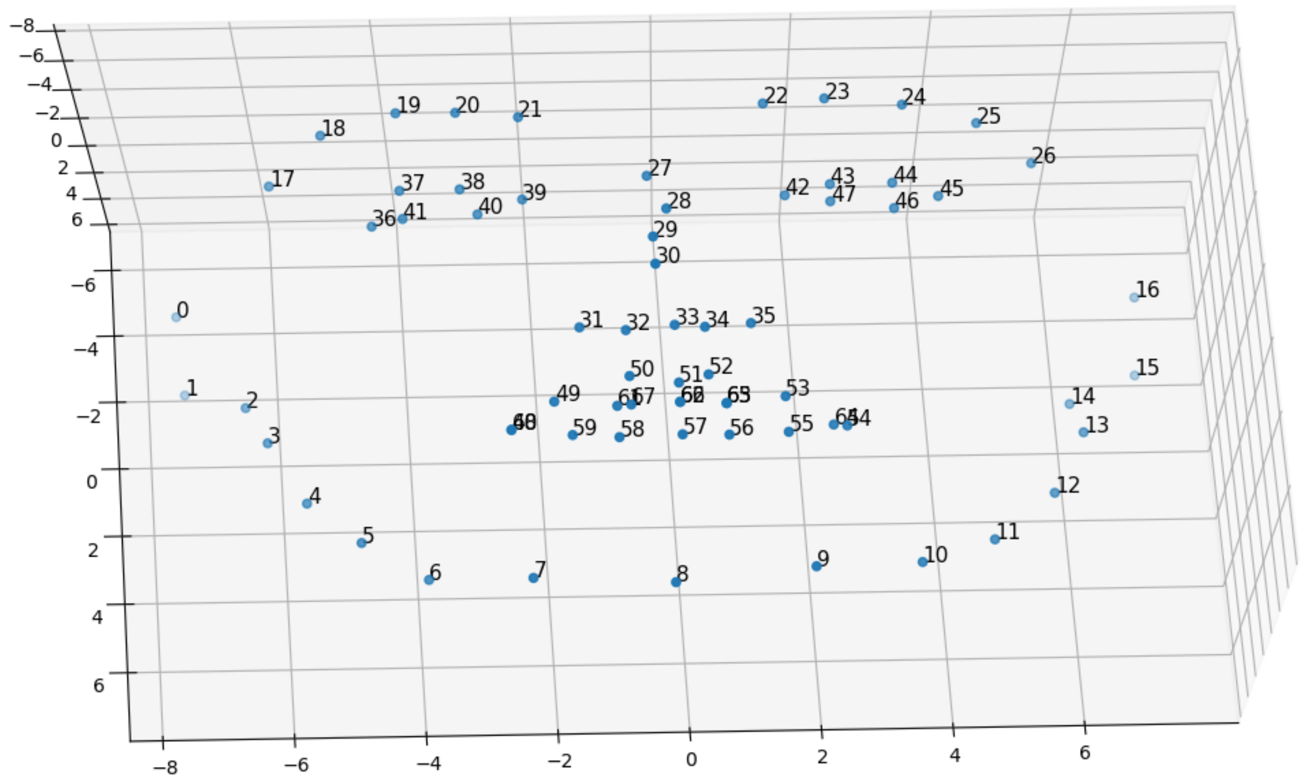

Figure 1: 3D Facial Landmarks

- [TODO 2] Compute the Euclidean distance between the inner eyes, i.e. the distance between `M[:,39]` and `M[:,42]`. The Euclidean distance between $\mathbf{p}_1 = [x_1, y_1, z_1]^T$ and $\mathbf{p}_2 = [x_2, y_2, z_2]^T$ is defined by

$$d(\mathbf{p}_1, \mathbf{p}_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

- [TODO 3] 3D-to-2D Projection.

- Step 3.1: Construct the intrinsic matrix $\mathbf{K}$ as below

$$\mathbf{K} = \begin{bmatrix} 640 & 0 & 320 \\ 0 & 640 & 240 \\ 0 & 0 & 1 \end{bmatrix}$$

- Step 3.2: Construct the extrinsic matrix $[\mathbf{R}, \mathbf{t}]$, where

$$\mathbf{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{t} = \begin{bmatrix} 0 \\ 0 \\ 30 \end{bmatrix}$$

You might need `np.hstack()` to construct the extrinsic matrix.

