Q 1. a) Understanding n-dot-l lighting

The diagram 2.a represents the unit vector geometries of a photometric stereo. v is the direction to viewer; n is the direction of the normal of the surface; l is the direction of the light source.

n-dot-l represents the cosine of the angle at which light hits the surface. The dot product of two unit vector always give the angle in between of those two vectors; this comes from its properties.

Projected area, from 2b, will also utilize the n dot l as the angle of incoming light as intensity (lambert's law) depend on it.

At any viewing direction, the projected area will always be the same since the projection only depends on unit vector of light, unit surface normal, distance to light source & diffuse component of light (Lambert's law).

$$I_d = \frac{k_d}{a + bq + cq^2}(\mathbf{l} \cdot \mathbf{n})L_d$$

q = distance to light source,
L_d = diffuse component of light

Q 1.b) Rendering n-dot-l lighting

Renderings are not available because of coding problems. However here are the steps taken:

1. Initializing image with given resolution.
2. Compute how much a circle takes in terms of pixel from the given radius & pixel size.
3. Compute the center point of the image.
4. Initialize the image array where a circle exists.
5. For each of those pixel, compute the intensity from the n-dot-l
6. n for a point in the sphere is the point minus the center coordinates, which is just the point. l is the imported direction of light (1,1,sqrt3) & others.
7. Plot the circle projection.

Q 1.c) Load Data code

```
def loadData(path = "../data/"):

    """
    Question 1 (c)

    Load data from the path given. The images are stored as input_n.tif
    for n = {1...7}. The source lighting directions are stored in
    sources.mat.

    Paramters
    ---------
    path: str
        Path of the data directory

    Returns
    -------
    I : numpy.ndarray
        The 7 x P matrix of vectorized images

    L : numpy.ndarray
        The 3 x 7 matrix of lighting directions

    s: tuple
        Image shape

    """

    I = None
    L = None
    s = None

    for i in range(7):
        j = i+1
        temp = imread(path + 'input_' + str(j) + '.tif')
        temp = rgb2xyz(temp)
        fors = np.copy(temp)
        temp = temp[:,:,1] #Just take luminance (Y)
        temp = np.reshape(temp, (temp.shape[0]*temp.shape[1]))

        if i == 0:
            I = np.copy(temp)
        else:
            I = np.vstack((I, temp))

    sources = np.load(path + 'sources.npy')
    L = np.copy(sources)
    L = L.T

    # s = (431, 369, 3)
    s = (fors.shape[0], fors.shape[1], fors.shape[2])

    print(L.shape, temp.shape, I.shape, s)

    return I, L, s
```

Q 1.d) Initials

I matrix should have a rank of 3, because B has a shape of 3xP, which indicates that I should have 3 non-zero independent vector columns.

Singular values:

[79.36348099 13.16260675 9.22148403 2.414729   1.61659626 1.26289066 0.89368302]

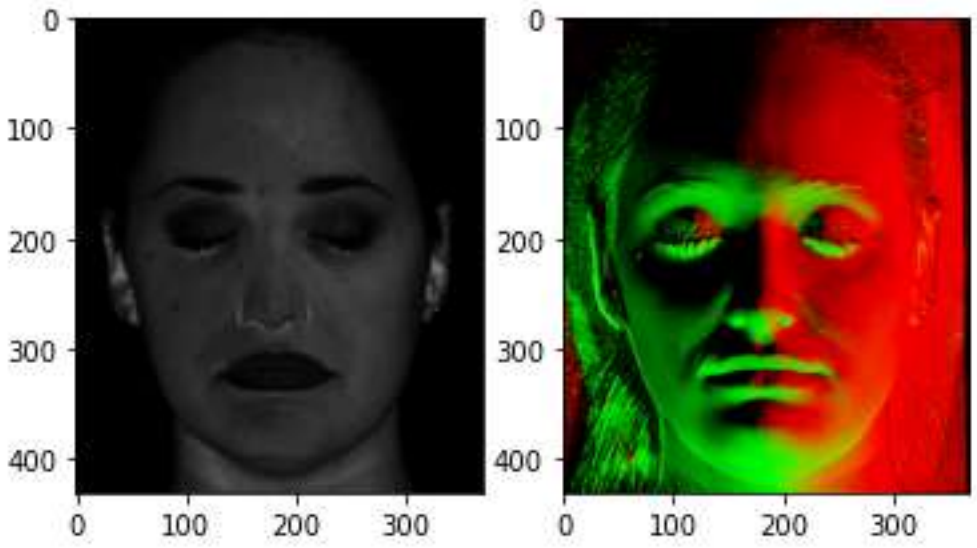
The SVD produces a rank of 7 for the I matrix. The SVD and singular values doesn't agree with the rank-3 requirements. This is because this pseudonormal estimation is done using the help of known light source directions. With unknown light source directions, the rank should be 3.

