16720 — PS6

Connor W. Colombo (cwcolomb)

December 3rd 2020

1 Calibrated photometric stereo

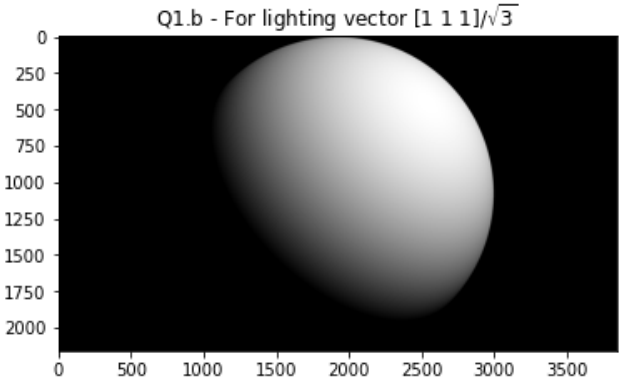
Q1.a Understanding n-dot-l lighting

In Figure (2a) provided in the assignment, $\overrightarrow{n}$ is the normal vector to infinitesimal patch of the surface dA, $\overrightarrow{l}$ is the lighting direction (vector pointing to the light source), and $\overrightarrow{v}$ is the direction to an observer. Notably, for a Lambertian surface, light is scattered with uniform brightness in all directions — the closer the lighting angle is to the normal, the brighter the surface appears (from all directions). The dot product comes into play because the formal expression of this relation is that the brightness observed from any angle is proportional to the cosine of the angle between the normal and the lighting direction; since, $\overrightarrow{n} \cdot \overrightarrow{l} = \cos(\theta_{\overrightarrow{n},\overrightarrow{l}})||\overrightarrow{n}||||\overrightarrow{l}||$, $\cos(\theta_{\overrightarrow{n},\overrightarrow{l}}) = \frac{\overrightarrow{n}\cdot\overrightarrow{l}}{||\overrightarrow{n}||||\overrightarrow{l}||}$, thus the observed surface brightness is proportional to $\frac{\overrightarrow{n}\cdot\overrightarrow{l}}{||\overrightarrow{n}||||\overrightarrow{l}||}$ or just the dot product $\overrightarrow{n} \cdot \overrightarrow{l}$ if $\overrightarrow{n}$ and $\overrightarrow{l}$ are normalized. The reason for this relation is that as $\theta_{\overrightarrow{n},\overrightarrow{l}}$, the angle between $\overrightarrow{n}$ and $\overrightarrow{l}$, increases, the projected area of the cone of light onto the surface grows and is spread out over the surface and thus a smaller proportion of the light falls onto dA (and is received by the surface). Basic projection math says that this area is just $\cos(\theta_{\overrightarrow{n},\overrightarrow{l}})dA$ and thus observed intensity is $I \propto \cos(\theta_{\overrightarrow{n},\overrightarrow{l}})dA$. In this case, the viewing direction doesn't matter because Lambertian surfaces assume matte/body reflection where light enters the surface slightly and then emerges being scattered uniformly in all directions.

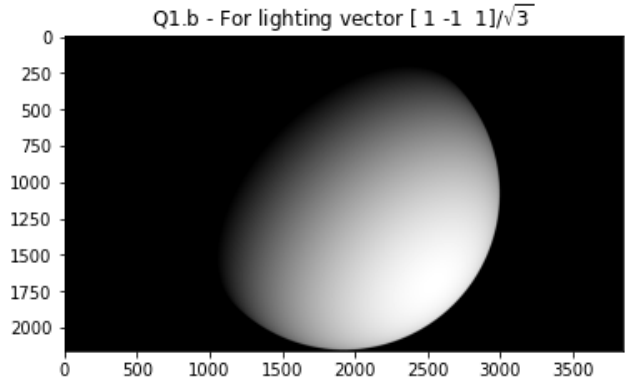
Q1.b Rendering n-dot-l lighting

Notes on assumptions and formatting:

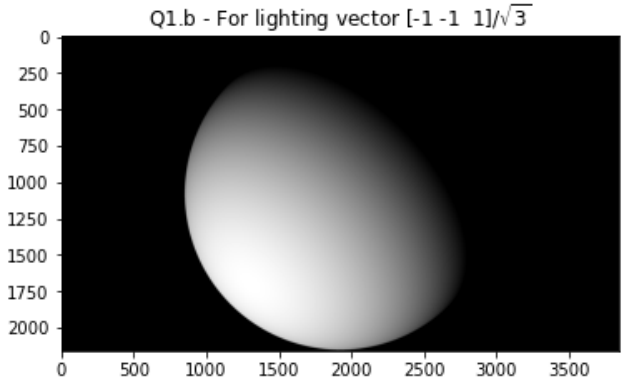
- Images were generated assuming an orthographic camera and in accordance with Figure 3 in the assignment. The code took into account the fact that the center of the coordinate system should be in center of the camera and that positive y in the real world should appear at the top of the image and +x in the real world should appear at the right of the image.
- Lighting vectors were assumed to be given in world coordinates e.g. [0,1,0] points up along the y axis, out from the center of the coordinate frame (center of sphere) (that is, the light points down from the top).
- Background was treated as unlit empty space and so is rendered black.



(a)



(b)



(c)

Figure 1: Various Lighting Sources Rendered on a Lambertian Sphere with an Orthographic Camera

Q1.c Loading Data

As requested, a screenshot of `loadData(...)` can be found below.

```python
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

    """

    # Load the nth image, process it, and return the processed vector:
    def load_image_row(n):
        # Load rgb image:
        rgb = imread(path + 'input_{}.tif'.format(n))
        # Convert to xyz:
        xyz = rgb2xyz(rgb)
        # Extract the luminance channel (Y), ravel, and build row for I:
        return xyz[:,:,1].ravel().reshape(1,-1)

    # Load first image to build first row of I:
    I = load_image_row(1)

    # Repeat for rest of images:
    for i in np.arange(2,8):
        I = np.append(I, load_image_row(i), axis=0)

    # Load lighting data:
    L = np.load(path + 'sources.npy').T # Transpose to force into correct shape (comes as 7x3)

    # Grab original image shape:
    s = imread(path + 'input_1.tif').shape[0:2]

    return I, L, s
```

Figure 2

Q1.d Initials

Each pseudonormal vector has three degrees of freedom (2 from the normal, one from the scalar albedo). So, so long as $\mathbf{B}$ contains $P \geq 3$ pseudonormals and those pseudonormals don't all lie on a common plane, $\mathbf{B}$ should have rank 3. Likewise, the same is true for the lighting vectors in $\mathbf{L}$ so long as there are $N_L \geq 3$ lighting directions and they also don't all lie on some common plane. Thus, assuming it's truly the case that $\mathbf{I} = \mathbf{L}^T\mathbf{B}$ (that is, there is no noise and the surface is perfectly Lambertian), one should expect $\mathbf{I}$ to also have rank 3 so long as $P \geq 3$. The caveat to this is that no real surface is perfectly Lambertian and that, in real data, there is always noise; so, to realize an $\mathbf{I}$ with rank 3, one should want $N_L \gg 3$.

