Homework 6: Photometric Stereo

Yasser Corzo

December 2023

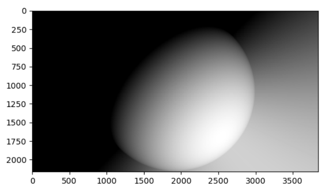
Q1.a

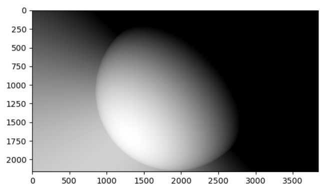
The dot product comes from the geometry of the light source, surface normal, and the viewing direction. Say that the angle between $\vec{l}$ and $\vec{n}$ is θ_i. Also it's important to note that the Lambertian BRDF is simply a constant: $\frac{\rho_d}{\pi} = \frac{kc}{\pi}$ where k is the light source intensity. Hence, by the lambertian case equation:

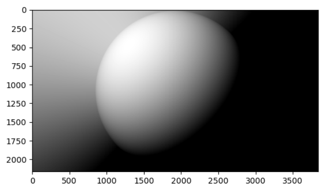
$$I = \frac{\rho}{\pi} kc \cos\theta_i = \rho n * l$$

where I is the image irradiance, ρ is the albedo, n is the surface normal, and l is the light source direction. Viewing direction doesn't matter because surface appears equally bright from all directions. The projected area would be the area that causes the same amount of light to be projected into the area since it is perpendicular to the surface.