Q 1.e) Estimating Pseudonormals

I did this problem using matrix multiplication operator with inversing the transpose of L. I used np.linalg.pinv instead of the regular np.linalg.inv to make this work.

In theory, the matrix A can be separated into 7 diagonal matrices, because of its 7xP shape. Each of that diagonal matrix represents the seven different lighting conditions. In addition, the y vectors will correspond to each of those matrix as well.

Q 1.f) Display Albedos and Normals



Nothing jumps out from the ordinary other than the apparent nose and ear features as well as the darker spots in the eye and mouth area in the albedo image. After converting the normal into an image with 3 channels, the normal image matches with the grayscale albedo image.

Q 1.g) Normals & Depth

Normal at point (x,y) is N = (n1, n2, n3). Since N represents the tangential normal at that point in a 3D map, the normal vector is $\mathbf{N} = \begin{bmatrix} f_x(x_0, y_0) \\ f_y(x_0, y_0) \\ -1 \end{bmatrix}$, also the gradient, and fx & fy are partial derivatives. To get this equation, n (1 or 2 or 3) is the derivative of the substraction of the surface at (x,y) with z. This shows to be

$$F_x = \frac{\partial}{\partial x}(f(x,y) - z) = f_x \qquad F_y = \frac{\partial}{\partial y}(f(x,y) - z) = f_y$$

$$F_z = \frac{\partial}{\partial z}(f(x,y) - z) = -1$$

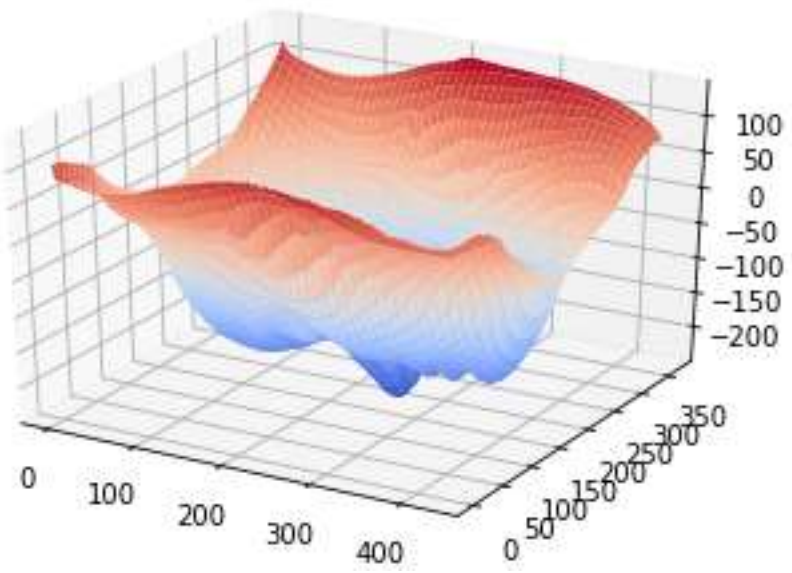
.

