Lab Instructions - session 5

Linear Equations

Back to face models

Remember the linear combination of faces from Lab 2 where you had to tune the scalars `a`, `b`, and `c` to reconstruct `TargetFace2` as a linear combination of `Face1`, `Face2`, and `Face3`:

`face1.py`

```python
import matplotlib.pyplot as plt
import numpy as np
from face_data import Face1, Face2, Face3, TargetFace2, edges

def plot_face(plt,X,edges,color='b'):
    "plots a face"
    plt.plot(X[:,0], X[:,1], 'o', color=color, markersize=3)
    for i,j in edges:
        xi = X[i,0]
        yi = X[i,1]
        xj = X[j,0]
        yj = X[j,1]
        # draw a line between X[i] and X[j]
        plt.plot((xi,xj), (yi,yj), '-', color=color)
    plt.axis('square')
    plt.xlim(-100,100)
    plt.ylim(-100,100)
# make a guess
a = 1/3.
b = 1/3.
c = 1/3.

Face = a * Face1 + b * Face2 + c * Face3

plot_face(plt, TargetFace2, edges, color='r')
plot_face(plt, Face, edges, color='g')
# change a,b,c until the two plots align
plt.show()
```

In Lab 2 we found `a`, `b`, and `c` by trial and error. Now, we find them analytically.

To do this, first, vectorize the 68 by 2 face matrices to obtain 136-dimensional vectors:

```
face1 = Face1.ravel()
face2 = Face2.ravel()
face3 = Face3.ravel()
t = TargetFace2.ravel();
```

Then, arrange the face vectors as the columns of a 136 by 3 matrix **`F`**:

```
F = np.stack((face1, face2, face3), axis=1)
```

Let **x = [a,b,c]**^{T}. The relation **F x = t** gives a system of 136 equations and 3 unknowns. Solving for **x** gives the coefficients **a**, **b**, and **c**.

Task 1

Choose the first 3 equations from the above to get a system of 3 equations and 3 unknowns. Solve the equations to find **a**, **b**, and **c**.

`task1.py`

```
x = # solve the equations
a,b,c = x
Face = a * Face1 + b * Face2 + c * Face3

plot_face(plt, TargetFace2, edges, color='r')
plot_face(plt, Face, edges, color='g')
plt.show()
```

Task 2

Instead of choosing the first three equations from **F x = t**, choose three random equations. You may use **`np.random.choice`** to select 3 indices without replacement.

`task2.py`

```
for i in range(5):
    inds = np.random.choice(range(136), 3, replace=False)
    # choose the equations
    a,b,c = # solve the equations

    Face = a * Face1 + b * Face2 + c * Face3
    plot_face(plt, TargetFace2, edges, color='r')
    plot_face(plt, Face, edges, color='g')
    plt.show()
```

- Does choosing a different set of equations affect the result?

Task 3 - Noisy measurements

Add a little Gaussian noise to the target face.

```
TargetFace2 += 3 * np.random.randn(*TargetFace2.shape)
```

Now, repeat Task 2. What happens?

- Do we get a face close to `TargetFace2` in most iterations?
- Does the quality of the result depend on the choice of the 3 equations?
- Why do you think this happens?
- (Optional) Can you think of a better way to find the coefficients **a,b,c**?