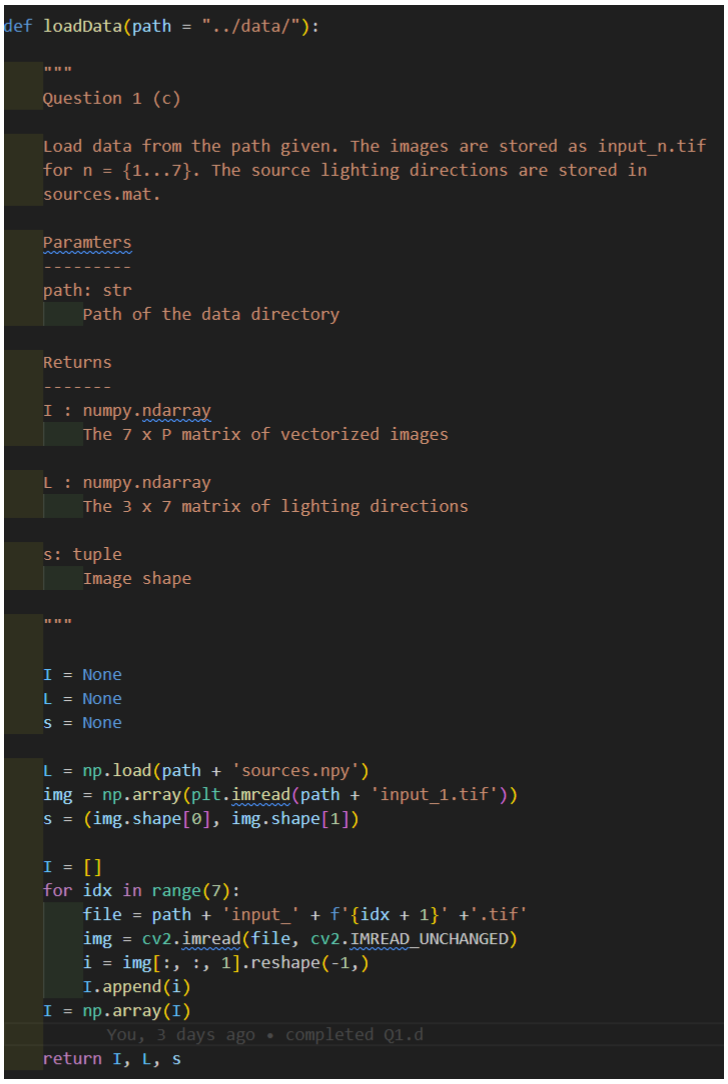
Q1.b


0
250
500
750
1000
1250
1500
1750
2000
0
500
1000
1500
2000
2500
3000
3500


0
250
500
750
1000
1250
1500
1750
2000
0
500
1000
1500
2000
2500
3000
3500


0
250
500
750
1000
1250
1500
1750
2000
0
500
1000
1500
2000
2500
3000
3500

Q1.c



```
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

    L = np.load(path + 'sources.npy')
    img = np.array(plt.imread(path + 'input_1.tif'))
    s = (img.shape[0], img.shape[1])

    I = []
    for idx in range(7):
        file = path + 'input_' + f'{idx + 1}' +'.tif'
        img = cv2.imread(file, cv2.IMREAD_UNCHANGED)
        i = img[:, :, 1].reshape(-1,)
        I.append(i)
    I = np.array(I)
          You, 3 days ago • completed Q1.d
    return I, L, s
```

Q1.d

Rank of I should be 3 because since B has size 3 x P, and by definition rank is the number of linearly independent columns of a matrix, the rank of B can be at most P. In the case of L^T, the shape of L^T is 7 x 3, hence by definition the rank of L^T can be at most 3. Since I is the product of L^T and B, the rank of I can be at most the minimum of rank(L^T) & rank(B). Hence rank of I should be 3.

```
Singular values of I:
[72.40617702 12.00738171  8.42621836
2.23003141  1.51029184  1.17968677
0.84463311]
```

The singular values do not agree with the rank-3 requirement because the top 3 singular values should be non-zero, while every other value should be 0.

Q1.e

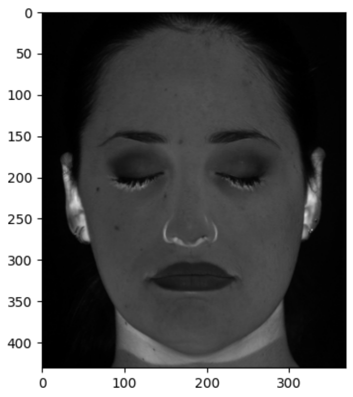
How I constructed matrix A and vector y was in the following ways:

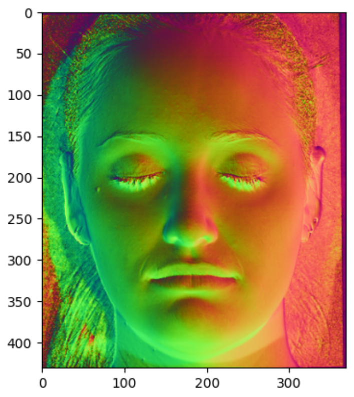
$$A = L * L^T$$

$$y = L * I$$

Applying the least squares algorithm to A and y would solve for B.

Q1.f





Unusual features that I can discern from the albedo are outline of white areas in the ears, edges of nose and the upper half of the neck. This could be a result of noise or unique features on the face that cause different reflective properties of light.
The normals do match my expectation of the curvature of the face. Normals that are pointing directly toward the viewer are more consistent in areas where light reflects directly up to the viewer, i.e. brighter areas on the face. Where there's shadows or darker areas, normals are pointing in other directions.

Q1.g

Since the implicit form of a surface is: $z = f(x, y)$, the explicit form of a surface is: $F(x, y, z) = z - f(x, y) = 0$. The gradient of F is as follows:

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

We know that ∇F is orthogonal to the surface at a point, and so is the normal, hence both the normal and gradient are parallel and proportional. Since $n = (n_1, n_2, n_3)$

$$n_1 = a * -\frac{\partial f}{\partial x} \tag{1}$$

$$n_2 = a * -\frac{\partial f}{\partial y} \tag{2}$$

$$n_3 = a * 1 = a \tag{3}$$

Since $n_3 = a$, plug in n_3 into the other two equations:

$$n_1 = n_3 * -\frac{\partial f}{\partial x}$$

$$n_2 = n_3 * -\frac{\partial f}{\partial y}$$

Hence

$$\frac{\partial f}{\partial x} = -\frac{n_1}{n_3}$$
$$\frac{\partial f}{\partial y} = -\frac{n_2}{n_3}$$