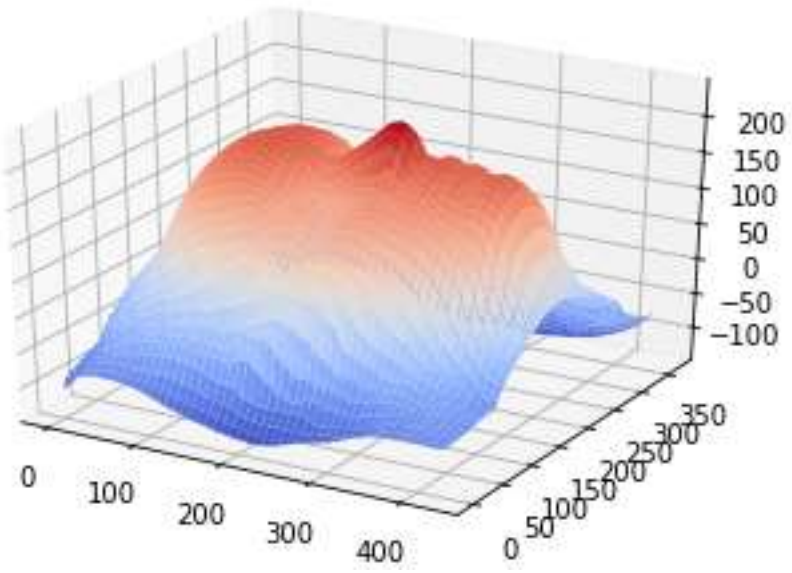
Therefore the partial derivatives can be represented n divided by n3 as n3 is the normal due to z axis.

Q 1.h) Integrability of gradients

The two methods in computing gradient matrix are using gx for first row & gy for rest of the g and gy for first column & gx for rest of g. The two methods will not result the same thing as the first one will result 1 s in the first row and the second one will result in 4 s in the first column. To make this non-integrable, we can use its partial derivatives. In our matrix case, we can represent derivative of x to be -n1/n3, while derivative of y to be -n2/n3.

