16720 HW6

vccheng

December 2021

Calibrated Photometric Stereo

1 Question 1a

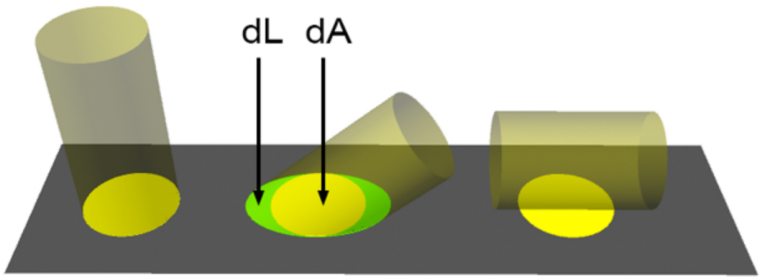
The dot product between the surface normal vector and the lighting vector is equal to the cosine of the angle at which the light hits the surface, and affects how bright it appears.

$$\cos\theta = n \cdot l$$

We denote a small differential area dA as the small area that the beam hits.

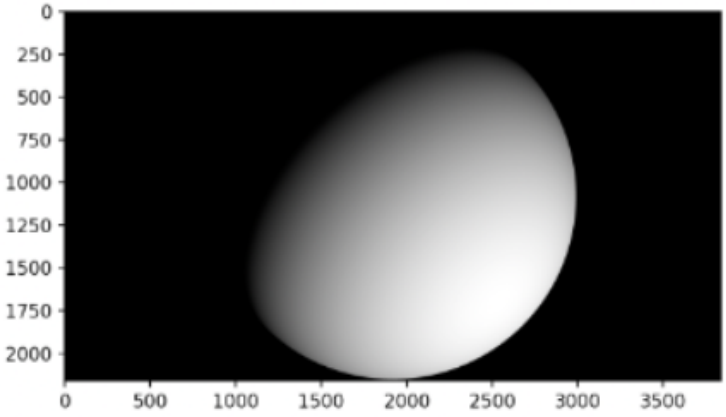
The larger the angle between the light and surface normal gets, the less energy dA receives (the energy will have to be spread over a larger area, so the energy reflected per unit area is smaller).

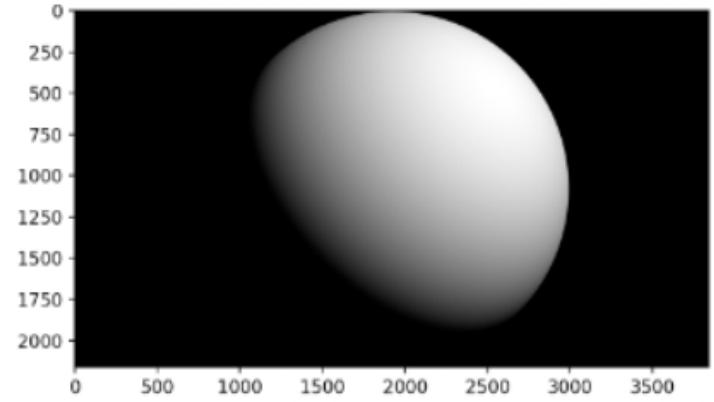
The extreme case where the light is perpendicular to the normal (see right) means that dA receives no light energy, since $\cos(90) = 0$.

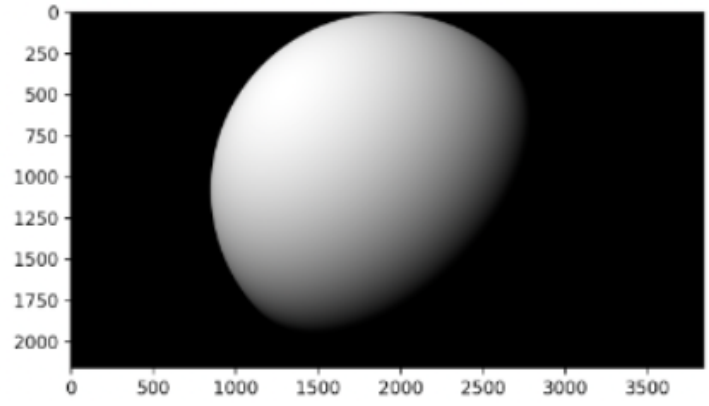


The viewing direction doesn't matter because the amount of light reflected by the surface doesn't change based on where you are looking at it from.