Running SVD on $\mathbf{I}$ (code included in the `__main__` section of `q1.py`), produces a $\mathbf{\Sigma}$ matrix with the following diagonal $\mathbf{s}$:

```
s = [79.36348099 13.16260675  9.22148403  2.414729    1.61659626  1.26289066
0.89368302]
```

Since the first three singular values are non-zero, $\mathbf{I}$ has rank ≥ 3; however, since all the singular values are non-zero, $\mathbf{I}$ has rank 7. The reason for this discrepancy is that this system is highly overdetermined (containing $N_L = 7$ images each with $P = 159039$ pixels) and, since it contains real world data, almost certainly contains noise and a target which is not perfectly Lambertian. That is, it is not truly the case that $\mathbf{I} = \mathbf{L}^T\mathbf{B}$ where each pseudonormal is corresponded with each image intensity perfectly as determined by the each lighting condition. Rather, this noise adds additional degrees of freedom which makes $\mathbf{I}$ have full row rank.

Q1.e Estimating Pseudonormals

$$\text{let: } \mathbf{I} = \begin{bmatrix} \mathbf{I_1} \\ \mathbf{I_2} \\ \vdots \\ \mathbf{I_7} \end{bmatrix}$$

$$\text{let: } \mathbf{L}^T = \begin{bmatrix} \mathbf{L_1} \\ \mathbf{L_2} \\ \vdots \\ \mathbf{L_7} \end{bmatrix}$$

$$\text{let: } \mathbf{B} = \begin{bmatrix} \mathbf{B_1} | \mathbf{B_2} | \dots \mathbf{B_P} \end{bmatrix}$$

where $\mathbf{I_i}$ and $\mathbf{L_i}$ represent the row vectors of image pixels and lighting directions respectively for the i^{th} image, and $\mathbf{B_j}$ represents the column vectors of the pseudonormals for each pixel.

With these definitions, the system $\mathbf{I} = \mathbf{L}^T\mathbf{B}$ could also be written sparsely in the form $\mathbf{Ax} = \mathbf{y}$ where $\mathbf{x}$ is a $3P \times 1$ vector derived from stacking $\mathbf{B}$, $\mathbf{y}$ is a $7P \times 1$ vector derived from stacking $\mathbf{I}$, and $\mathbf{A}$ is a sparse $7P \times 3P$ derived from $\mathbf{L}^T$ as follows:

$$\mathbf{x} = \begin{bmatrix} \mathbf{B_1} \\ \mathbf{B_2} \\ \vdots \\ \mathbf{B_P} \end{bmatrix}$$

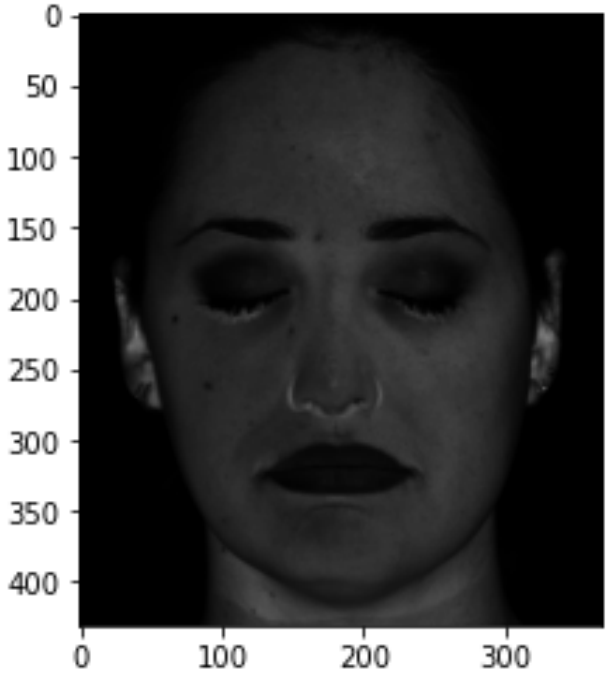
$$\mathbf{y} = \begin{bmatrix} \mathbf{I_1}^T \\ \mathbf{I_2}^T \\ \vdots \\ \mathbf{I_7}^T \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} \mathbf{L_1} & \mathbf{0}_{1\times3} & \dots & \mathbf{0}_{1\times3} \\ \mathbf{0}_{1\times3} & \mathbf{L_1} & \dots & \mathbf{0}_{1\times3} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & \dots & \mathbf{L_1} \\ \mathbf{L_2} & \mathbf{0}_{1\times3} & \dots & \mathbf{0}_{1\times3} \\ \mathbf{0}_{1\times3} & \mathbf{L_2} & \dots & \mathbf{0}_{1\times3} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & \dots & \mathbf{L_2} \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{L_7} & \mathbf{0}_{1\times3} & \dots & \mathbf{0}_{1\times3} \\ \mathbf{0}_{1\times3} & \mathbf{L_7} & \dots & \mathbf{0}_{1\times3} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & \dots & \mathbf{L_7} \end{bmatrix}$$

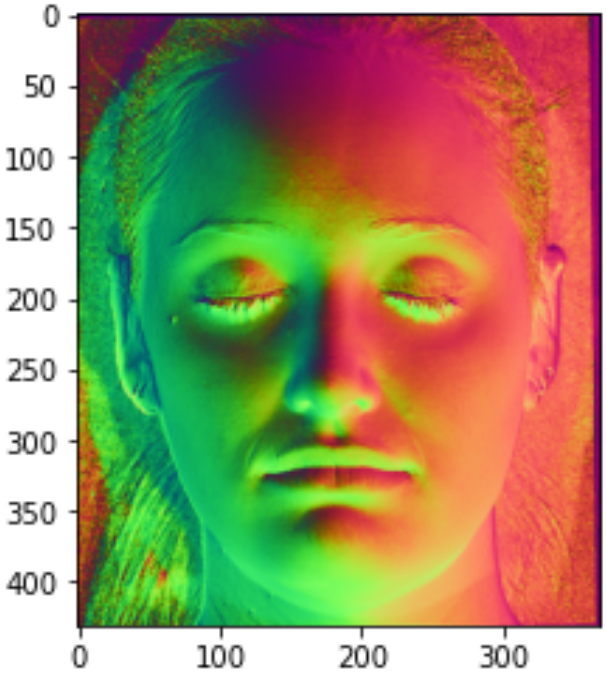
Q1.f Albedos and normals

Note: the normals image was normalized to a range of 0 to 1 for display purposes in order to prevent values below 0 from being clipped. This was only done for the display for this problem and was not done inside `estimateAlbedosNormals`, only inside `displayAlbedosNormals`. Notably, the value of `normalIm` which is returned from `displayAlbedosNormals` will **not** have this normalization since it was not explicitly requested and I worry it may throw off the autograder.