Q1.h

1. Following the partial derivative formula g_x:

$$g_x(0,0) = g(1,0) - g(0,0) = 2 - 1 = 1$$

$$g_x(1,0) = g(2,0) - g(1,0) = 3 - 2 = 1$$

$$g_x(2,0) = g(3,0) - g(2,0) = 4 - 3 = 1$$

   Hence $g(x_{i+1}, y_j) = g(x_i, y_j) - g_x(x_i, y_j)$. Using this arranged formula, the first row of g would be:

$$g[0,:] = [1,2,3,4]$$

   Using g_y (similar to formula for g_x) and $g_y(x_i, y_{j+1})$ (similar to $g(x_{i+1}, y_j)$),

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

2. Following the partial derivative formula g_y:

$$g_y(0,0) = g(0,1) - g(0,0) = 5 - 1 = 4$$

$$g_y(0,1) = g(0,2) - g(0,1) = 9 - 5 = 4$$

$$g_y(0,2) = g(0,3) - g(0,2) = 13 - 9 = 4$$

   Hence $g(x_i, y_{j+1}) = g(x_i, y_j) - g_y(x_i, y_j)$. Using this arranged formula, the first column of g would be:

$$g[:,0] = \begin{bmatrix} 1 \\ 5 \\ 9 \\ 13 \end{bmatrix}$$

   Using g_x (similar to formula for g_y) and $g(x_{i+1}, y_j)$ (similar to $g_y(x_i, y_{j+1})$),

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

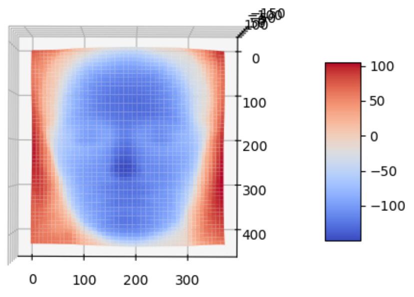
One way to modify g_x and g_y to be non-integrable would be to add a multiplication factor m, k and a constant such that

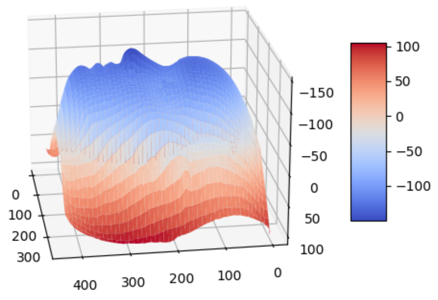
$$g_y(x_i, y_j) = g(x_i, y_j) - m * g(x_i, y_{j+1})$$

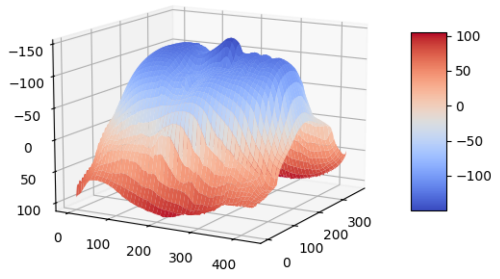
$$g_x(x_i, y_j) = g(x_i, y_j) - k * g(x_{i+1}, y_j)$$

Specifically, m and k can be values that change with every row/column. This way, the difference between values across columns/rows is not constant, therefore yielding different values depending with which derivative to start to reconstruct g.