2 Question 1b







3 Question 1c

```
    """
def loadData(path = "../data/"):
    # first image
    im = cv2.imread(f'{path}input_1.tif', cv2.IMREAD_UNCHANGED)
    im = cv2.cvtColor(im, cv2.COLOR_BGR2RGB)
    im = skimage.color.rgb2xyz(im)
    # extract luminance, flatten into vector
    lum = im[:,:,1]
    s = lum.shape
    I = lum.flatten()
    # W(431)*H(369) = 159039 PIXELS * 3 CHANNELS = 477117
    # stack remaining images
    for i in range(1,7):
        im = cv2.imread(f'{path}input_{i+1}.tif', cv2.IMREAD_UNCHANGED)
        im = cv2.cvtColor(im, cv2.COLOR_BGR2RGB)
        im = skimage.color.rgb2xyz(im)
        # assert(im.dtype == np.uint16)
        lum = im[:,:,1]
        I = np.vstack((I,lum.flatten()))
    # divide by max to get relative luminance (range 0->1)
    I = I / np.max(I)
    L = np.load(f'{path}/sources.npy').T # (3x7)
    print('L', L.shape) # (3,7)
    print('I', I.shape) # (7, 159039)
    print('s', s)       # (431, 369)
    return I, L, s
```

4 Question 1d

Rank of I should be 3 because if our image formation model is correct, there will only be three nonzero eigenvalues so Σ will have only three nonzero elements.

Due to noise, shadows, etc, this is not necessarily true in our case (we see that there are more than 3 nonzero eigenvalues). However, we can use SVD to get the best least squares solution.

Singular values:
[136.98374873
22.71905408
15.91655805
4.16789468
2.79029364
2.17978716
1.54252369]

5 Question 1e

Since $L^T B = I$, we can use least squares to solve $Ax = y$, where A is L^T, and y is I.

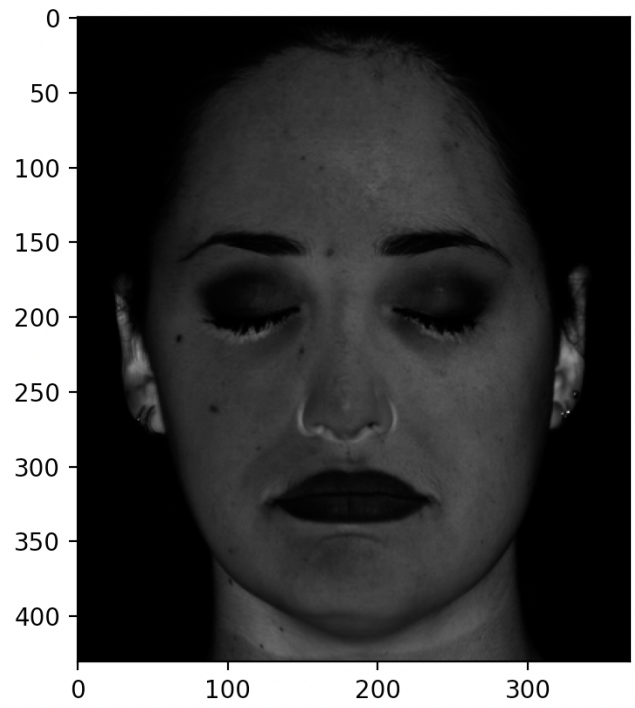
To prevent out of memory errors, I multiplied L by a sparse identity matrix of dimension P by P, to get a matrix of $7P$ by $3P$.

