HW6, 16-720A
April 22, 2022
Nicholas Magal, nmagal@andrew.cmu.edu

Q1.a)

The geometry shows us a beam of light *I* hitting a small area on the ground called *dA*. We can view this as irradiance, which is power of electromagnetic radiation/area, which is watts/meter^2. This measures the amount of light hitting a surface.

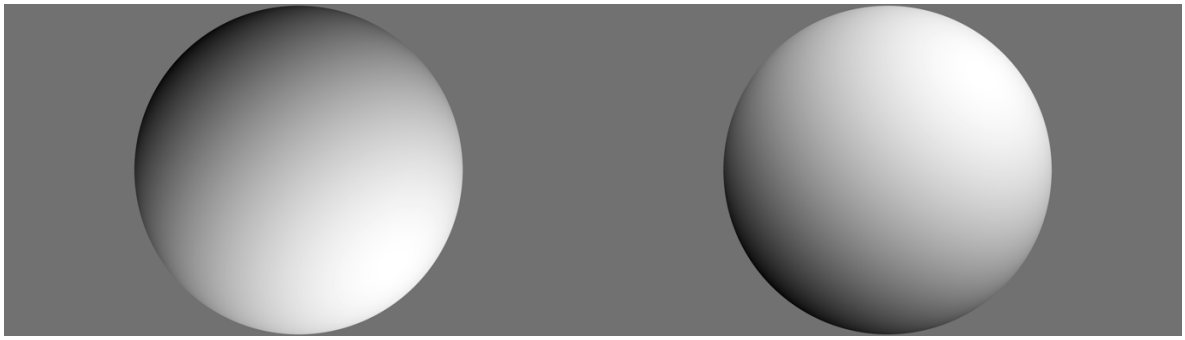
The second part of the picture shows us *v* leaving *dA*. This is referred to as radiance, which measures the amount of light traveling along a particular direction in space.

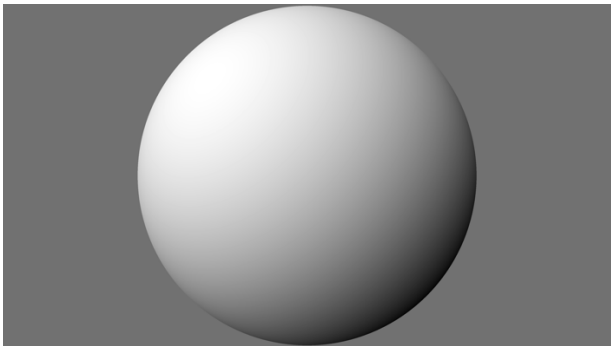
The dot product comes from the equation: $L(w_r) = A \frac{\rho_d}{\pi} \boldsymbol{n} * \boldsymbol{s}$. This equation gives us our radiance. The dot product is equivalent to cos θ. The purpose of cos θ is to measure the angle between the normal and our light source. If the angle is perfectly equal, then we will have a higher radiance compared to if their angle is 0. This comes from Lamberts Cosine Law.

The projected area refers to the 'foreshortened area'. It is equal to the original area cos θ. It comes from measuring the solid angle of the surface irradiance/radiance.

Since we are assuming a diffuse object, the viewing direction will not matter. This is because a diffuse object will equally project light photons out in equal amounts in a hemisphere around the point that the light is hitting.

Q1.b)





Q1.c)

```python
def loadData(path = "../data/", debug = False):

    """
    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.

    Parameters
    ----------
    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

    """

    #Let us read in our images and convert them to XYZ color space
    image_folder = []


    for i in range(1, 8):
        image_name = path +'input_'+str(i)+'.tif'
        image = imread(image_name)

        image = rgb2xyz(image)
        s = image.shape[0:2]

        image_folder.append(image)

    if debug == True:
        #Plot them to confirm read worked
        for image in image_folder:
            io.imshow(image)
            plt.show()

    #Now extracting the luminance channel Y and vectorizing images
    im_fol_lum_flat = []

    for image in image_folder:
        im_fol_lum_flat.append(image[:,:, 1].flatten())


    I = np.stack(im_fol_lum_flat, axis = 0)

    #Now getting image lighting directions
    L = np.load(path+"sources.npy")

    return I, L, s
```

Q1.d).

