Advanced Computer Vision HW5

Ricky Yuan
rickyy@andrew.cmu.edu

November 21, 2024

1 Calibrated photometric stereo

1.1 a

The dot product represents the cosine angle between the surface normal and the light direction, which measures how aligned the light direction is with the surface normal.
The projected surface represents how much of the surface is effectively illuminated, which depends on $n \cdot l$. Since a Lambertian surface ensures that light reflects equally in all directions, the viewing direction does not matter and the observed intensity is solely depends on the $n \cdot l$.

1.2 b

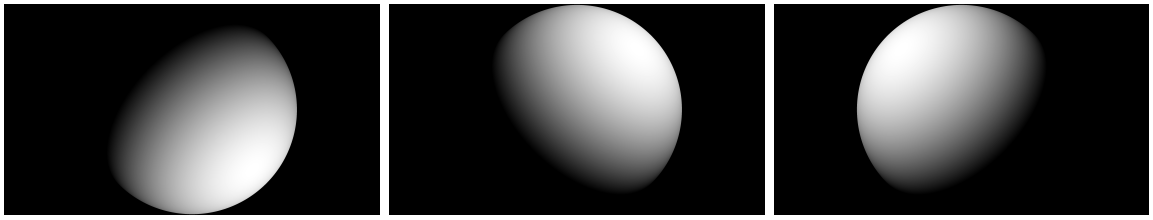


Figure 1: Rendering Results

```
def renderNDotLSphere(center, rad, light, pxSize, res):
    [X, Y] = np.meshgrid(np.arange(res[0]), np.arange(res[1]))
    X = (X - res[0] / 2) * pxSize * 1.0e-4
    Y = (Y - res[1] / 2) * pxSize * 1.0e-4
    Z = np.sqrt(rad**2 + 0j - X**2 - Y**2)
    X[np.real(Z) == 0] = 0
    Y[np.real(Z) == 0] = 0
    Z = np.real(Z)

    # Normals
    N = np.stack((X, Y, Z), axis=-1)  # Shape: (2160, 3840, 3)
```

```
    # n-dot-l shading
    image = np.maximum(0, np.sum(N * light, axis=-1))
    return image
```

1.3 c

```
def loadData(path="../data/"):
    I = []
    L = np.load(path + "sources.npy").T  # (3, 7)

    for i in range(7):
        img = plt.imread(f"{path}input_{i+1}.tif").astype(np.uint16)
        img = rgb2xyz(img)
        img = img[:, :, 1]  # 1 is the luminance channel
        s = img.shape
        img = img.flatten()
        I.append(img)

    I = np.array(I)  # (7, P)
    return I, L, s
```

1.4 d

```
# Part 1(d)
U, S, V = np.linalg.svd(I, full_matrices=False)
print("Singular values:", S)
rank = np.sum(S > 1e-5)
print(f"Estimated rank of I: {rank}")
```

The output:

```
Singular values: [79.18619979 13.17407209  9.2290384   2.42091186
   1.62443194  1.27069921 0.89704252]
```

The rank of I should be 3 because we need 3 DoF to describe a surface normal. However, the singular values have 7 non-zero values. That is probably because we have input images from 7 different directions, which is more than required.

1.5 e

Let $A = L^T$, $\mathbf{x} = B$, and $\mathbf{y} = I$, I directly called `np.linalg.lstsq` to solve the least square problem $|L^T B - I|^2$.

```
def estimatePseudonormalsCalibrated(I, L):
    # Solve the linear system using least squares: argmin_B ||L^T B - I
       ||^2
    B = np.linalg.lstsq(L.T, I, rcond=None)[0]  # rcond=None uses default
       machine precision cutoff
    return B
```