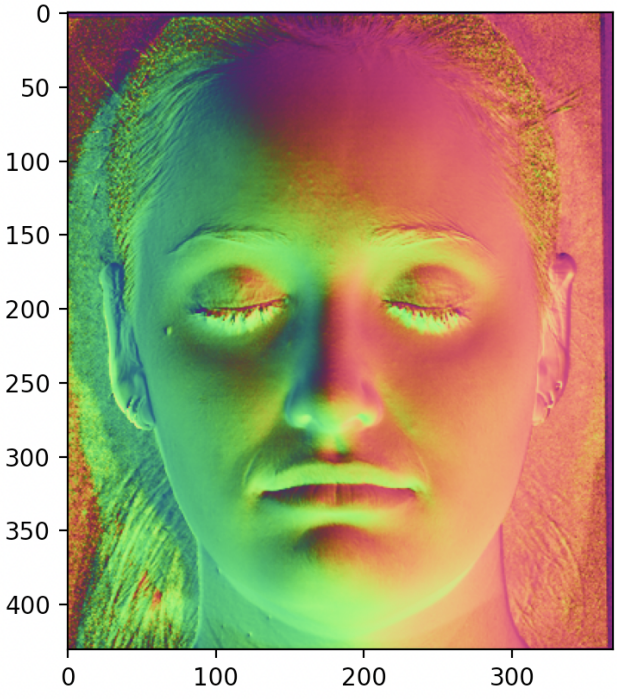
I is reshaped into a $7P$ matrix, and the resulting B will be of dimensions $3P$.

6 Question 1f

The albedo image is interesting because it is dark except the bottom edge of the nose, and the ears.They might be happening because the face is lit up from underneath, causing certain edges to appear bright.

Yes, the normals do match the face curvature, such as the chin, depth of the eyes, contours of the cheeks.





7 Question 1g

Let $z = f(x, y)$ be the depth image at each pixel (x, y).

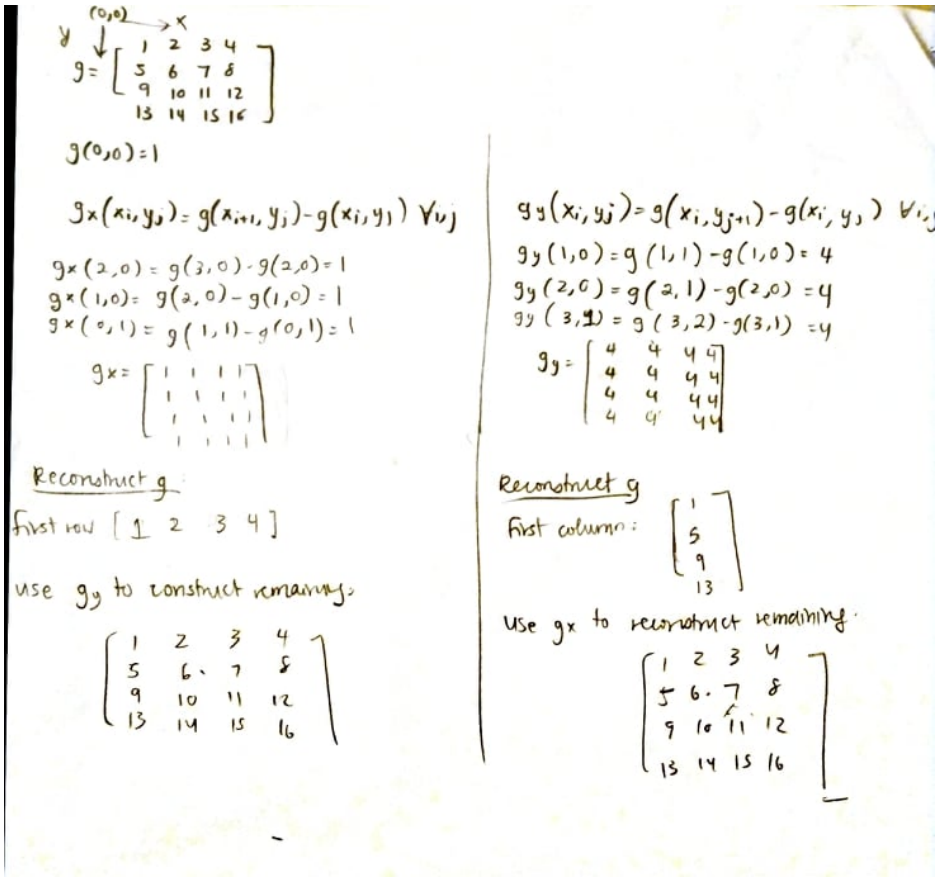
Then the normal is $(df/dx, df/dy, -1)$.