The fact that this $\boldsymbol{I}$ matrix is rank 3 comes from it's decomposition, $\boldsymbol{L^T B}$**.** From the Hayakawa paper, given that there are 3 surface normals that do not lie in a plane, then the rank of B is rank 3. Also, given that three light sources do not lie in a plan, then $\boldsymbol{L}$ will also be rank 3**.**

Given these facts, if the $\boldsymbol{I}$ matrix does not have noise, then the $\boldsymbol{I}$ matrix will also be of rank 3.

The SVD values I am getting are:

[79.36348099, 13.16260675, 9.22148403, 2.414729, 1.61659626 , 1.26289066, 0.89368302]

Since there are 7 values of nonzero singular values, the rank of the matrix is equal to 7. This might be 7 instead of 3 due to the matrix containing noise. The top three singular values have significantly higher magnitudes then the last 4, confirming our intuition about the noise causing the 7 nonzero values.

Q1.e

For this question I ended up solving for the equation using the closed form solution, and followed along from the First Principles of Computer vision Lambertian case youtube video.

To solve for it, first consider the case where we have three different light sources. To solve for the image intensities, we get the equations:

$$I_1 = \frac{\rho}{\pi} n * l_1$$

$$I_2 = \frac{\rho}{\pi} n * l_2$$

$$I_3 = \frac{\rho}{\pi} n * l_3$$

Where l is the light source direction, n is the normal, $\frac{\rho}{\pi}$ is the albedo divided by π, and I is the image intensity.

We can put these equations into matrix form as:

$$\begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix} = \frac{\rho}{\pi} \begin{bmatrix} l_{x_1} & l_{y_1} & l_{z_1} \\ l_{x_2} & l_{y_2} & l_{z_2} \\ l_{x_3} & l_{y_3} & l_{z_3} \end{bmatrix} n$$

$$I = LB$$

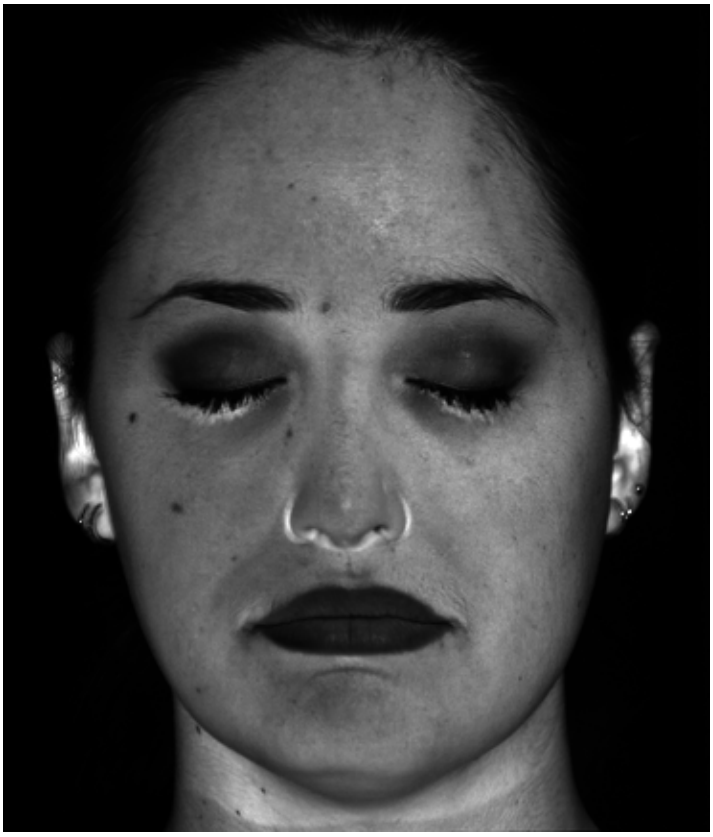
From here we can solve for B as

$$B = (L)^{-1} I$$

However, please note that the above solution only works if L is invertible, which it may not be. Therefore, we have to use the pseudo inverse, which leads us to this formula:

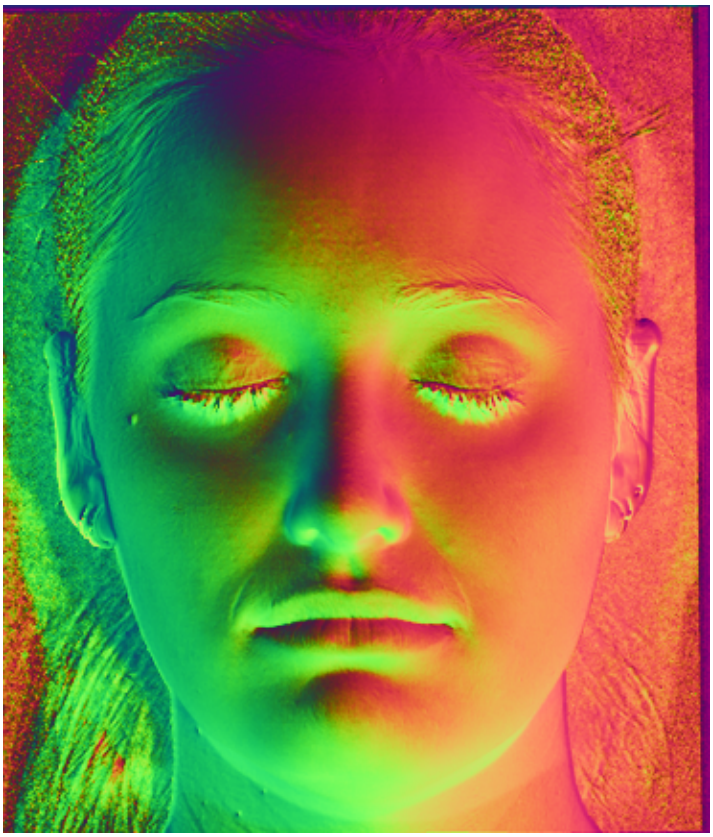
$$B = (L^T L)^{-1} L^T I$$

Q1.f)



Albedo Image after clipping to values between 0 and .5

An unusual feature shown in the albedo is that certain areas on the face have a brighter albedo then other parts of the face. This could be due to noise in our data



Normal image

For the most part, the normals do match my expectation of the curvature of the face. One part that does not is that the face's normal are not entirely symmetrical. For example, the forehead and the chin normal directions extend over the face in an uneven way.

Q1.g

From Epstein et al paper, "Learning Object Representations from Lighting Variations", we learn where $f_x = \frac{-n_1}{n_3}$ and $f_y = \frac{-n_2}{n_3}$ comes from. This comes from the surface integrability of the normals, which make sure that the normal vectors are consistent with a surface. This leads to the above equations.

Q1.h

g_x can be written as:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

g_y can be written as:

$$\begin{bmatrix} 4 & 4 & 4 & 4 \\ 4 & 4 & 4 & 4 \\ 4 & 4 & 4 & 4 \end{bmatrix}$$

For part 1 of this question, we get the result:

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

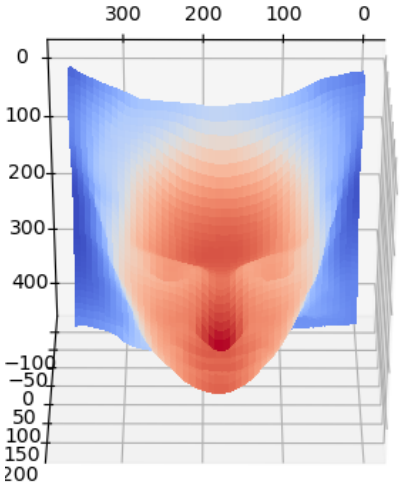
For part 2 of this question, we get the result:

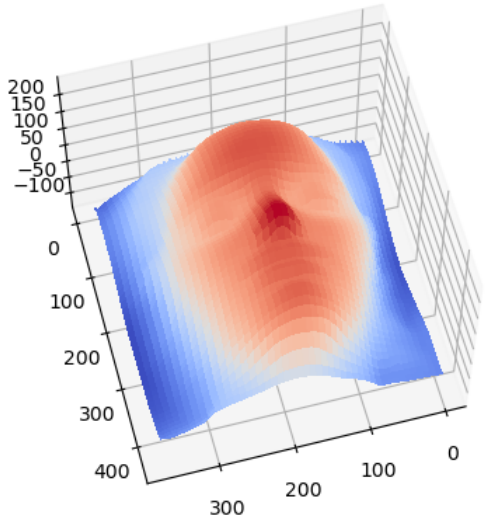
$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

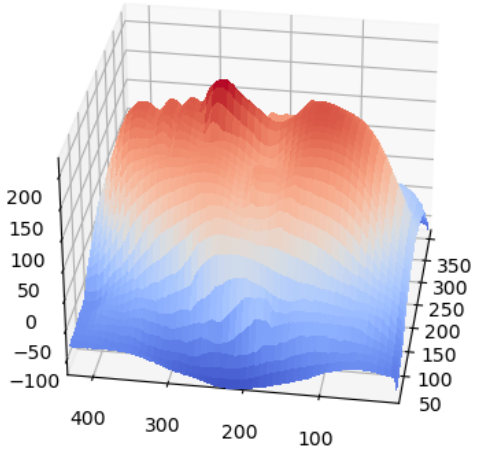
In order for a function to be integrable it must be smooth and continious. Therefor if we had a step like object, the gradients would be undefined at the edge of this object. This will be non-integrable.

Also, if our gradients have noise, then this will also lead back to our method not being able to recreate g. In practice, normals that we computed tend to be noisy, so this is indeed a potential problem. The way we calculated this is specifically vulnerable to this as each value we calculate is reliant on the previous value. In order to fix this, we could average over different integration paths.

Q1.i)


300
200
100
0
0
100
200
300
400
−100
−50
0
50
100
150
200


200
150
100
50
0
−50
−100
0
100
200
300
400
300
200
100
0


200
150
100
50
0
−50
−100
400
300
200
100
350
300
250
200
150
100
50

Q2.a)

The solution follows from the paper by Hayakawa, *Photometric stereo under a light source with arbitrary motion.*

In order to satisfy the required constraints, first, we do a rank-k approximation of the matrix I. We get the following equation:

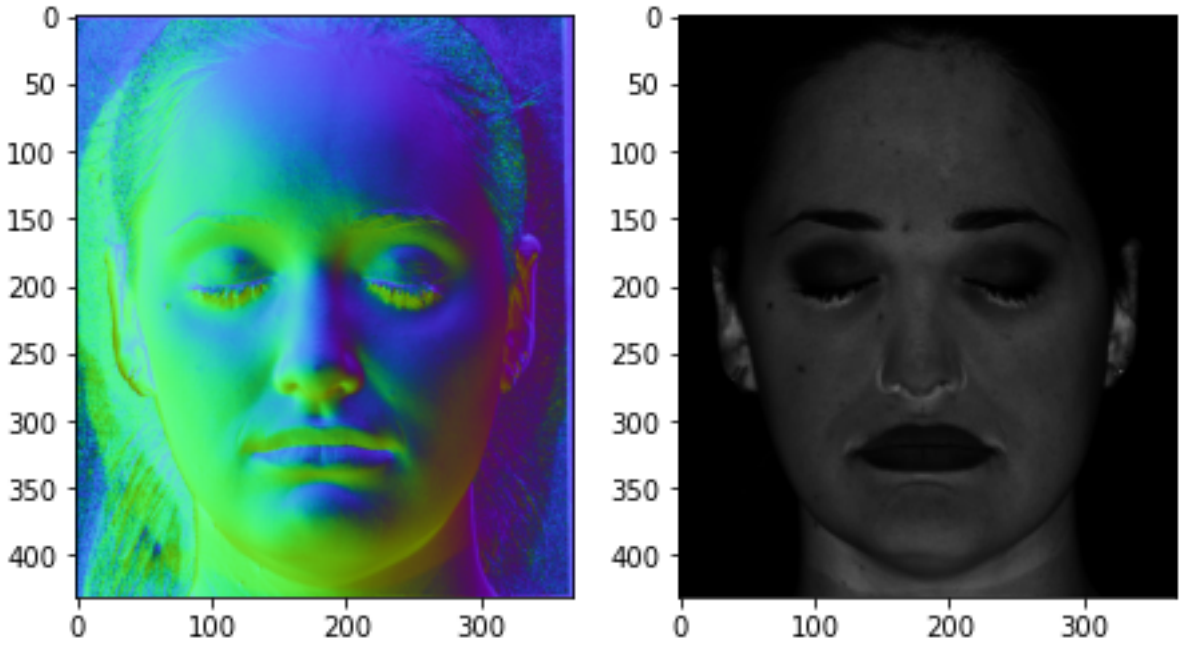
$$I = U\Sigma V^T$$

From this we set all but the first three columns of *U* equal to 0. Then, we set all but the first three singular values equal to zero. Following this, we finally set all but the first three rows of V^T equal to 0. Following this, using these updated values, we can solve for our pseudo normals and light matrix with:

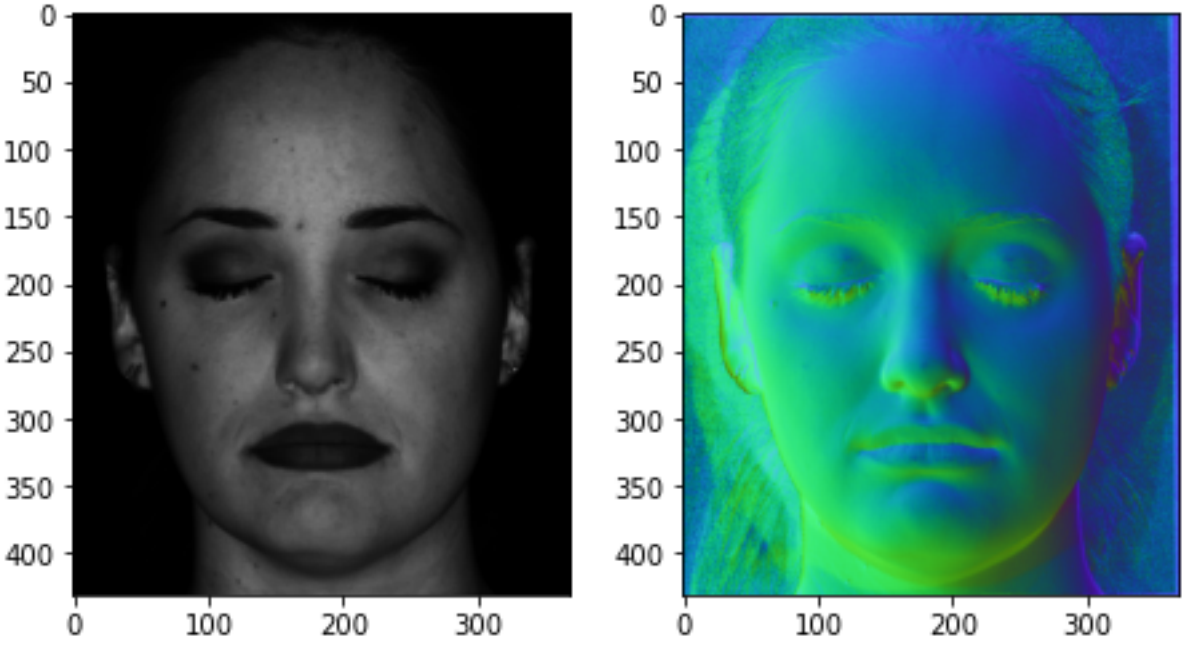
$$B' = U' \pm \Sigma'^{\frac{1}{2}}$$

$$L' = \pm \Sigma'^{\frac{1}{2}} V'$$

Q2.b)



Albedos and Normal when not including Sigma in my pseudo normal calculations



Albedos and Normal when including Sigma in my pseudo normal calculations

Q2.c)

My L turns out to be different from the original L.