1.6 f

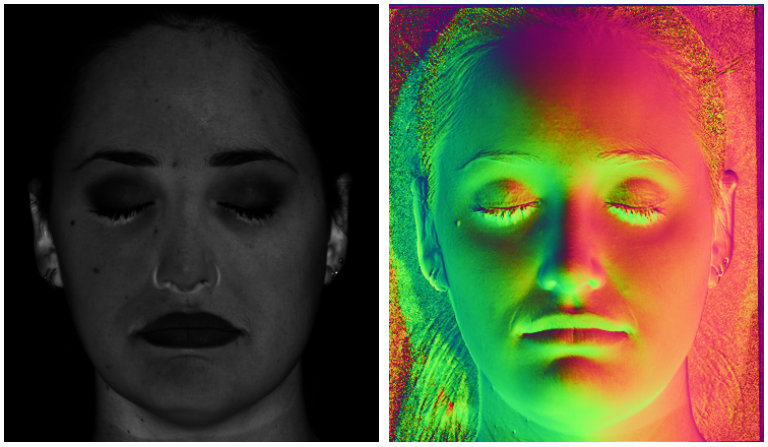


Figure 2: Albedos and normals

Overall, both the albedo and the normal image I generated match my expectation. However, I noticed that there are some bright part around the nose, neck, and ear in the albedo image, and the overall image is darker. I think it is probably because we did not consider the effect of shadow, causing some wrong estimation of these areas.

1.7 g

The partial derivative of f is $f_x = \frac{\partial f(x,y)}{\partial x} = z_{x+1,y} - z_{x,y}$, therefore the direction of the gradient is $\mathbf{g}_x = (1, 0, z_{x+1,y} - z_{x,y})$. We know that the gradient is perpendicular to the surface normal, we then have the following relation:

$$\mathbf{n} \cdot \mathbf{g}_x = (n_1, n_2, n_3) \cdot (1, 0, z_{x+1,y} - z_{x,y}) = n_1 + n_3 \cdot (z_{x+1,y} - z_{x,y}) = 0$$

Substitute $z_{x+1,y} - z_{x,y}$ with f_x, we have:

$$f_x = \frac{\partial f(x,y)}{\partial x} = z_{x+1,y} - z_{x,y} = \frac{-n_1}{n_3}$$

Similarly, we can derive the relation for y:

$$f_y = \frac{\partial f(x,y)}{\partial y} = z_{x,y+1} - z_{x,y} = \frac{-n_2}{n_3}$$

1.8 h

Gradient with respect to x, y:

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

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

Reconstruction g from either x or y gives us the same result.

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

By definition, this is integrable and corresponds to a true surface. If we add some noise, or add some values along a gradient direction, the result would be non-integrable.

1.9 i

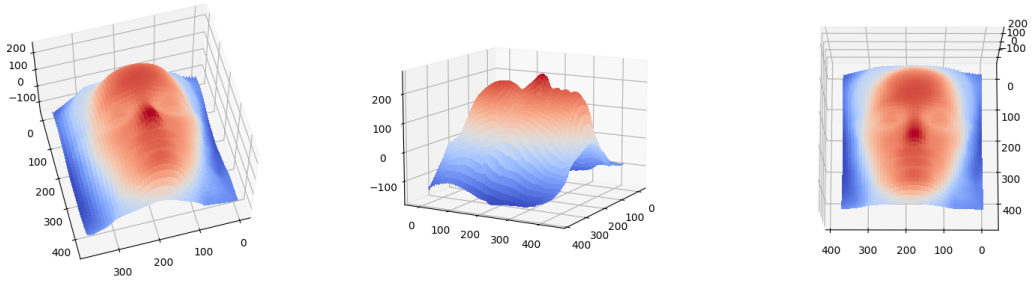


Figure 3: 3D reconstruction result from calibrated photometric stereo

```python
def estimateShape(normals, s):
    normals = normals.T.reshape((s[0], s[1], 3))  # (2160, 3840, 3)
    fx = -1.0 * normals[:, :, 0] / normals[:, :, 2]  # dz/dx
    fy = -1.0 * normals[:, :, 1] / normals[:, :, 2]  # dz/dy

    surface = integrateFrankot(fx, fy)
    return surface
```

2 Uncalibrated photometric stereo

2.1 a

Since I is known, we can do SVD to I to construct the $U\Sigma V^T$ matrices. Because we know that the rank of the reconstructed $\hat{I}$ is 3, we select the top $k = 3$ from Σ and their corresponding $\hat{U}$ and $\hat{V}^T$. Therefore we form a 3 x P matrix $B = \hat{V}^T$, and a 7 x 3 matrix $L = \hat{U}$, where P is the number of points.