The normal vector n is related to the partial derivatives of f at (x, y).

We can think of a 2D example. For a parabola $y = x^2$, the slope of the tangent is $(1, 2x)$, where $2x = dy/dx$. So as x moves by 1, y moves by $2x$. The normal vector would be $(2x, -1)$, which is perpendicular, meaning it also relates to the partial derivatives in the x, y directions.

Notice that the normal vector to the tangent plane is also normal to the surface.

8 Question 1h



These are the same.

The gradients from part g can be made non-integrable if there is a case where g_x or g_y is undefined due to the normal direction lying on the image plane, which can cause severe numerical instability. Or, if either g_x or g_y involved a square root, in which case there can be multiple answers.

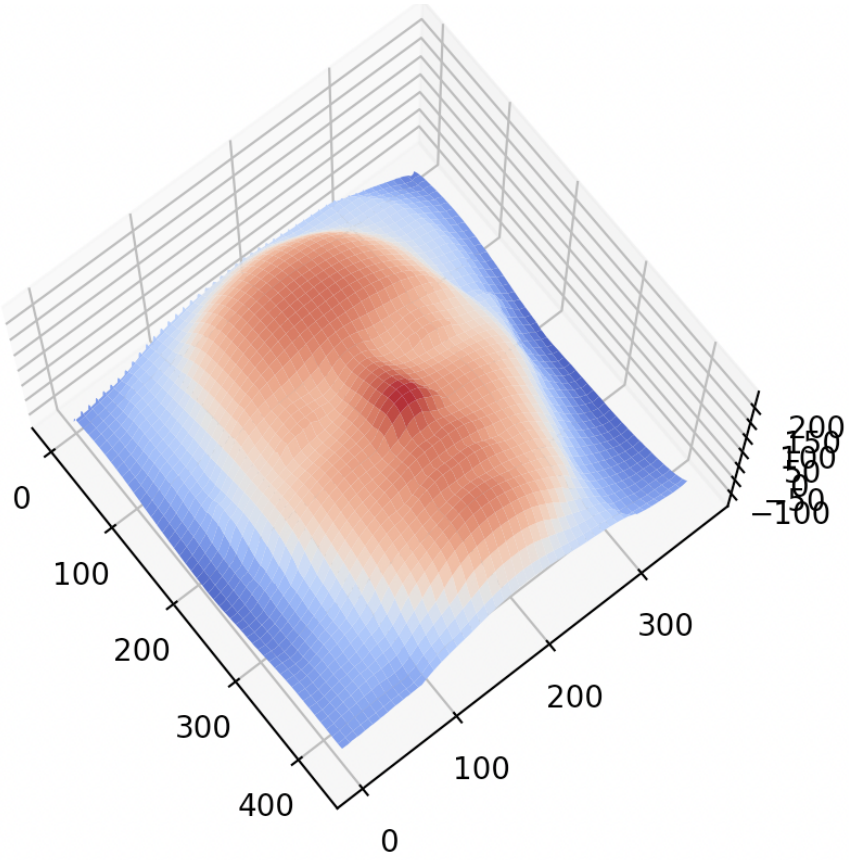
The integrability constraint means that differentiation order shouldn't matter for pseudonormals b (see below).

$$\frac{d}{dy}\frac{b_1(x,y)}{b_3(x,y)} = \frac{d}{dx}\frac{b_2(x,y)}{b_3(x,y)}$$

This means g_x and g_y should yield the same results when reconstructing g.

If we find pseudonormals b_0 from SVD, we can find a transform D such that $b = Db_0$ is the closest to satisfying integrability using least squares. One way is to use Fourier functions to decompose a function into different frequency components, projecting the gradient field onto a set of basis functions.