Q 1.i) Shape Estimation



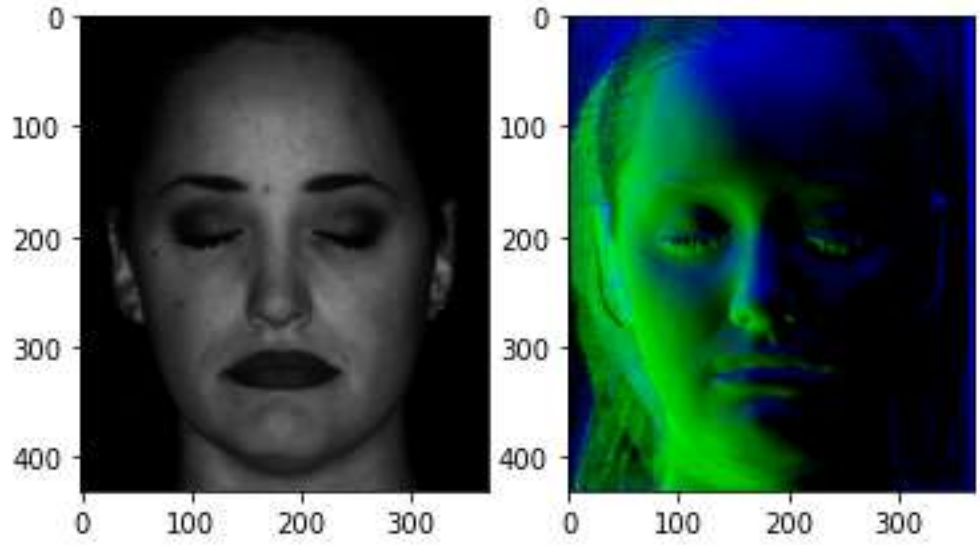


Q 2.a) Uncalibrated normal estimation

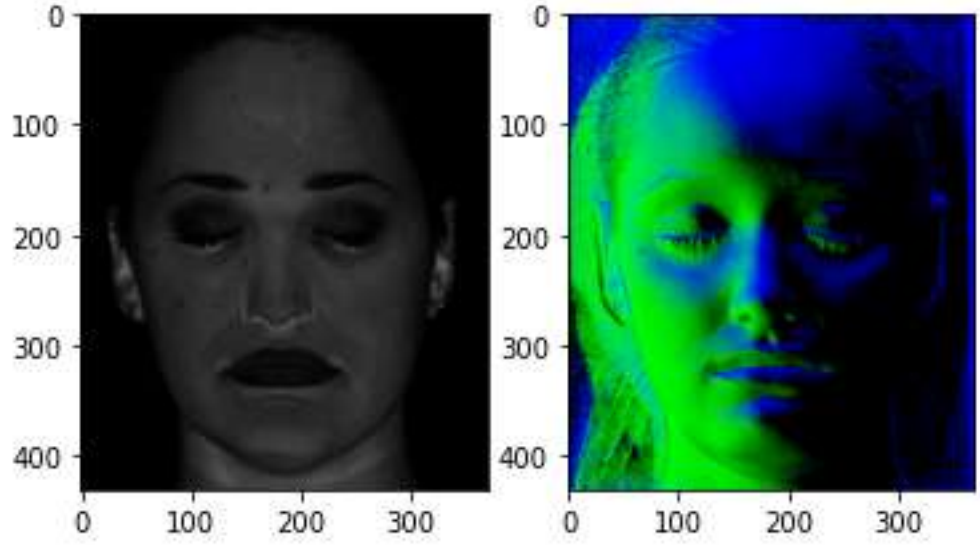
With $\hat{\mathbf{M}} = \mathbf{U}\tilde{\mathbf{\Sigma}}\mathbf{V}^T$ and the knowledge that sigma is the singular values, B and L matrix will constitute of the sigma matrix with the product of another term in the equation. Because the shape of B and L needs to have a shape of 3 (from the light sources), the term U, sigma & V should also have the three top columns selected.

The images shown in the next numbers use two variations of constructing B & L.