2.2 b

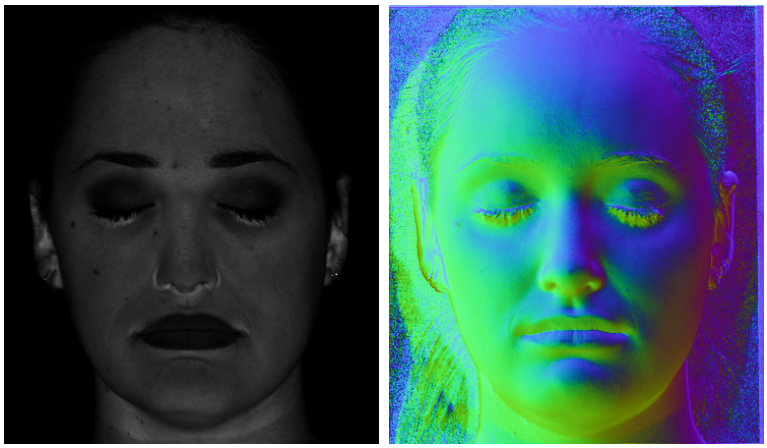


Figure 4: Estimated Albedos and normals

2.3 c

```
L:
[[-0.33572973 -0.43475204 -0.26979124 -0.42054946 -0.40319422 -0.38011231
   -0.37626599]
 [ 0.26078278 -0.63869389  0.13728144 -0.17231364  0.64143508  0.12875076
   -0.21796455]
 [ 0.618832    0.33379549  0.1414761  -0.00576238 -0.10159911 -0.3004124
   -0.62049081]]
L0:
[[-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]]
```

As we can see, the two matrices are pretty different. One way to change the $\hat{L}$ and $\hat{B}$ while keeping the images rendered is to normalize them.

2.4 d

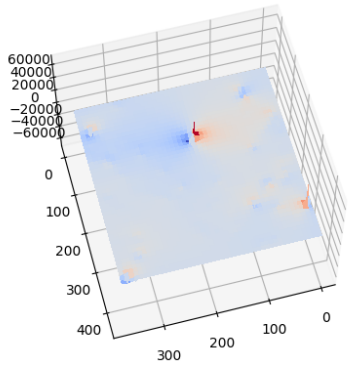


Figure 5: Reconstructing the shape, attempt 1

It does not look like a face due to the integrability issue.

2.5 e

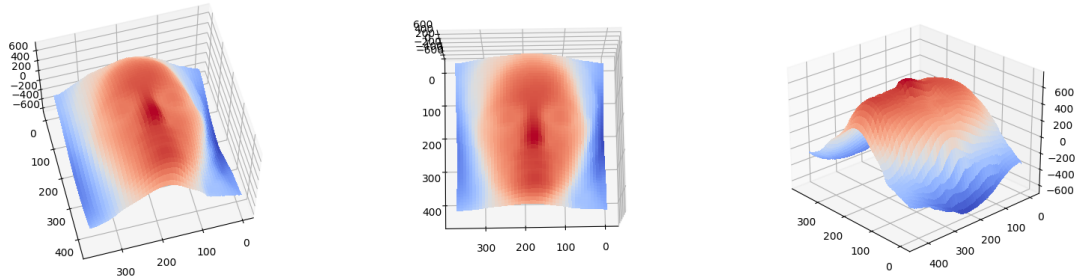
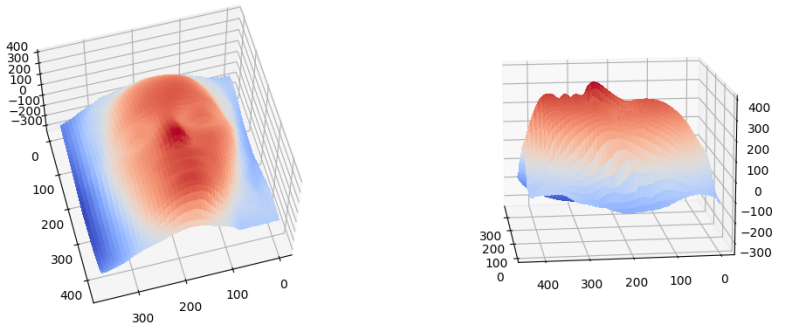