9 Question 1i

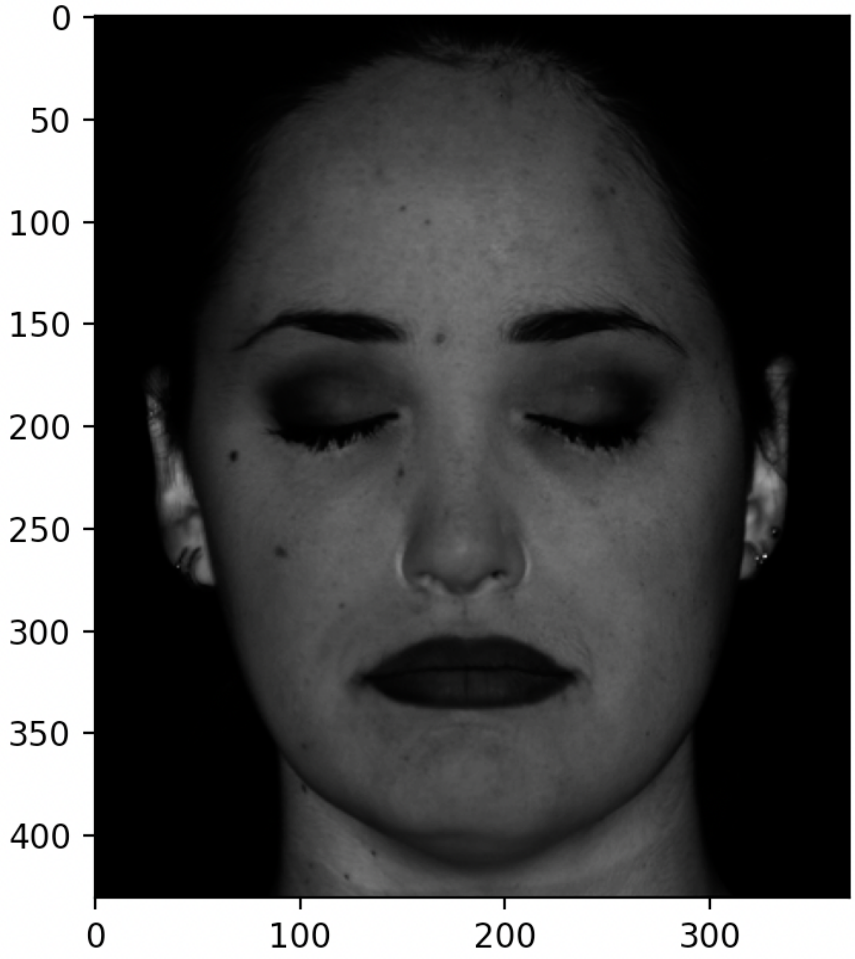


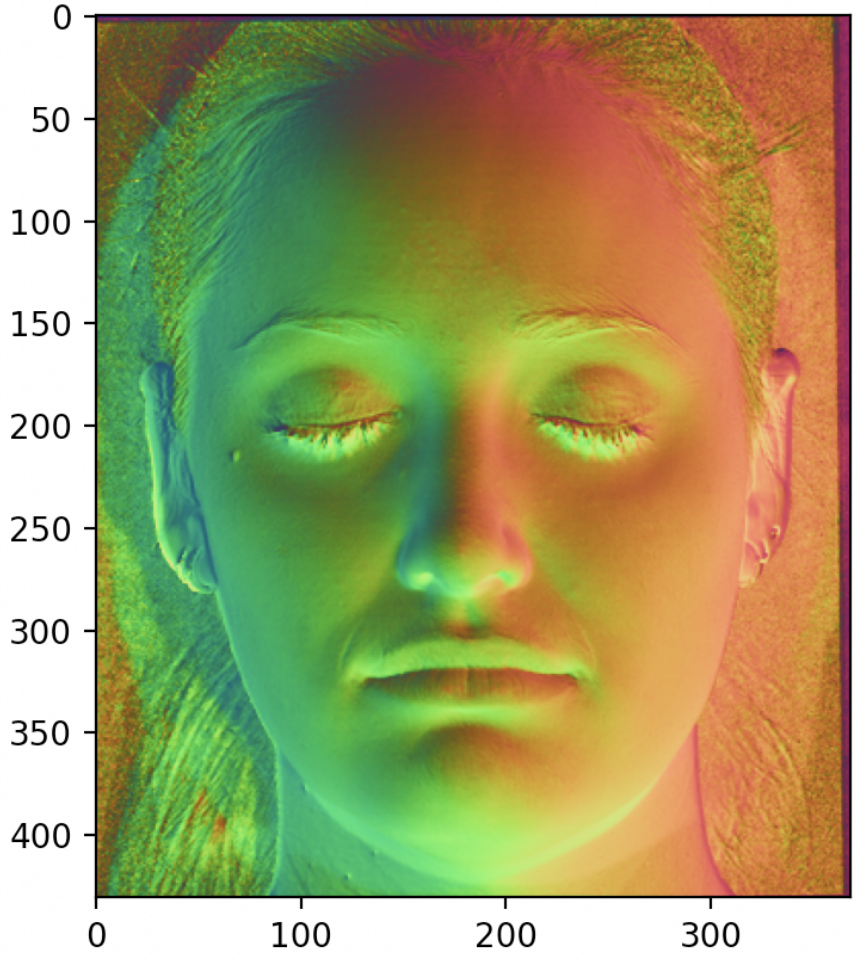
10 Question 2a

We want a rank-3 matrix M. Since $M = U\Sigma V^T$, we can set all singular values except the top 3 to zero. Then, we can slice the first 3 columns of U and first three rows of V^T, so that the new shapes are as follows:
U: (7, 3)
Σ : (3, 3)
V^T: (3, NumPixels)

Then, since $I = LB$, we can factorize I into $(LQ^{-1})(QB)$, where Q is a 3 by 3 matrix.