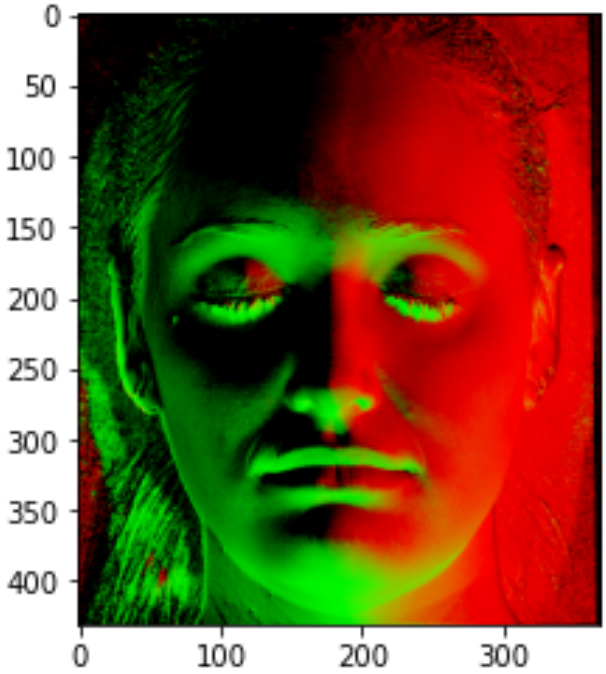
Also, in order to make grading easier, the non-normalized normals image has also been included in Figure 3(c) below in case this is the result expected. However, in the analysis below, when the "normals image" is referred to, it's Figure 3(b) which is being referenced since it doesn't suffer from clipping.



(a) Albedos Image



(b) Normals Image



(c) Non-normalized Normals Image

Figure 3

The albedos match expectation. Salient in the image are areas of high albedo where the nose meets the face and in the crevices of the ears; however, these should be expected since those areas on a face are more oily and generally reflective.

Likewise, the displayed normals match what one would expect, following the curvature of the face consistently.

Q1.g Normals and depth

The gradient vector of a function is defined as the direction in which the value increases most. That is, for $z = f(x, y)$, $\nabla f = \begin{bmatrix} f_x & f_y \end{bmatrix}^T$ is the direction of greatest ascent between contours (slices of the surface) at any given z value. Alternatively, this can be seen as the direction of the shortest path between two level slices. Thus, $\nabla f = \begin{bmatrix} f_x & f_y \end{bmatrix}^T$ will always point internal to the surface and perpendicular to the curve, in the plane of the level contour. An example of this is depicted below for two level contours of an arbitrary surface at arbitrary levels $z = z_1^*$ and $z = z_2^*$.

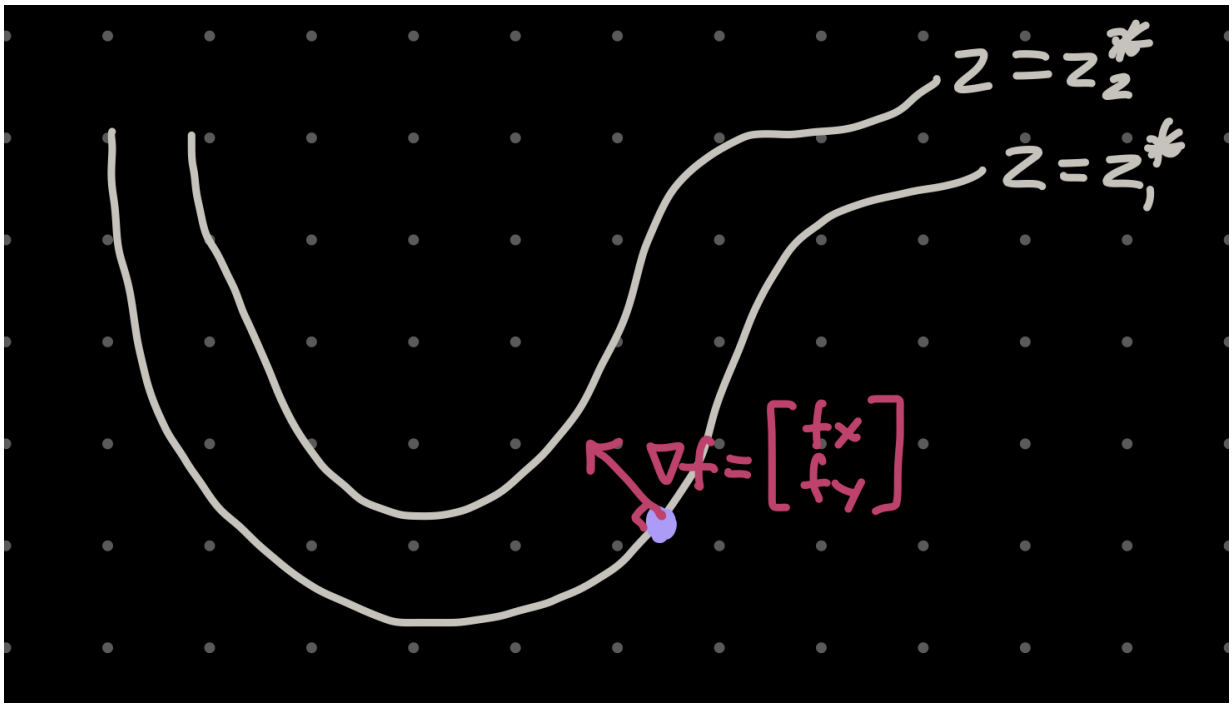


Figure 4

Extending to the full three dimensions of the surface: since the gradient is defined as the direction of greatest ascent between contours at different z-levels, the vector $\begin{bmatrix} f_x & f_y & 1 \end{bmatrix}^T$ will clearly be perpendicular to surface and pointing internally. As a result, the vector $\begin{bmatrix} f_x & f_y & -1 \end{bmatrix}^T$ will be normal to the surface, pointing perpendicularly and away. Normalizing this gives the unit normal to be:

$$\mathbf{n} = \begin{bmatrix} \frac{f_x}{\sqrt{f_x^2+f_y^2+1}} \\ \frac{f_y}{\sqrt{f_x^2+f_y^2+1}} \\ \frac{-1}{\sqrt{f_x^2+f_y^2+1}} \end{bmatrix}$$

As a result, it can be clearly seen that, $f_x = \frac{-n_1}{n_3}$ and $f_y = \frac{-n_2}{n_3}$. due to the definition of the gradient. As shown, this result is a natural extension of the geometric definition of a gradient, which is the vector of the partial derivatives.

Q1.h Understanding the integrability of gradients

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

$$\therefore g_x = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad g_y = \begin{bmatrix} 4 & 4 & 4 & 4 \\ 4 & 4 & 4 & 4 \\ 4 & 4 & 4 & 4 \end{bmatrix}$$

Procedure 1:

$$\text{let: } g(0,0) = 1$$

$$g(0,:) = \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}$$

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

Procedure 2:

$$\text{let: } g(0,0) = 1$$

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

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

Notably these two procedures produce the same result.

