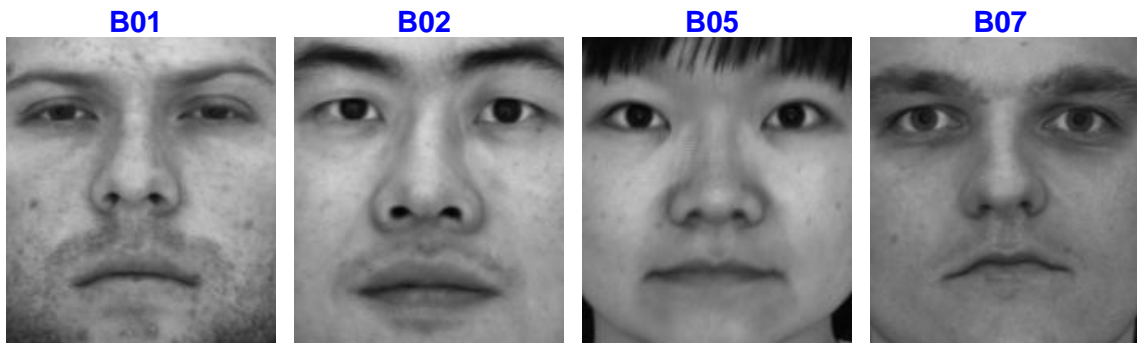


Name : Yen-Ting Liu (ytliu2)

Part-1 : Estimate the albedo and surface normals

- 1) Insert the albedo image of your test image here:



- 2) What implementation choices did you make? How did it affect the quality and speed of your solution?

Language

I try to follow the instruction using vectorized calculations instead of loops. Reconstruct surface by `average` method only takes roughly 0.6 s (from loading to final surface plot). Computation-wise these are mathematically equivalent, so no known impact in accuracy.

Shading model

I closely follow the model from lecture 4, assuming the result follows Lambert's law

$$I = \rho(\vec{S} \cdot \vec{N}) = \rho \vec{N} \cdot \vec{S}$$

where ρ is the albedo map, \vec{N} is the surface normal, and \vec{S} is the lighting direction. Each dataset has more than 3 images, allowing least square estimations for surface normals.

Surface retrieval

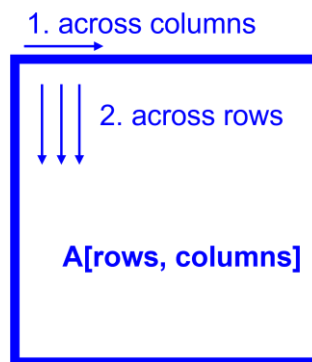
In the `get_surface` function, all methods (column-first, row-first, average, and random walk) start from the top-left corner, whose pixel index is (0, 0). However, since the farther the row/column from (0, 0) for column/row method, the more distortions there are (see next question for an example and its explanation).

Since average method simply averages height maps from column/row method, it will have smoothed out artefacts as well as improvements from both methods (next question has an example).

- 3) What are some artifacts and/or limitations of your implementation, and what are possible reasons for them?

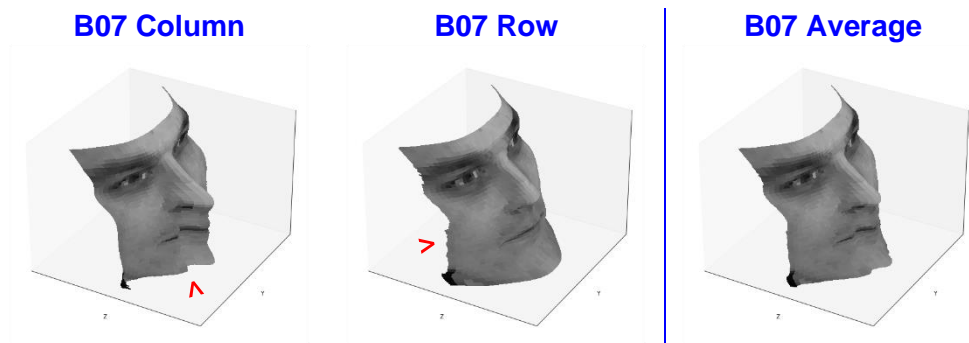
In all integration methods, *none of them* factor in the integrability constraint at all. This is easily seen in the extreme discrepancy between column and row method. On the other hand, we can observe that column and row method have smooth row and column.

According to the latest clarification post on Piazza, the column method integrates across columns (horizontal direction, along a row), then integrates across rows (vertical direction, along columns). In my implementation, taking column method as an example, I integrate along row 0, therefore, row 0 will be smooth, and all columns are each smooth (vertically).