11 Question 2b





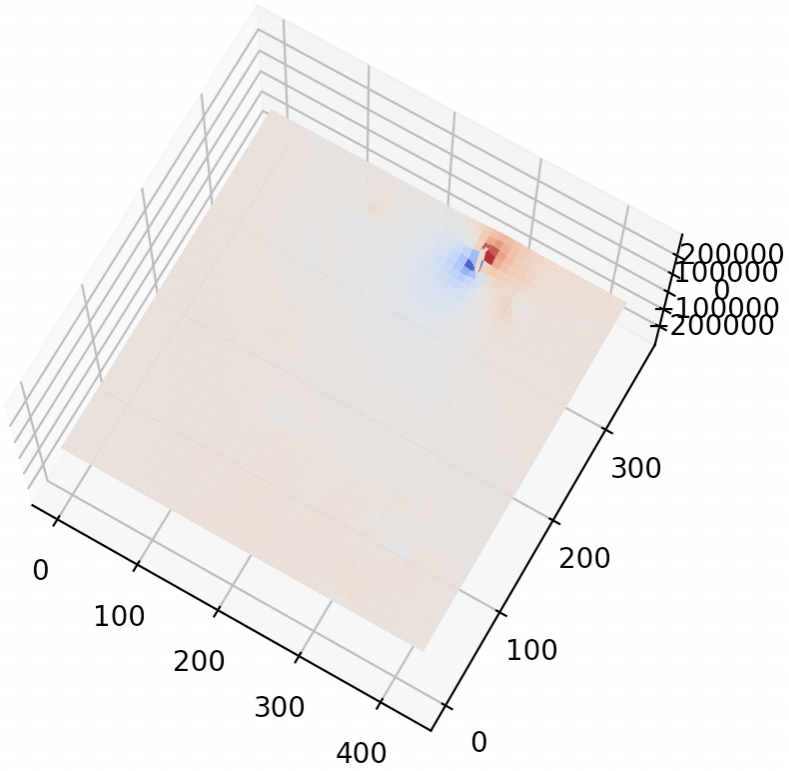
12 Question 2c

By SVD, $I = U\Sigma V^T = U\sqrt{\Sigma}\sqrt{\Sigma}V^T$.
So $L = U\sqrt{\Sigma}$ and $B = \sqrt{\Sigma}V^T$.
This factorization yields the same L matrix.

```
Comparing light matrices
L0 [[-3.93172998  1.24521141  2.46905712]
 [-5.08432705 -3.04415647  1.33298772]
 [-3.16363282  0.65572427  0.56417335]
 [-4.92013395 -0.82242163 -0.0227324 ]
 [-4.71826743  3.05544462 -0.40828866]
 [-4.44935012  0.61229067 -1.19913826]
 [-4.40452175 -1.04145122 -2.47386706]]
L [[-3.93172998  1.24521141  2.46905712]
 [-5.08432705 -3.04415647  1.33298772]
 [-3.16363282  0.65572427  0.56417335]
 [-4.92013395 -0.82242163 -0.0227324 ]
 [-4.71826743  3.05544462 -0.40828866]
 [-4.44935012  0.61229067 -1.19913826]
 [-4.40452175 -1.04145122 -2.47386706]]
```

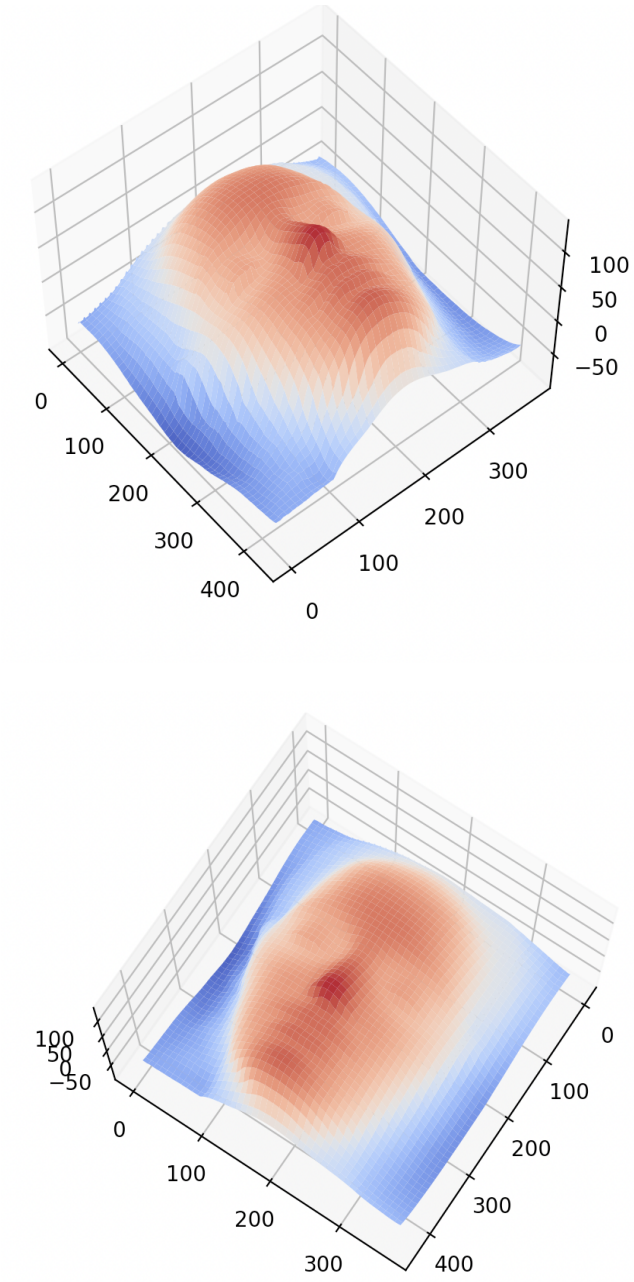
13 Question 2d

No, this does not look like a face.