In the example provided, if any row or column of g_x or g_y excluding the first of each were modified, the two procedures would produce different results. For example:

$$\text{let: } g(0,0) = 1, \quad g_x = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad g_y = \begin{bmatrix} 4 & 4 & 4 & 4 \\ 4 & \mathbf{5} & 4 & 4 \\ 4 & 4 & 4 & 4 \end{bmatrix}$$

In which case, procedure 2 would still produce

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

but procedure 1 would now produce:

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

The gradients estimated in (g) could be non-integrable because nothing constrains them to have compatible matrices (that is, they're calculated independently) and, as shown above, even small changes to the gradient matrices can cause non-integrability. Specifically, the second derivatives (derivatives of g_x and g_y) must be equal for g_x and g_y to be integrable and there is nothing strictly guaranteeing that.

Q1.i Shape estimation

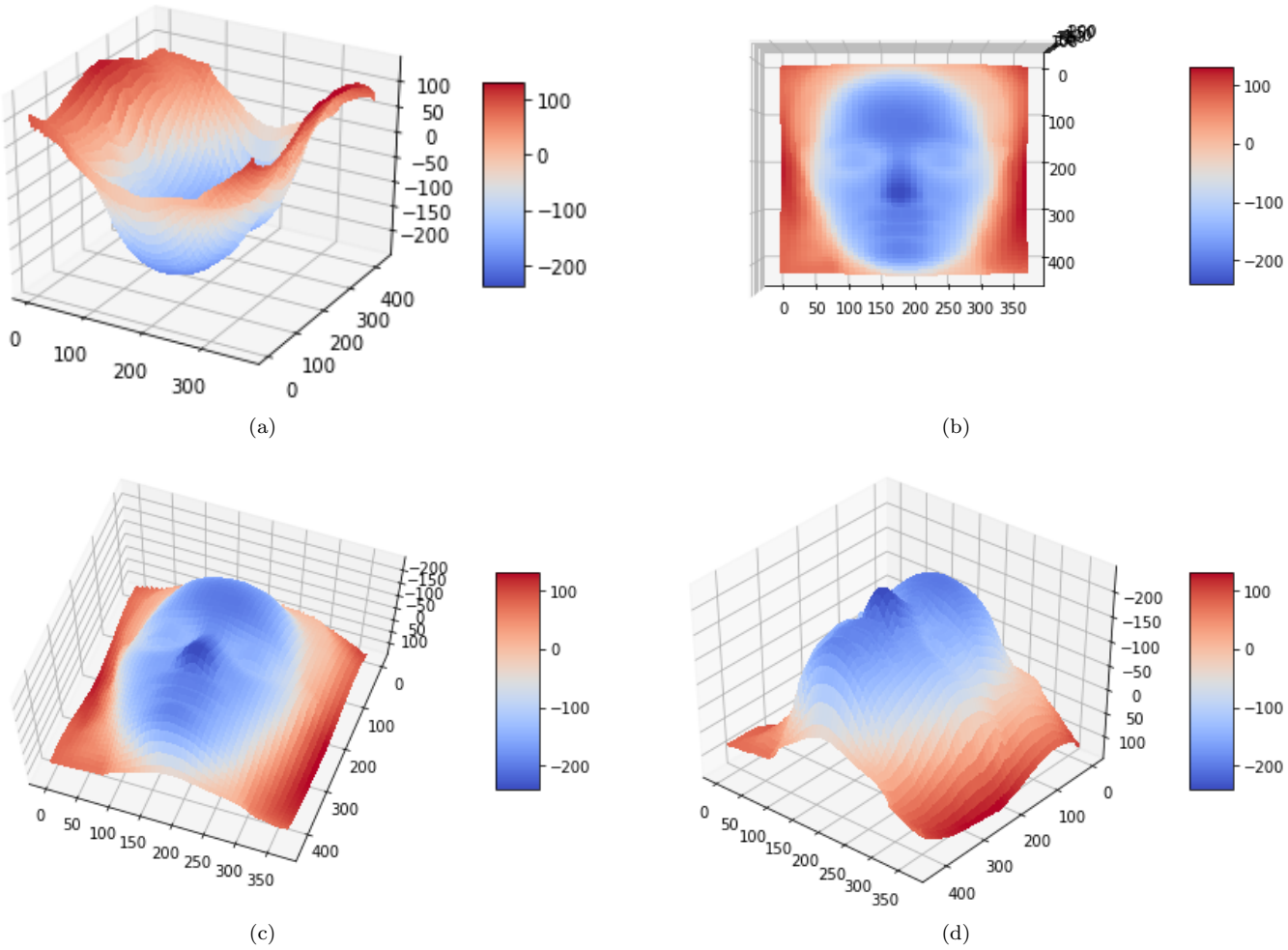


Figure 5: Views of the Face Mesh.

Note: the face points in the negative z-direction. This was deemed okay in `@612` on Piazza.

2 Uncalibrated photometric stereo

Q2.a Uncalibrated Normal Estimation

Essentially by definition, the rank of a matrix is the number of non-zero singular values that it has. So, by decomposing a matrix $\mathbf{M}$ into $\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ and setting all of but the top k singular values in $\hat{\mathbf{\Sigma}}$ to zero, the rank of $\hat{\mathbf{M}} = \mathbf{U}\hat{\mathbf{\Sigma}}\mathbf{V}^T$ will inherently be k. Moreover, since the singular values are ordered in descending order, keeping the top k will preserve as much information content (related to the top eigenvalues) as possible will still ensuring the constraint on k is met.

For the system $\mathbf{I} = \mathbf{L}^T\mathbf{B}$, since the dimensions of $\mathbf{I}$ are $7 \times P$ and the desired rank $k = 3$ is $k \leq \min(7, P)$ so long as $P \geq 3$, the system can be factorized as follows:

$$
\begin{aligned}
&\text{let: } \mathbf{I} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \\
&\text{let: } \begin{cases} \hat{\mathbf{\Sigma}}_{ii} = \mathbf{\Sigma}_{ii} \text{ for } i \in [1, k] \\ \hat{\mathbf{\Sigma}}_{ij} = 0 \text{ otherwise} \end{cases} \\
&\rightarrow \hat{\mathbf{I}} = \mathbf{U}\hat{\mathbf{\Sigma}}\mathbf{V}^T \\
&\text{let: } \hat{\mathbf{I}} = \hat{\mathbf{L}}^T\hat{\mathbf{B}} \\
&\therefore \hat{\mathbf{L}}^T = \mathbf{U}\hat{\mathbf{\Sigma}}^{1/2}, \;\; \hat{\mathbf{B}} = \hat{\mathbf{\Sigma}}^{1/2}\mathbf{V}^T \\
&\rightarrow \hat{\mathbf{L}} = (\mathbf{U}\hat{\mathbf{\Sigma}}^{1/2})^T, \;\; \hat{\mathbf{B}} = \hat{\mathbf{\Sigma}}^{1/2}\mathbf{V}^T
\end{aligned}
$$