	0	1	2	3	4	5	6
0	-0.1418	0.1215	-0.069	0.067	-0.1627	0	0.1478
1	-0.1804	-0.2026	-0.0345	-0.0402	0.122	0.1194	0.1209
2	-0.9267	-0.9717	-0.838	-0.9772	-0.979	-0.9648	-0.9713

Ground Truth L

	0	1	2	3	4	5	6
	-0.33593	-0.434409	-0.270303	-0.42038	-0.403133	-0.380156	-0.376326
	0.261245	-0.638663	0.137571	-0.172544	0.641031	0.128458	-0.218496
	0.61888	0.334119	0.141413	-0.00569798	-0.102339	-0.300569	-0.620086

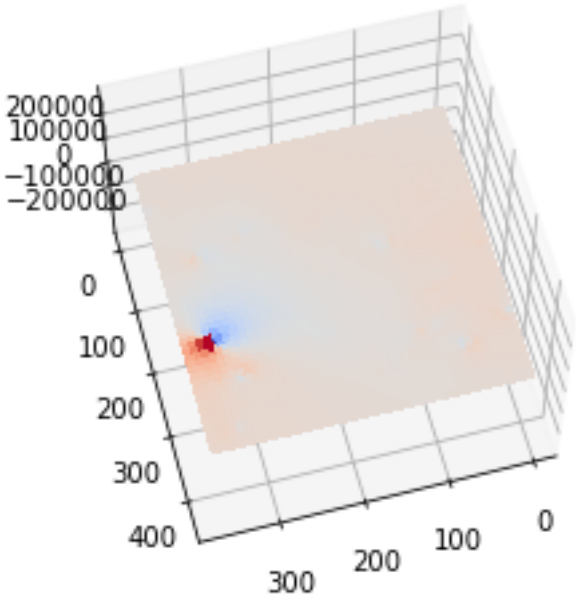
Estimated L

A simple factorization that could be made is changing the allocation of Σ. In order to find the optimal allocation of Σ, we would need to minimize over a function that choses allocation of Σ based off of finding L'=L. The only problem with this approach is that for uncalibrated reconstruction, we do not have access to L, which is why there is a ambiguity in this case.

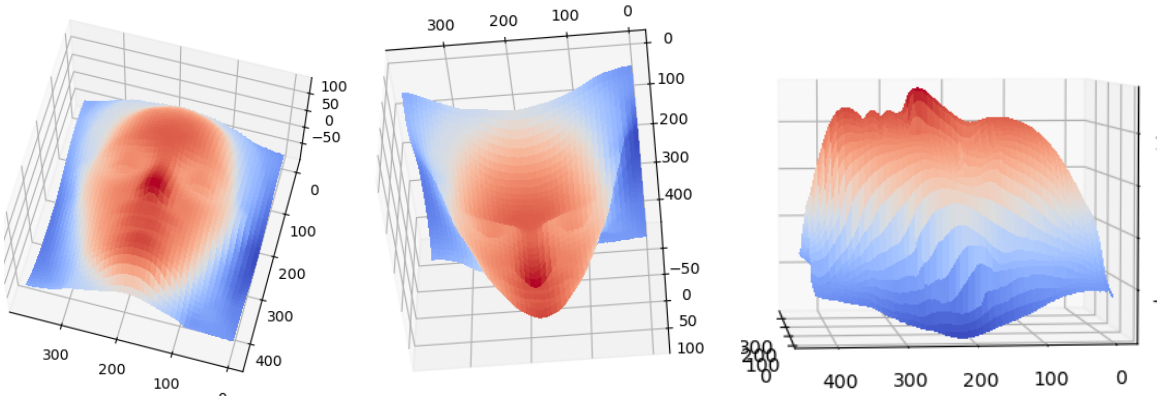
Depending on where we allocate Σ, we could get different L' and B', but still get the same I matrix.