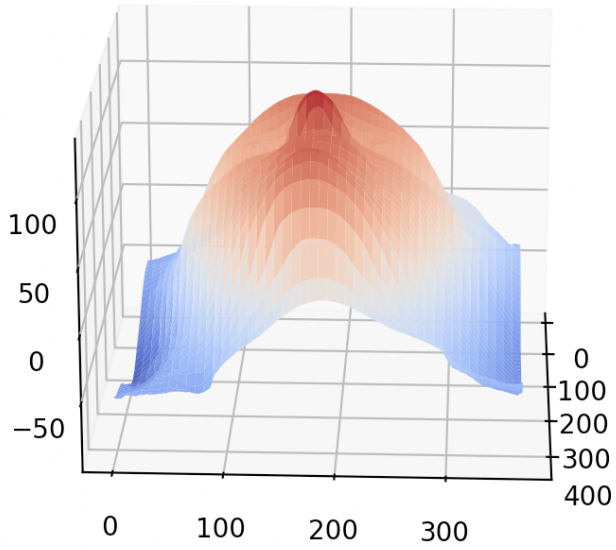
14 Question 2e

Yes, after enforcing integrability this looks like the plot from Part 1.



15 Question 2f

Bas, or "Low" relief means to be "flattened" against the background, i.e. they have clear contours but appear attached to the background plane.

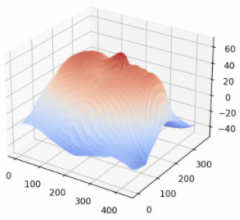
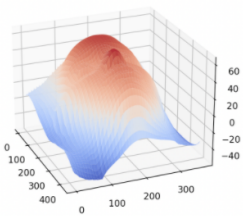
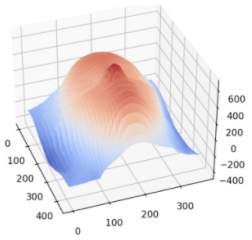
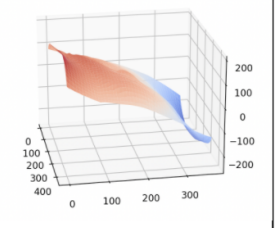
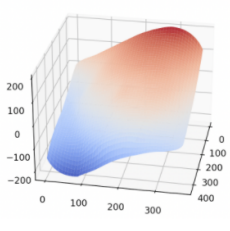
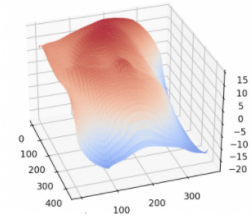
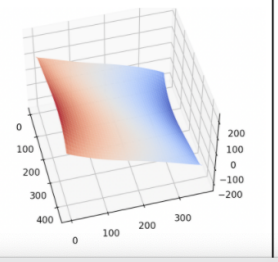
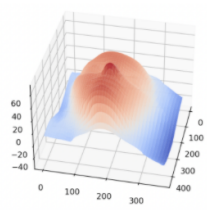
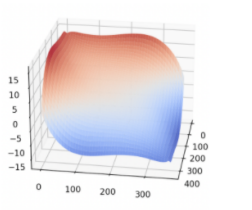


Increasing μ flattens the surface and slants it horizontally in one direction.

Decreasing v seems to make the face slope upwards vertically.

Decreasing λ also flattens the surface, and increasing λ scales up the height of the surface vertically (intensifies the "depth").

I imagine that λ affects $z = f(x, y)$, while v affects y and μ affects x.

μ,v,λ		μ,v,λ		μ,v,λ	
0.5,0.5,0.5		0.5,1,0.5		0.5,0.5,5	
1,0.5,0.5		0.5,5,0.5		0.5,0.5,0.1	
5,0.5,0.5		0.5,0.00 1,0.5		0.5,0.5,0.01	

16 Question 2g

To make the estimated surface as flat as possible, we can decrease λ and also pull out the corners either horizontally/vertically by adjusting v, μ to stretch out the plane and flatten any slopes.

17 Question 2h

Acquiring more lighting pictures for more lighting will not help resolve the ambiguity. Ambiguity cannot be removed by shadows, but can be removed using interreflections (reflections of light from one surface to another surface) or additional assumptions.