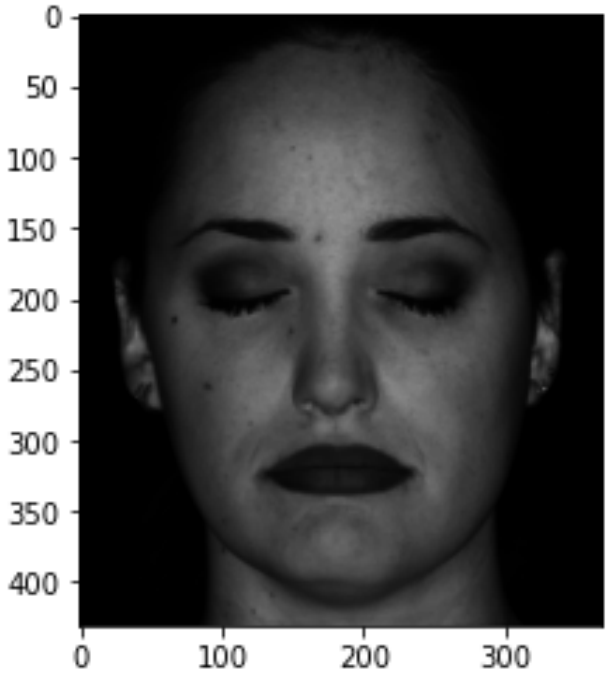
This works since, if $\mathbf{I} = \mathbf{L}^T\mathbf{B}$, it will naturally still be the case that $\hat{\mathbf{I}} = \hat{\mathbf{L}}^T\hat{\mathbf{B}}$ and since $\hat{\mathbf{I}}$ will necessarily have rank 3 if $P \geq 3$, both constraints have been satisfied.

Note: because $\hat{\mathbf{\Sigma}}$ is diagonal, $\hat{\mathbf{\Sigma}}^{1/2}$ is simply $\hat{\mathbf{\Sigma}}$ with every diagonal element raised to the power $1/2$.

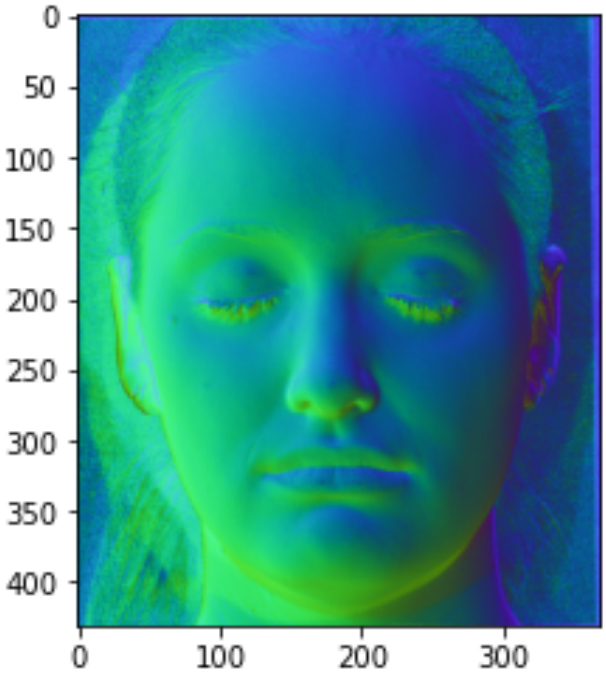
Q2.b Calculation and Visualization

Note: the normals image was normalized to a range of 0 to 1 for display purposes in order to prevent values below 0 from being clipped. This was only done for the display for this problem and was not done inside `estimateAlbedosNormals`, only inside `displayAlbedosNormals`.

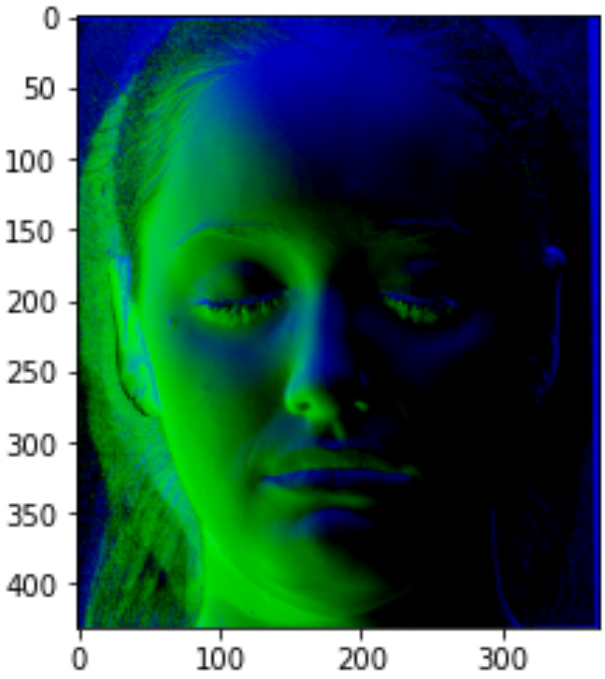
Also, in order to make grading easier, the non-normalized normals image has also been included in Figure 3(c) below in case this is the result expected.



(a) Albedos Image



(b) Normals Image



(c) Non-normalized Normals Image

Figure 6

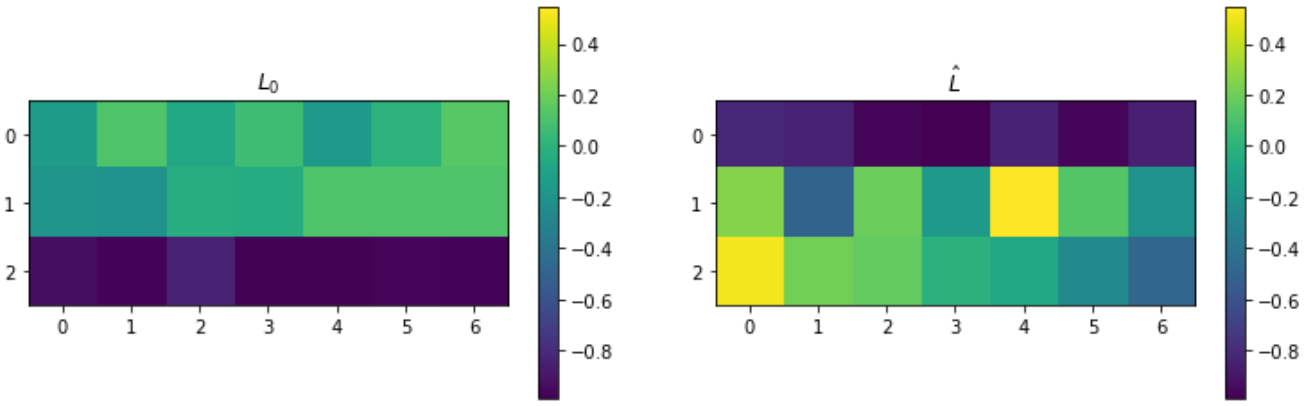
Q2.c Comparing to ground truth lighting

The recovered lighting matrix $\hat{\mathbf{L}}$ and the ground truth lighting matrix $\mathbf{L}_0$ are as shown below. Note: since $\mathbf{L}_0$ is given as a matrix of unit vectors, the vectors in $\hat{\mathbf{L}}$ were normalized so that it had the same formatting as $\mathbf{L}_0$ and could more easily be compared.

```
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]]

L = [[-0.818  -0.8371 -0.9646 -0.9863 -0.8372 -0.9571 -0.8539]
 [ 0.2591 -0.5012  0.1999 -0.1649  0.5421  0.1317 -0.2019]
 [ 0.5137  0.2195  0.172  -0.0046 -0.0724 -0.258  -0.4796]]
```