- Step 3.3: Use perspective projection to project each column of $\mathbf{M}$ to the 2D space, and save the results in $\mathbf{m}$, where $\mathbf{m}$ is a matrix of shape (2,68). To obtain $\mathbf{m}$, first compute $\widetilde{\mathbf{m}}$, where the *i*th column of $\widetilde{\mathbf{m}}$ is given by

$$\widetilde{\mathbf{m}}_i = \mathbf{K}[\mathbf{R} \quad \mathbf{t}]\widetilde{\mathbf{M}}_i \text{ with } \widetilde{\mathbf{M}}_i = \begin{bmatrix} \mathbf{M}_i \\ 1 \end{bmatrix}$$

  where $\mathbf{M}_i$ is the *i*th column of $\mathbf{M}$. Note that $\widetilde{\mathbf{M}}_i \in \mathbb{R}^4$ and $\widetilde{\mathbf{m}}_i \in \mathbb{R}^3$.

  Let $\mathbf{m}_i$ be the *i*th column of $\mathbf{m}$.

  If $\widetilde{\mathbf{m}}_i = \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix}$, then $\mathbf{m}_i = \begin{bmatrix} m_1/m_3 \\ m_2/m_3 \end{bmatrix}$.

- Step 3.4: Use `plt.plot()` to draw $\mathbf{m}$ as Figure 2.

  You might need `np.vstack()` to construct $\widetilde{\mathbf{M}}_i$.

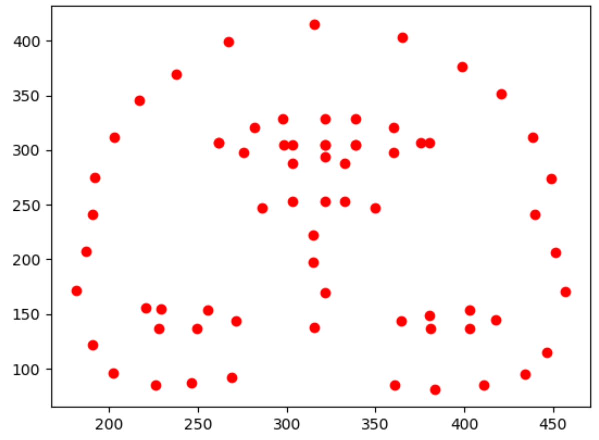

Figure 2

- Step 3.5: Add the following two lines before `plt.plot()` in Step 3.4.

  ```
  white_image = np.full((480,640,3),255)
  plt.imshow(white_image)
  ```

  and you will obtain Figure 3.

- Question 1:

  Why is the orientation of the face in Figure 2 different from that in Figure 3?

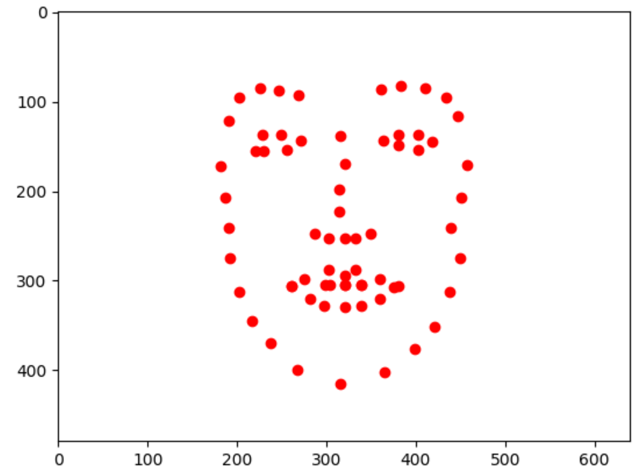

Figure 3

- [TODO 4] Repeat Steps 3.1, 3.2, 3.3, 3.5 but with different rotation matrices, i.e. replace the rotation matrix in Step 3.2 with

$$R = R_z(\alpha)\,R_y(\beta)\,R_x(\gamma) = \overset{\text{yaw}}{\begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}} \overset{\text{pitch}}{\begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}} \overset{\text{roll}}{\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix}}$$

There are two cases to be considered:

Case 1: $\alpha = 0°, \beta = 20°, \gamma = 0°$.

Case 2: $\alpha = 0°, \beta = 60°, \gamma = 0°$.

The plots for Cases 1 and 2 are as in Figure 4.

Note that the input of `cos` or `sin` function is in radians. Thus, you need to convert 20 from degrees to radians.