Figure 6: Reconstructing the shape, attempt 2

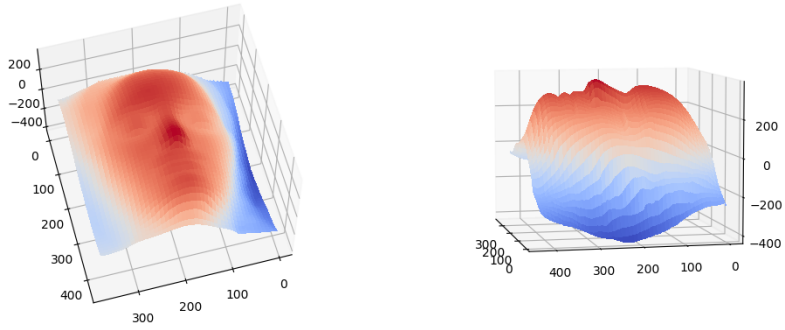
```
# GBR (generalized bas-relief) transform matrix
G = np.array([[1, 0, 0], [0, 1, 0], [0, 0, -1]])
B = enforceIntegrability(B, s)
B = np.linalg.inv(G.T) @ B
albedos, normals = estimateAlbedosNormals(B)
surface = estimateShape(normals, s)
plotSurface(surface)
```

The reconstructed shape after applying GBR looks like a face now.