To ease evaluation, both of these matrices were also visualized using `matplotlib`'s `imshow`:



(a) Ground Truth

(b) Reconstructed

Figure 7

As expected, these two are very different. A simple modification that can be made to the factorization which changes $\hat{\mathbf{L}}$ and $\hat{\mathbf{B}}$ but keeps the rendered images $\mathbf{I}$ the same would be to change the way the $\mathbf{U}$, $\hat{\mathbf{\Sigma}}$, and $\mathbf{V}$ matrices are partitioned. For example, instead of:

$$\hat{\mathbf{L}} = (\mathbf{U}\hat{\mathbf{\Sigma}}^{1/2})^T, \quad \hat{\mathbf{B}} = \hat{\mathbf{\Sigma}}^{1/2}\mathbf{V}^T$$

one could use:

$$\hat{\mathbf{L}} = (\mathbf{U}\hat{\mathbf{\Sigma}})^T, \quad \hat{\mathbf{B}} = \mathbf{V}^T$$

Q2.d Reconstructing the shape, attempt 1

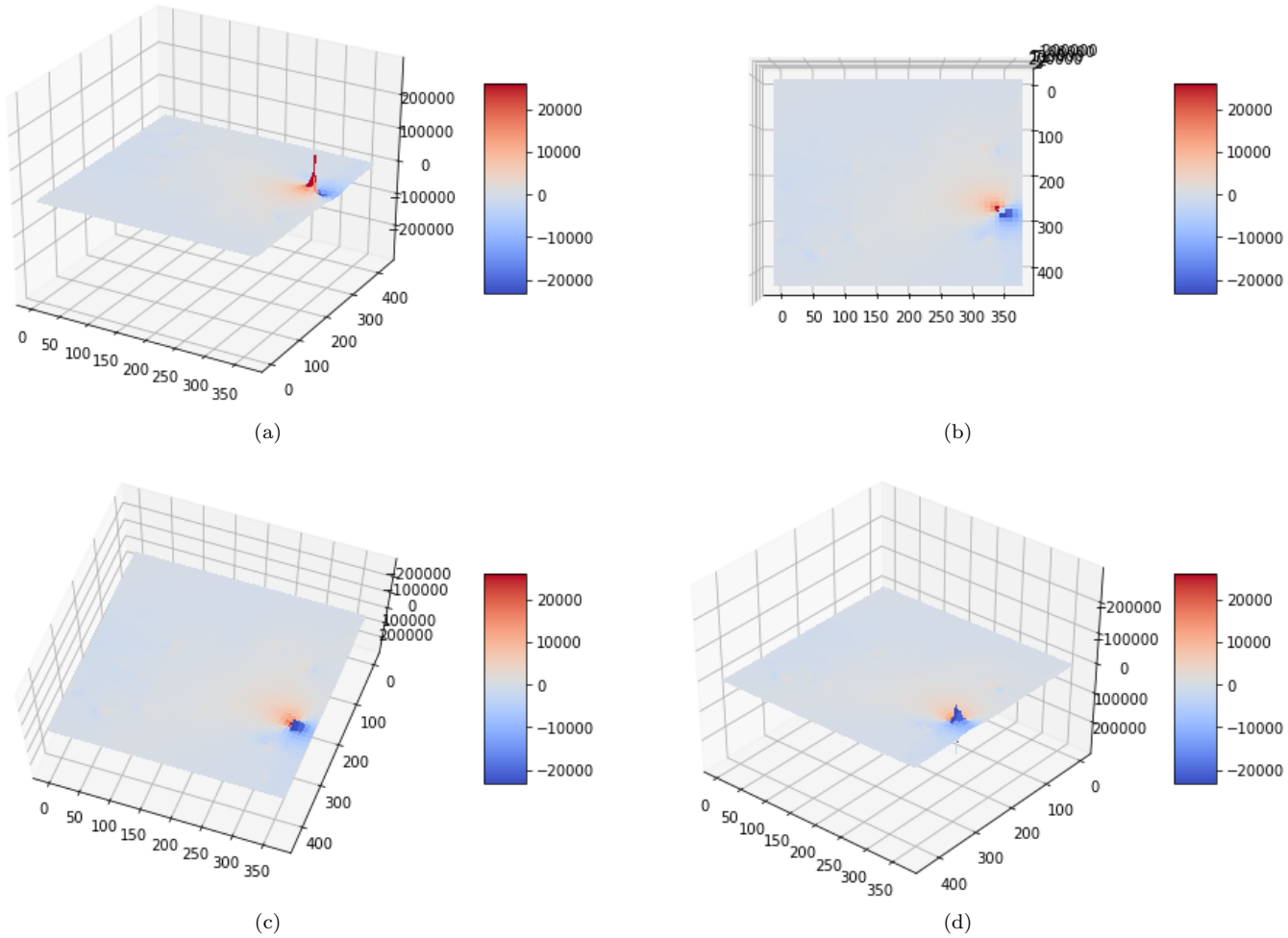


Figure 8: Views of the Reconstructed "Face" Mesh (attempt 1).

This does not look like a face. This is to be expected since it's unlikely the pseudonormals were factored out perfectly (only correct up to a matrix), which further decreases the odds that the normals are integrable. Quite evident from these results, the normals are unlikely to be integrable.