Q 2.b) Visualization



$L = U\Sigma^{1/2}$ & $B = \Sigma^{1/2}V^T$,



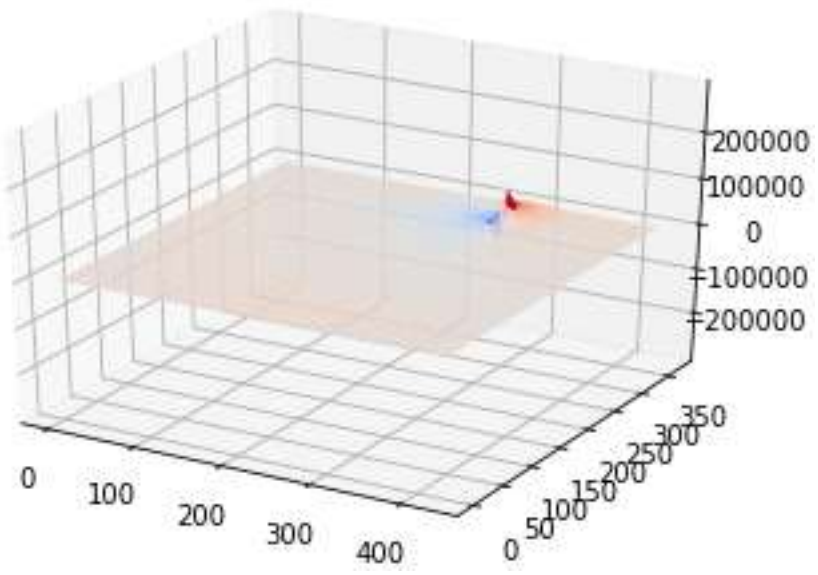
$L = U\Sigma^{1/2}$ & $B = V^T$

Q 2.c) Ground Truth Lighting

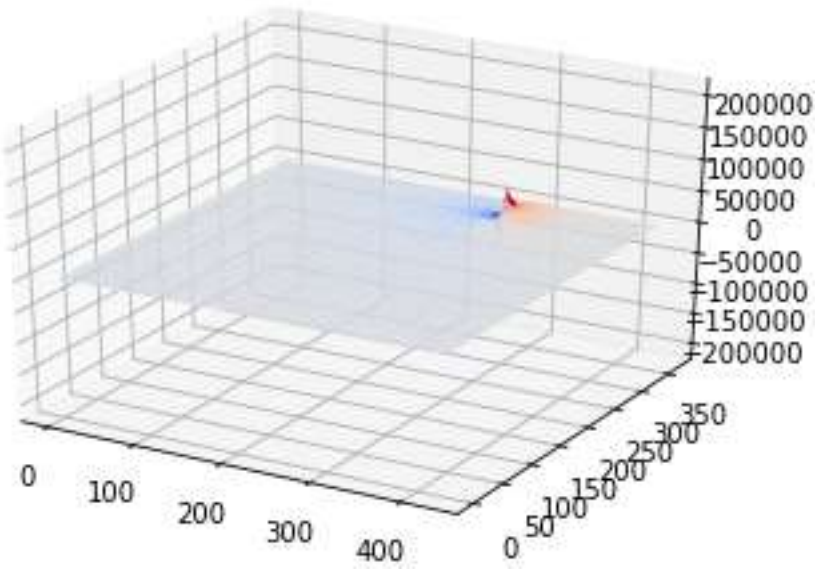
Although the ground truth lighting and the L estimated is quite similar, there are identified differences like the resolution of the projection.

To keep it 3 rank / column for (a) and uses the matrices calculated from SVD, the truncate function is used while keeping the whole shape of the array consistent.

Q 2.d) Reconstruction the shape, attempt 1



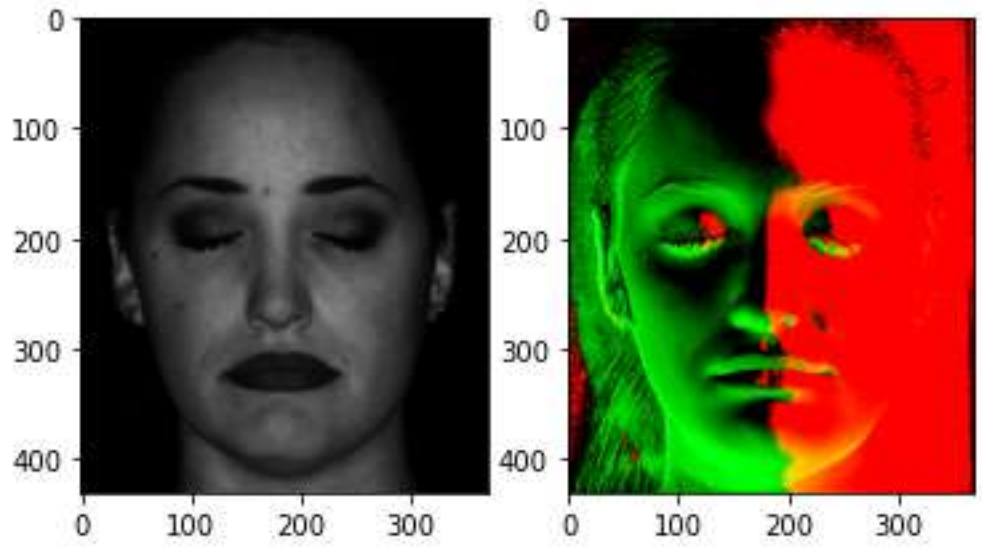
$L = U\Sigma^{1/2}$ & $B = \Sigma^{1/2}V^T$,