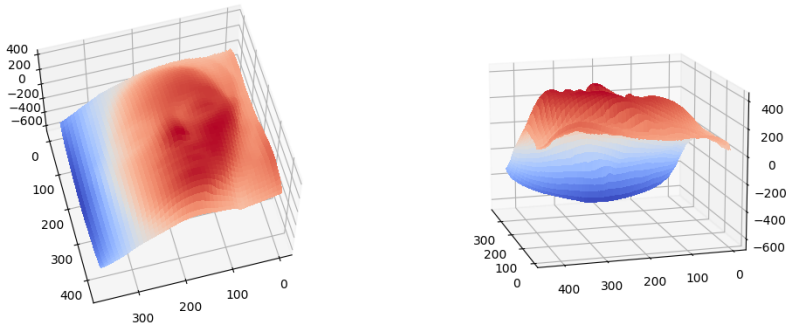
2.6 f



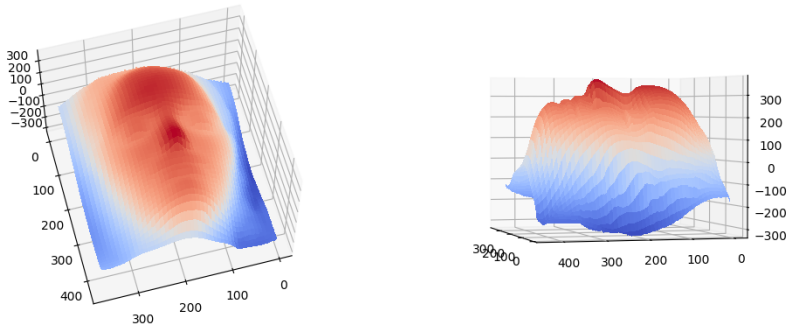
(a) $\mu = 3$



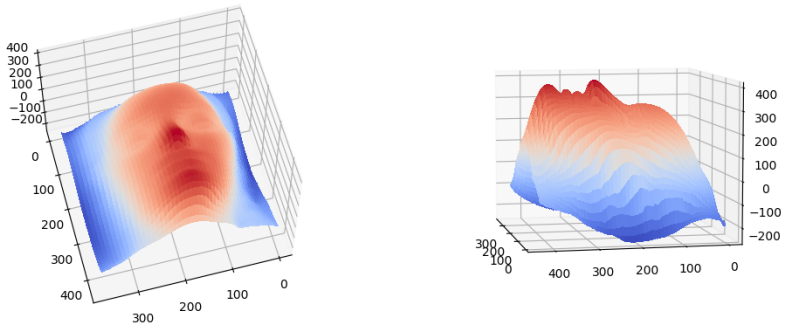
(b) $\mu = -3$



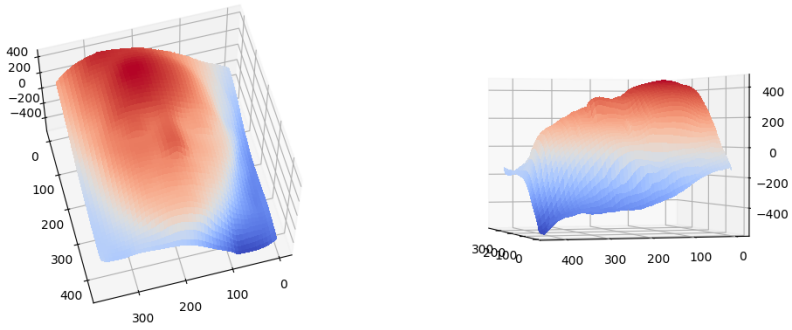
(c) $\mu = 10$



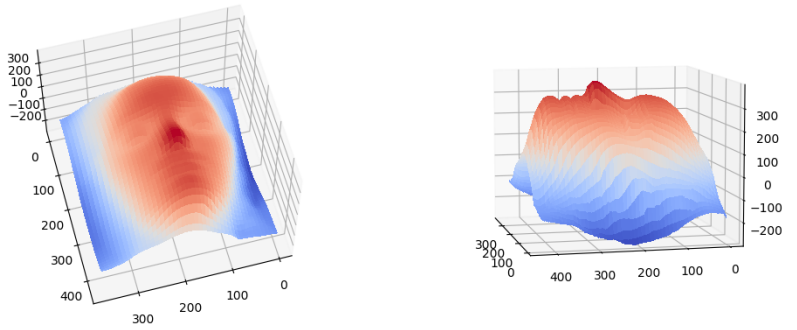
(d) $\nu = 3$



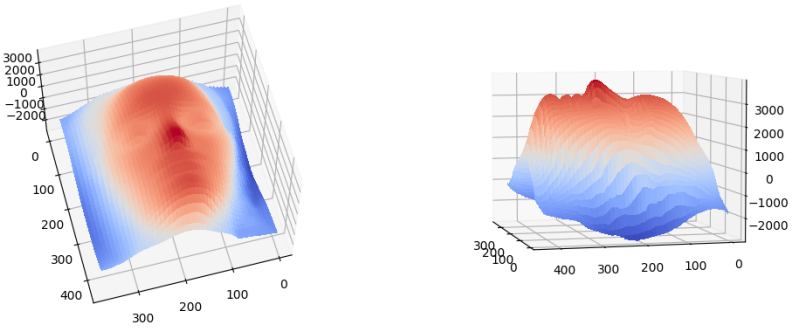
(e) $\nu = -3$



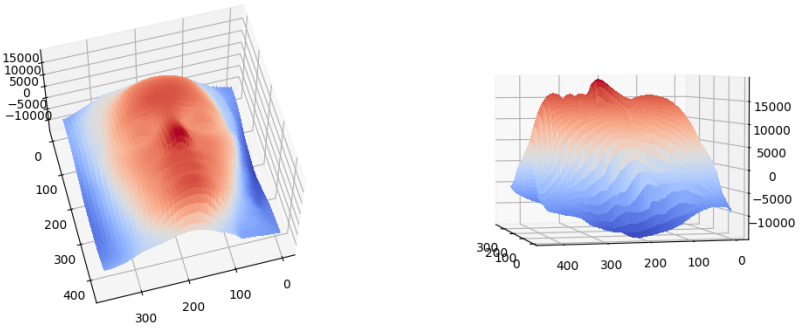
(f) $\nu = 10$



(g) $\lambda = 3$



(h) $\lambda = -3$



(i) $\lambda = 10$

For μ and ν, they controls the amount of "flatness" along some directions. The shape becomes more flat if we increases those values. λ controls the scale of the height of the shape. Also, negating the λ turns the shape upside down. The bas-relief ambiguity is named from a sculptural term, describing that depth variations are altered while maintaining consistent visual appearance from a particular perspective, causing ambiguity.

2.7 g

From my experiments, we can choose larger μ, ν, and choose λ close to zero.

2.8 h

No, the ambiguity is not caused by lacking of lighting directions. We already have 7 directions of light while it only needs 3, so acquiring more pictures would not help solving this ambiguity.

3 Collaboration

For this homework, I discussed with Tianzhi Li (tianzhli).