Q1.i


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

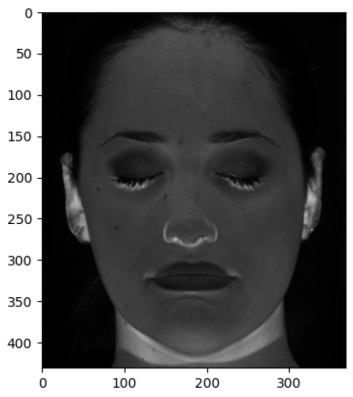

0
100
200
300
400
300
200
100
0
−150
−100
−50
0
50
100
100
50
0
−50
−100


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

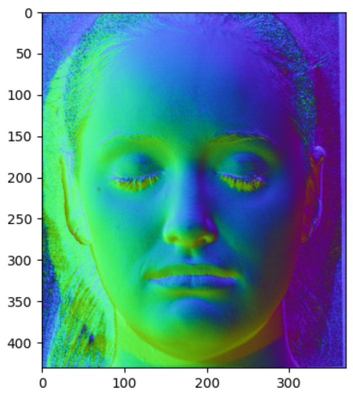
Q2.a

This can be used to construct a factorization of the form detailed above following the required constraints by first performing SVD on I. Since we want the best rank-k approximation of I, we set the top K singular values to non-zero values while the other singular values are 0. Since the top k singular values correspond to the first K columns of U and the first K rows of V^T, $B = V^T[:3, :]$ and $L = U[:, :3]^T$.

Q2.b



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



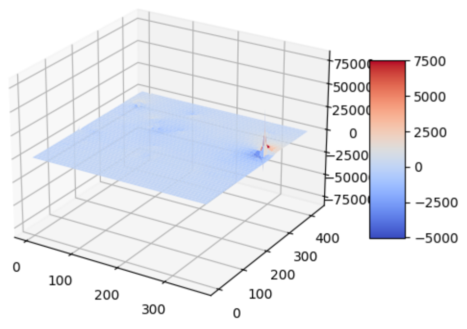
0
50
100
150
200
250
300
350
400
0
100
200
300

Q2.c

```
L = [[-0.1418  0.1215 -0.069   0.067  -0.1627  0.      0.1478]
 [-0.1804 -0.2026 -0.0345 -0.0402  0.122   0.1194  0.1209]
 [-0.9267 -0.9717 -0.838  -0.9772 -0.979  -0.9648 -0.9713]]
L_hat = [[-0.35443006 -0.39939468 -0.3246755  -0.40229751 -0.39346212 -0.38475709
  -0.38046721]
 [ 0.35010833 -0.63368646  0.17832041 -0.19499732  0.56425025  0.09046008
  -0.28192575]
 [ 0.63269835  0.3939417   0.10536782 -0.0166135  -0.20183754 -0.33103324
  -0.5317922 ]]
```

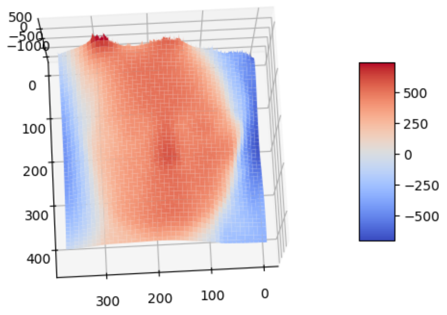
From above, L_0 and $\hat{L}$ are different.
The small change I would make to my approach in part a that would change $\hat{B}$ and $\hat{L}$ and keep the image rendered would be to not limit $\hat{B}$ and $\hat{L}$ to the top 3 rows and columns of U and V respectively. Instead, just set the other singular values of S (that are not top 3) to 0, multiply this new S with the U and V values calculated via SVD of I. This will yield $\hat{I}$. Then perform SVD on $\hat{I}$, yielding $\hat{U}$, $\hat{S}$, and $\hat{V}$ which will have new dimensions since $\hat{S}$ will only contain the 3 singular values, hence $\hat{U}$ and $\hat{V}$ will only contain the corresponding singular vectors and columns, thus changing $\hat{B}$ and $\hat{L}$.