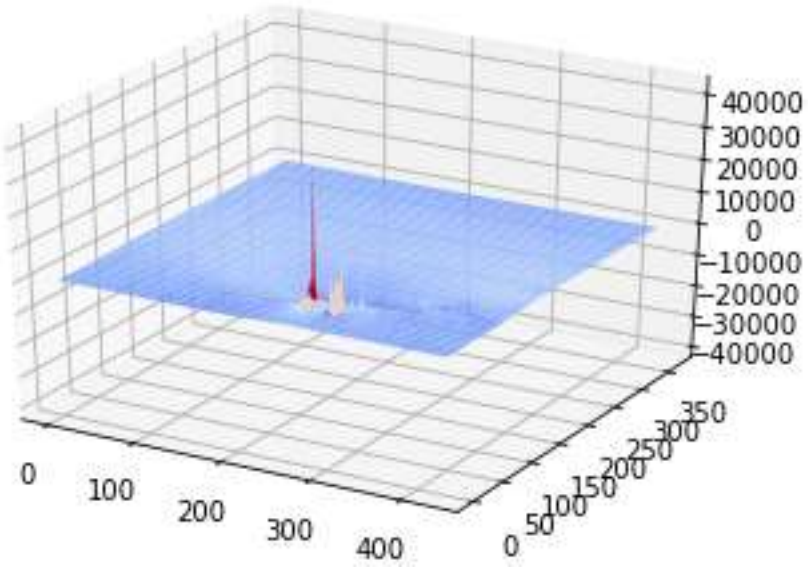


$L = U\Sigma^{1/2}$ & $B = V^T$

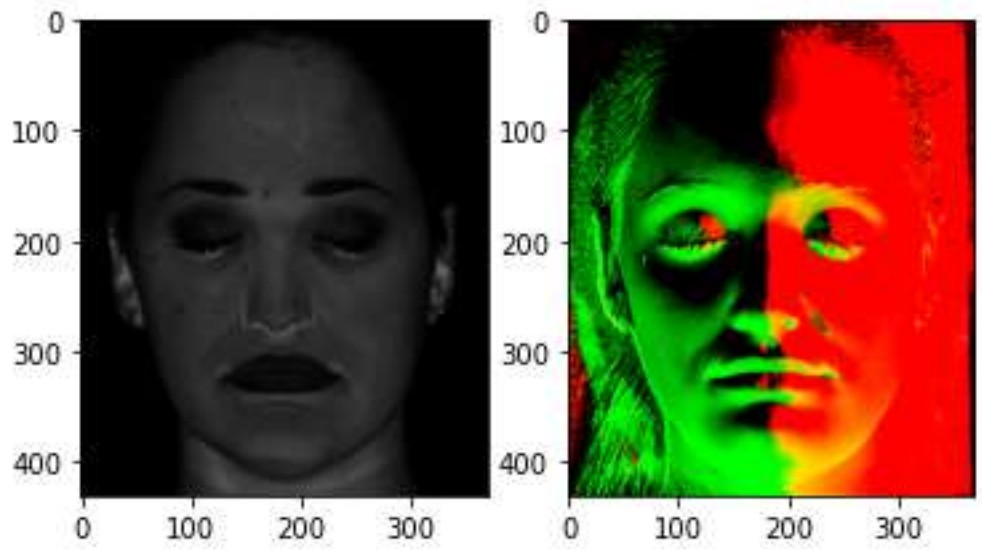
Definitely does not look like a face. The depth and surfaces is messed up compared to the previous reconstruction.

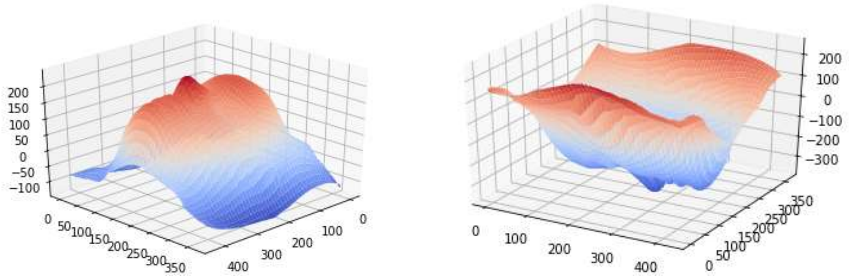
Q 2e) Reconstruction the shape, attempt 2