Q2.e Reconstructing the shape, attempt 2

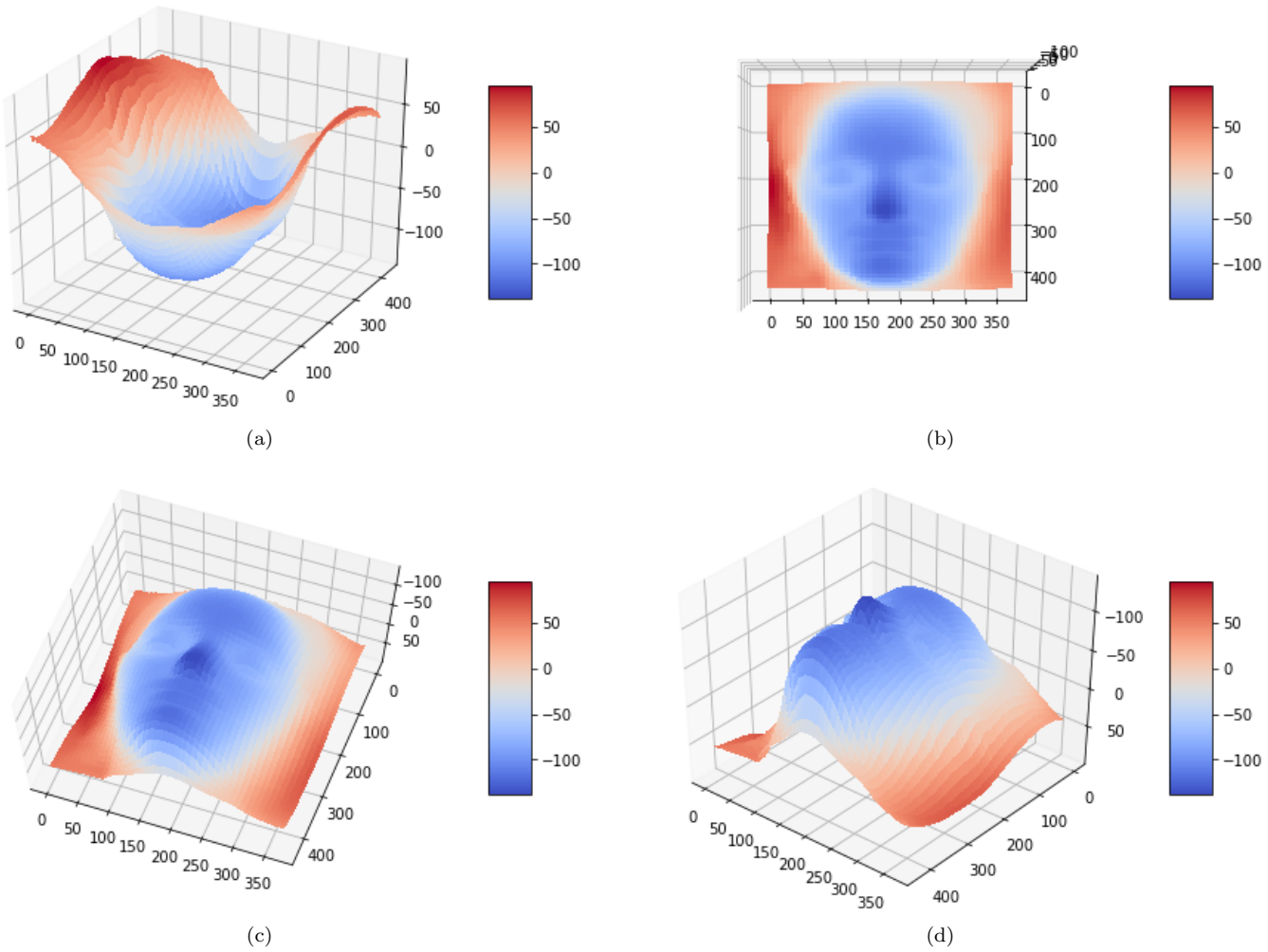
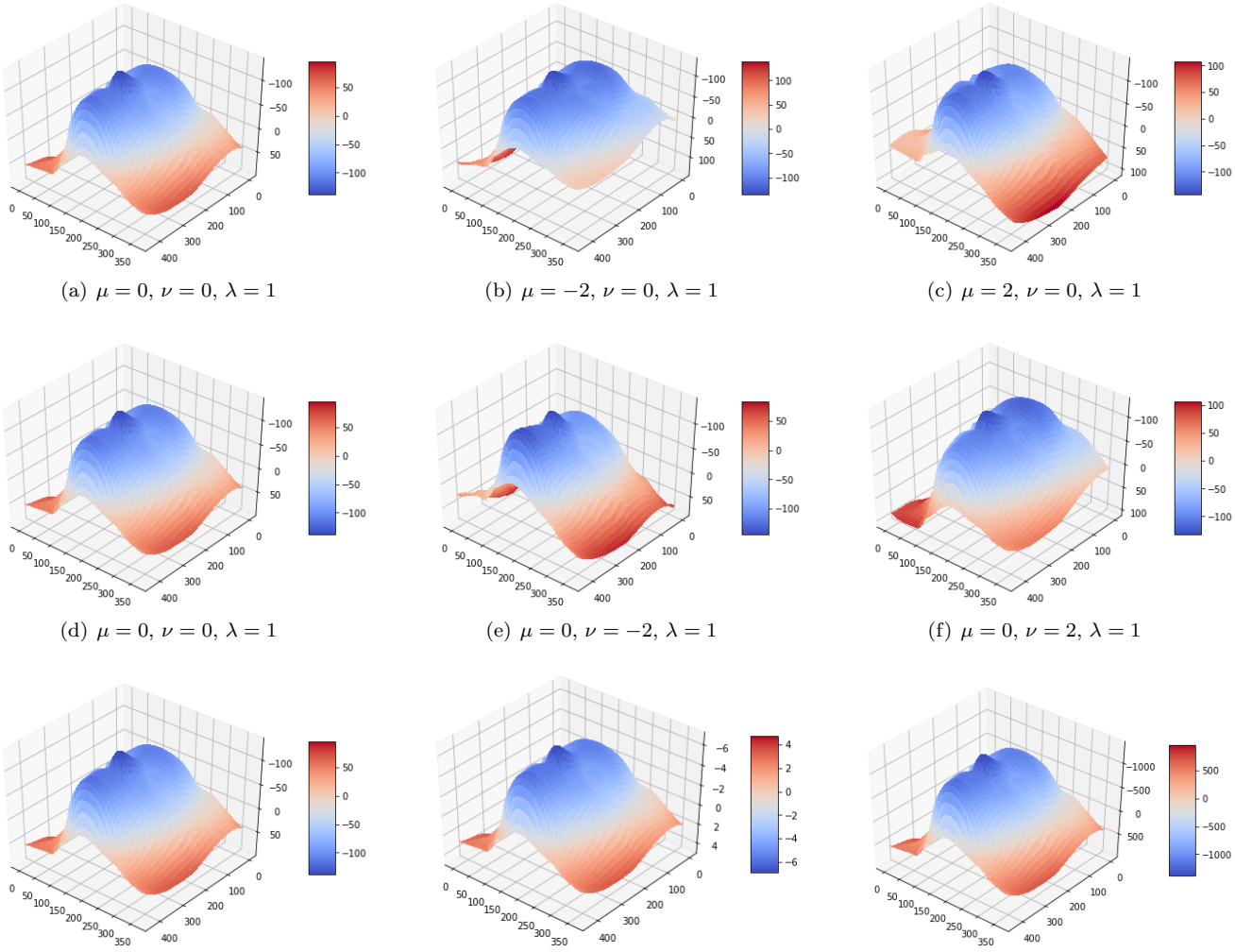


Figure 9: Views of the Reconstructed Face Mesh (attempt 2) with Integrability Enforcement.

This surface looks visually identical to the one that came out of the calibrated photometric stereo.

Q2.f Why low relief?