Q2.d)

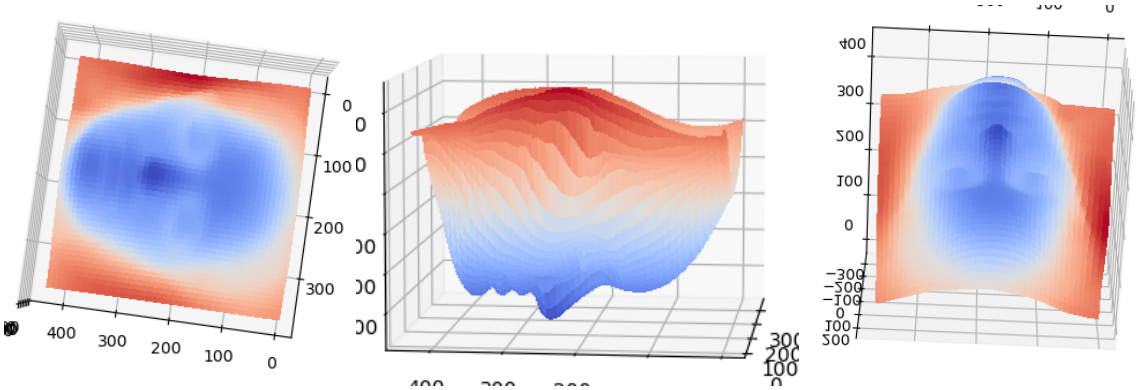
It does not look like a face.



Q2.e)



Surface calibrated with using Σ with normals

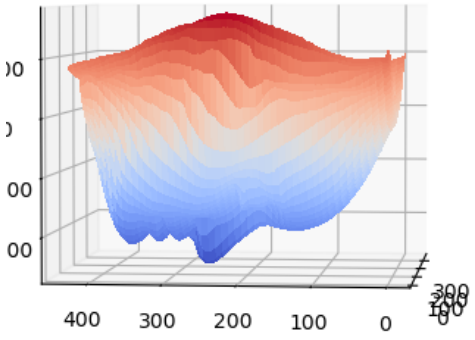


Surface calibrated without using Σ for normals

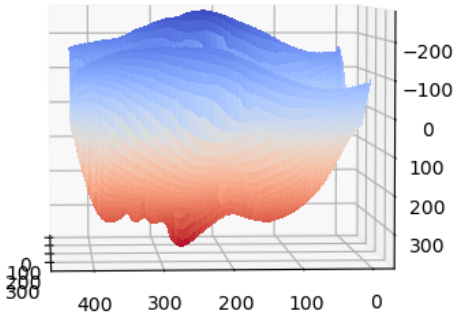
I found that the output looked very similar to the calibrated photometric stereo. However, if I changed how Σ was distributed I did notice that I got an inverted face.

Q2.f).