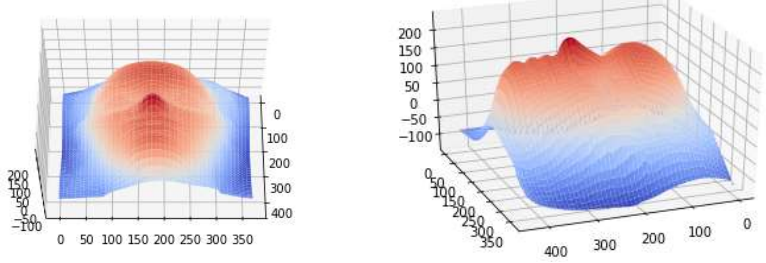




$L = U\Sigma^{1/2}$ & $B = \Sigma^{1/2}V^T$,





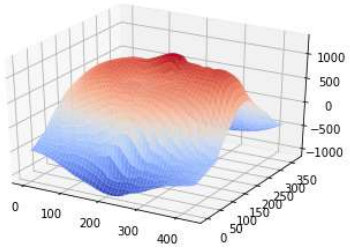


$L = U\Sigma^{1/2}$ & $B = V^T$

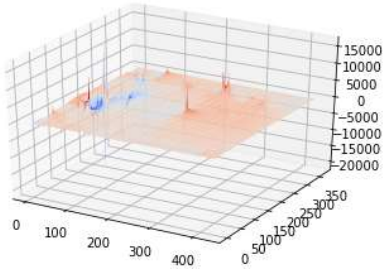
The 2nd approach seems to be more promising considering it produces a shape of a face, same like the one from calibrated photometric stereo.

Q 2.f)

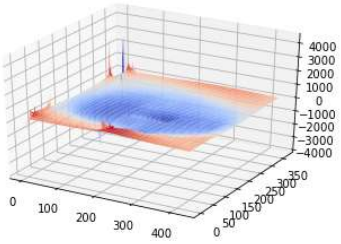
	μ	V	λ
1	0	0	0.5
2	0	0	5
3	10	10	15
4	10	0	0
5	0	10	0
6	10	10	0



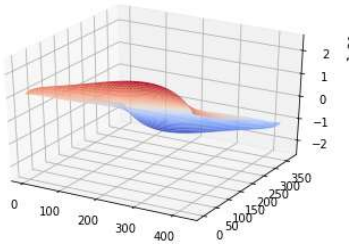
1



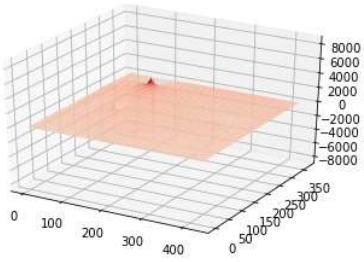
2



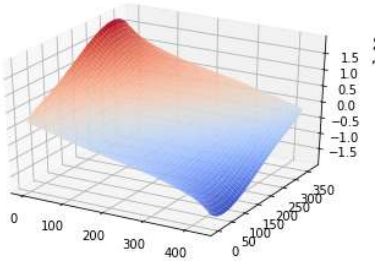
3



4



5



6

The three variables represent relief, elongation and some tilting. Although it is not well represented in the images above (especially the elongation), image 4 and 6 shows the tilting in action. **λ** should represent the elongation (stretch and compress) as it controls the scalability to the camera and, eventually, to the projection. Bas relief is so named, because of its ability to alter the projection.

Q 2.g) Flattest surface possible

To generate the flattest surface possible, the tilt should be 0, meaning μ & V will be 0. On the other hand, to eliminate any curves, the surface should be elongated heavily, stemming from a large number or an infinite number for λ.

Q 2.h) Measurements

With more than 7 pictures of the face, it will not entirely solve the ambiguity, but will help to minimize the error from the actual face. An infinite number of images of all the light luminance possibilities will only solve the ambiguity.