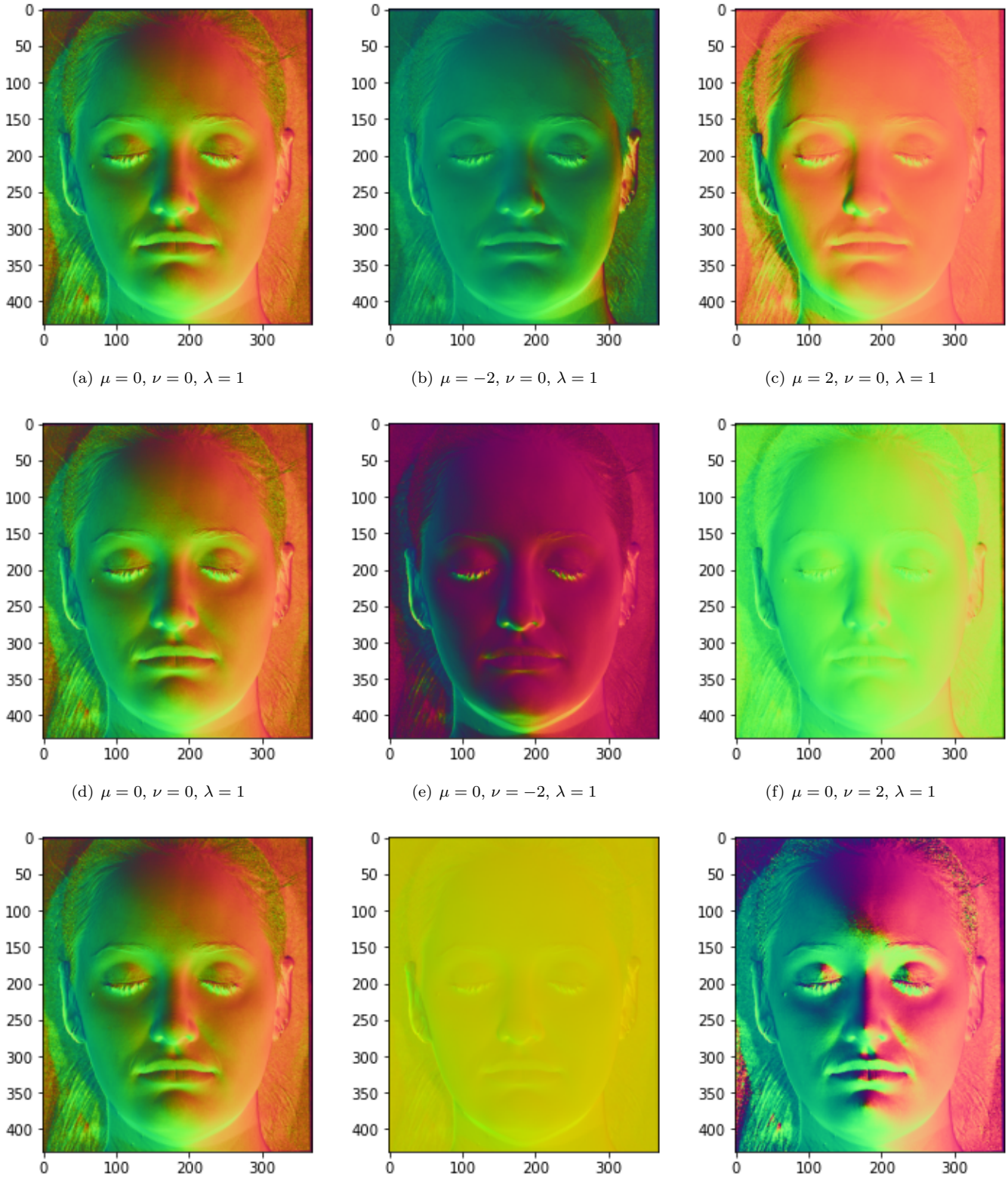


(a) $\mu = 0$, $\nu = 0$, $\lambda = 1$ (b) $\mu = -2$, $\nu = 0$, $\lambda = 1$ (c) $\mu = 2$, $\nu = 0$, $\lambda = 1$

(d) $\mu = 0$, $\nu = 0$, $\lambda = 1$ (e) $\mu = 0$, $\nu = -2$, $\lambda = 1$ (f) $\mu = 0$, $\nu = 2$, $\lambda = 1$

(g) $\mu = 0$, $\nu = 0$, $\lambda = 1$ (h) $\mu = 0$, $\nu = 0$, $\lambda = 0.05$ (i) $\mu = 0$, $\nu = 0$, $\lambda = 10$

Figure 10: Effect of Varying Parameters of Bas Relief on the Reconstructed Surface



(a) $\mu = 0$, $\nu = 0$, $\lambda = 1$ (b) $\mu = -2$, $\nu = 0$, $\lambda = 1$ (c) $\mu = 2$, $\nu = 0$, $\lambda = 1$

(d) $\mu = 0$, $\nu = 0$, $\lambda = 1$ (e) $\mu = 0$, $\nu = -2$, $\lambda = 1$ (f) $\mu = 0$, $\nu = 2$, $\lambda = 1$

(g) $\mu = 0$, $\nu = 0$, $\lambda = 1$ (h) $\mu = 0$, $\nu = 0$, $\lambda = 0.05$ (i) $\mu = 0$, $\nu = 0$, $\lambda = 10$

Figure 11: Effect of Varying Parameters of Bas Relief on the Normalized Surface Normal Images

The bas relief ambiguity can be expressed as manipulating the surface described by $z = f(x, y)$ so that it takes the form $z = \lambda f(x, y) + \mu x + \nu y$ while still being described by the same recovered pseudonormals. As such, λ can be viewed as scaling and squishing/stretching the surface along the z-axis and μ and ν can be viewed as tilting the surface about the y and x axes respectively. However, since the pseudonormals must remain valid through this transformation, when the surface is tilted, it also becomes squished against the plane $(\mu x + \nu y)$ which is being added, as can be seen in Figures 10 and 11 above. Considering all of this, it is sensible that it is called "low relief", because increasing the magnitude of μ or ν causes the

features of the surface to protrude less while decreasing λ makes the surface smaller in height and flattened, that is to say "lower" and flatter with respect to the z-axis.

Q2.g Flattest Surface Possible

The solution: As seen in the answer to Q2.f, particularly in Figure 11(h) above, decreasing λ causes the surface normals to become less pronounced in order for the reconstructed pseudonormals to remain valid. As such, bringing λ as close to 0 as possible will result in the flattest surface possible.

Something that might seem like a solution but isn't: Likewise, it should be noted that although tilting the plane $\mu x + \nu y$ to an extreme angle also causes the surface to get effectively squished by the plane (as can be seen in Figure 11(f) above), because the surface is also getting tilted (not just squished) and the pseudonormals still need to remain correct, the surface will inherently have to bend or deflect. So, although its features will become less pronounced due to massively increasing μ or ν, a surface will not become maximally flat as a result.

Q2.h More measurements

Although adding more measurements will improve the noise-resistance of the least squares result, this will not help resolve the bas-relief ambiguity since it is baked into the geometery of the problem. That is, without also adding additional constraints or assumptions, just adding more data will not resolve the ambiguity, even if it's from new lighting angles.