Modifying λ

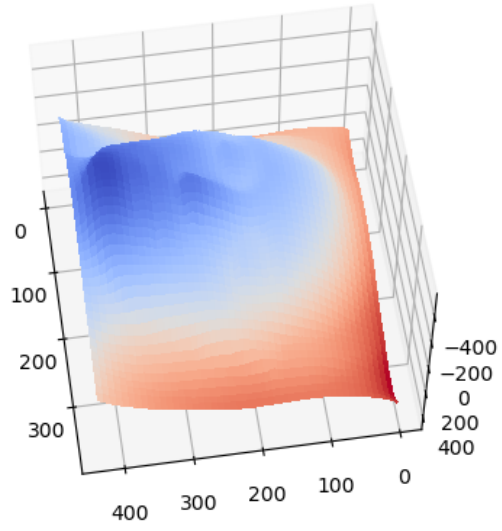


Parameters of $\mu = 0, v = .5, \lambda = .7$

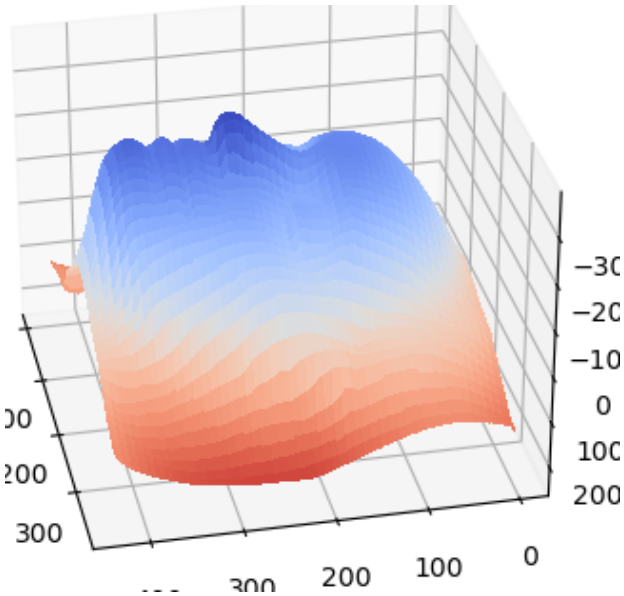


Parameters of $\mu = 0, v = .5, \lambda = -1$

Modifying v

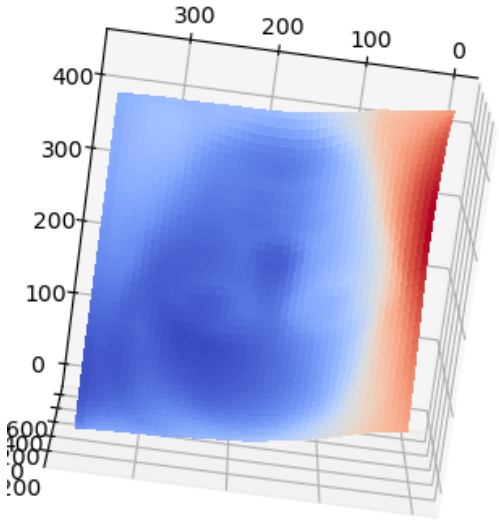


Parameters of $\mu = 0, v = 10, \lambda = 1$

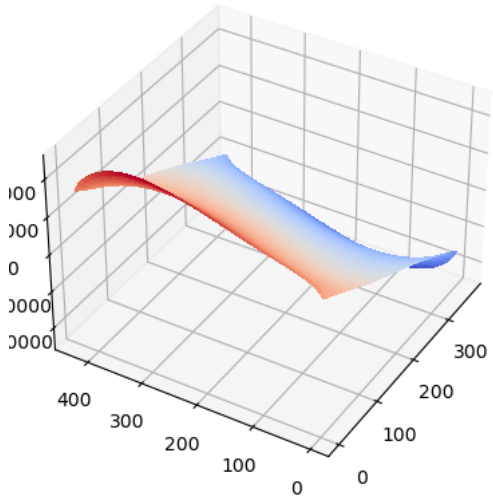


Parameters of $\mu = 0, v = 1, \lambda = 1$

Modifying μ



Parameters of $\mu = 10, v = 1, \lambda = 1$



Parameters of $\mu = 1000, v = 1, \lambda = 1$

It seems that the bas relief is called so due to the fact that from certain angles, using the bas relief does not appear to change the appearance, but when viewed from other angles can be seen as not a full relief, as an almost flattened version of the face. According to the paper, this was named after artisans creating flattened forms called bas-reliefs that looked like full figures when viewed from the right angle, but in reality were not proportional.

λ seemed to invert our surface, while μ and ν seemed to flatten the face in the x and y direction and created a 'bas-relief' in their respective directions. All of these faces are created by the ambiguity from the uncalibrated photometric stereo.

Q2.g

With our bas-relief ambiguity from equation 2, we could try to find a set of parameters, μ, v, λ That minimize the z dimension of the reconstruction. We would have to create a function that minimizes the total magnitude of all z coordinates on the picture.

Our minimization function would have to minimize this equation with respect to μ, v, λ:

$$min \sum^{i} z_i, Frankot(\frac{G^{-T}B}{||G^{-T}B||})$$

Also, from experimenting I found that setting μ or v equal to 1000 gave a very flat surface as a result too.

Q2.h

The more pictures we have the better, although adding images from more lightning directions will ultimately lead to more noise being added at the same time. In the end, no matter how many pictures we have, we will not be able to resolve this ambiguity.

Collaboration

I collaborated with Dante Poe on this assignment.