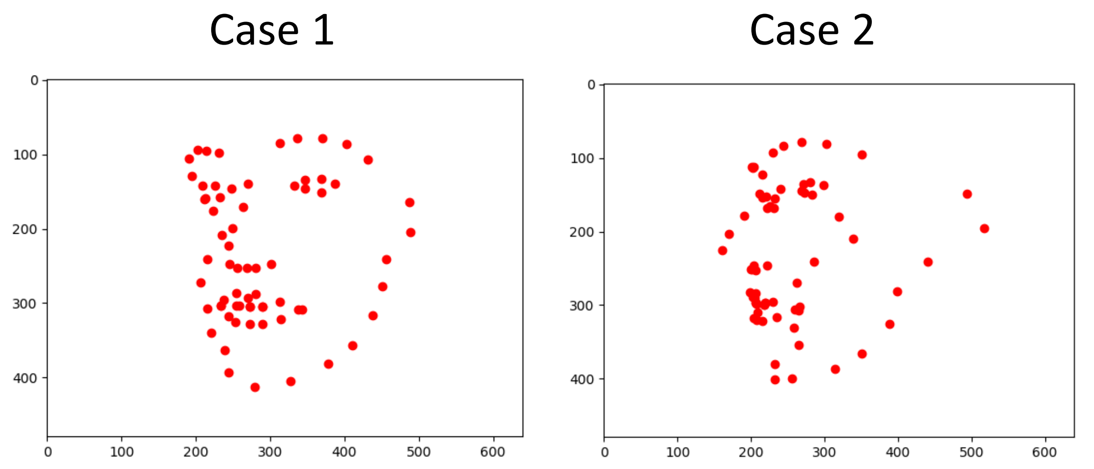

Figure 4

- [TODO 5] Augmented Reality
  - Step 5.1: Construct the cube that contains the 3D facial landmarks as in Figure 5.
    - Precisely, you need to construct the eight vertices of the cube as shown in Figure 6, where `xmin` is the minimum value of the x-components of $\mathbf{M}$ (3D facial landmarks) minus 1, `xmax` is the maximum value of the x-components of $\mathbf{M}$ plus 1, and `ymin`, `ymax`, `zmin`, `zmax` are defined in a similar manner.

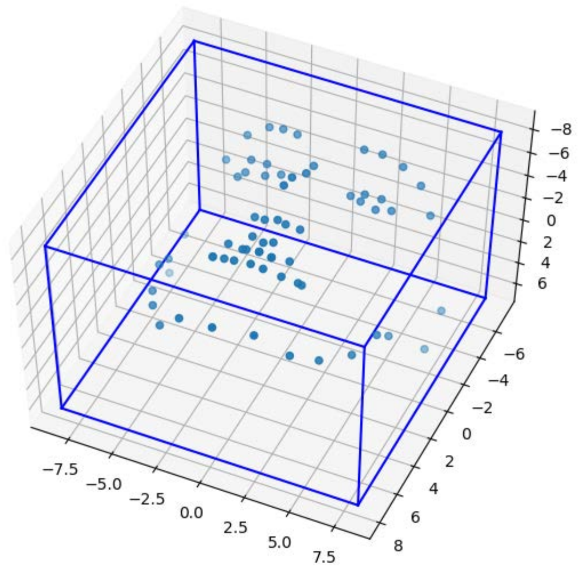

Figure 5

```
[xmin, ymin, zmin]
[xmax, ymin, zmin]
[xmax, ymax, zmin]
[xmin, ymax, zmin]
[xmin, ymin, zmax]
[xmax, ymin, zmax]
[xmax, ymax, zmax]
[xmin, ymax, zmax]
```

Figure 6

- Step 5.2: Follow TODO 4 to project the 3D facial landmarks and the eight vertices of the cube onto the 2D space. Then, use `plt.plot()` to draw the projected 2D facial landmarks and the edges of the cube. The result is shown in Figure 7 (corresponding to the rotation matrix with $\alpha = 0°, \beta = 20°, \gamma = 0°$).

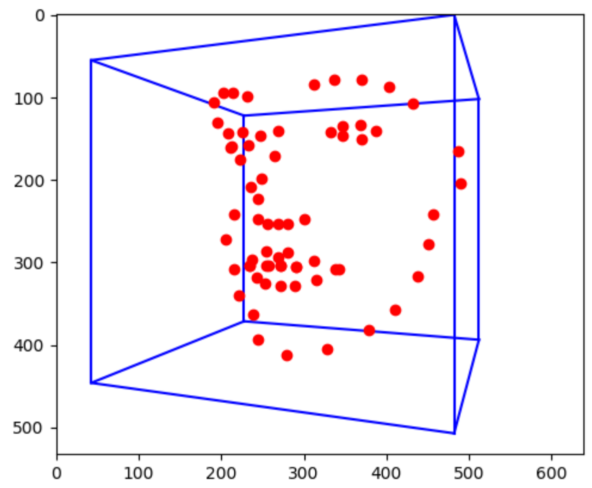

Figure 7

Grading

TODO 1: 5 points

TODO 2: 5 points

TODO 3: 40 points

Question 1: 5 points

TODO 4: 20 points

TODO 5: 25 points

Remarks

1. In TODO 3, you need to use numpy functions to implement 3D-to-2D projection, and cannot use existing functions.
2. In TODO 4, you need to use numpy functions to implement the rotation matrix, and cannot use existing functions.
3. In Step 3.3, it is not required to use for loops to obtain $\mathbf{m}$. In fact, it only requires matrix multiplication and element-wise division.

   If you use for loops, then you can only get 30 points (rather than 40 points) for this task even if your results are correct.