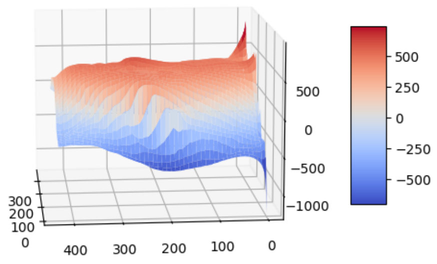
Q2.d

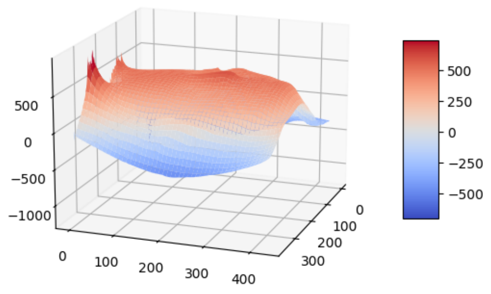


The surface reconstructed by the Frankot-Chellappa algorithm does not look like a face.

Q2.e

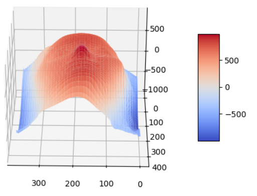




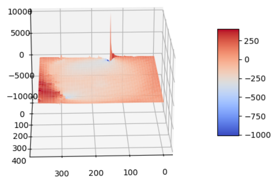


The surface looks similar, but not exactly like the one from calibrated photometric stereo. The range of depth is different. The tip of the nose is in the negative z value range, while in uncalibrated its in the positive z range. The facial features in calibrated is much more detailed and defined, such as the nose and mouth. In uncalibrated, the shape of these facial features are less defined, more smooth.

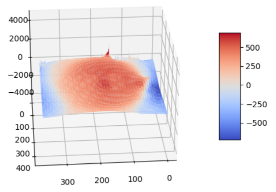
Q2.f



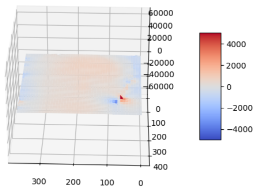
(a) $\lambda = 0.5$



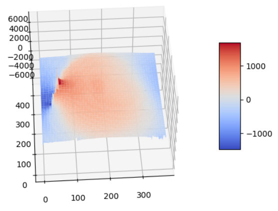
(b) $\lambda = 3$



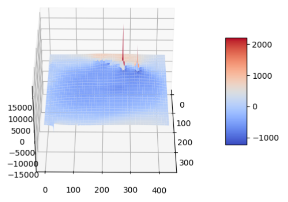
(c) $u = 0.5$



(d) $u = 1.5$



(e) $v = 0.5$



(f) $v = 1.5$

I believe the bas-relief ambiguity is so named because, as shown in the images above, there are limitations in the reconstruction process of 3D images from sets of 2D images under different lighting conditions. Changing parameters u, v, and λ does decrease this limitation by an extent. For example, in the top row of the figure above where $lambda$ is changed, the shape of the face does change at $\lambda = 0.5$ and produces a more defined face than the image produced in part 2.e. The nose, mouth, and eyes are more defined. However, increasing λ too much does change the shape drastically where at some points the gradient increases drastically, producing huge z values at some coordinates. Parameters u and v seem to have the greatest affect in tilting (to an extent) and flattening the plane.

Q2.g

To make a transformation that flattens the shape as much as possible, I would focus on decreasing the *lambda* parameter and increasing u and v, as shown in the figure in part 2.f.

Q2.h

Acquiring more pictures from more lighting directions can help resolve the ambiguity because with more lighting directions, we provide the lambertian algorithm with more information about how light interacts with the object's surface. With more images captured from different angles, the algorithm has a better chance of capturing unique features of a surface and how reflection is affected, thus increasing the accuracy of the reconstruction process.