This implies that vertically, each vertical line is smooth in its own, and has no relationship with neighbors, besides from the beginning. So for column method, the forehead will be smooth, and left-right side views are smooth, but far away from the first row will be rough, aka, having vertical folds. This is evident in **B07 Column** example (red angle) below.

For row method, similar behavior as column method, but we would expect smooth forehead and chin, with smooth right side (evident in next example with all viewpoints), but rough left side (see **B07 Row**, red angle).



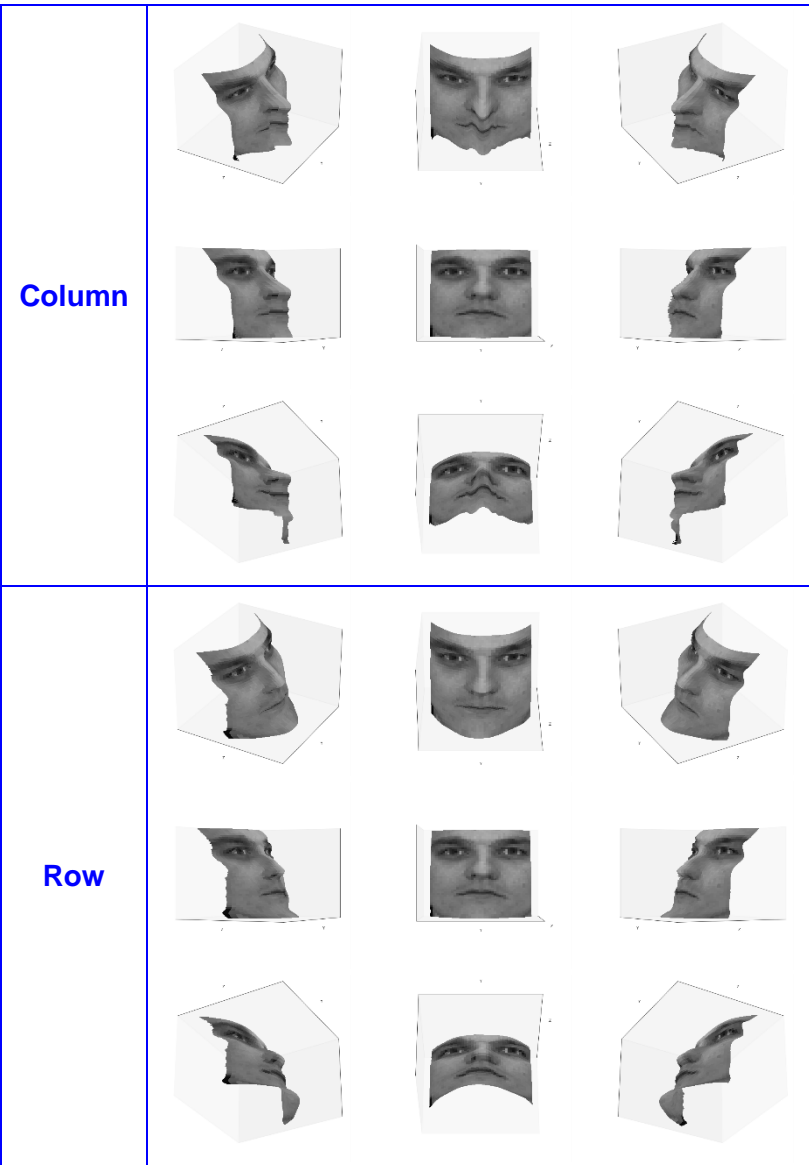
4) Display the surface normal estimation images below:

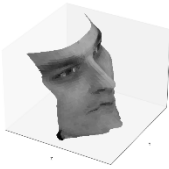
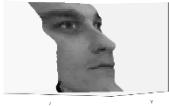
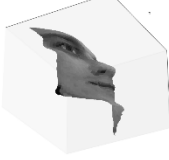



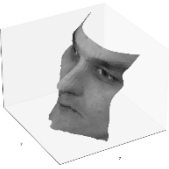

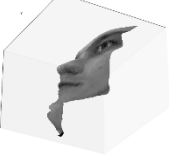
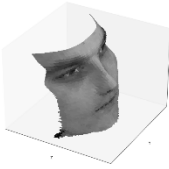




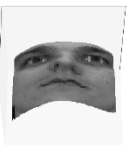
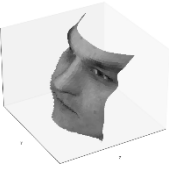

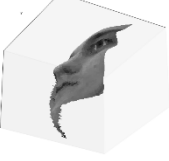
	N_x	N_y	N_z
B01			
B02			
B05			
B07			

Part-2 : Compute Height Map

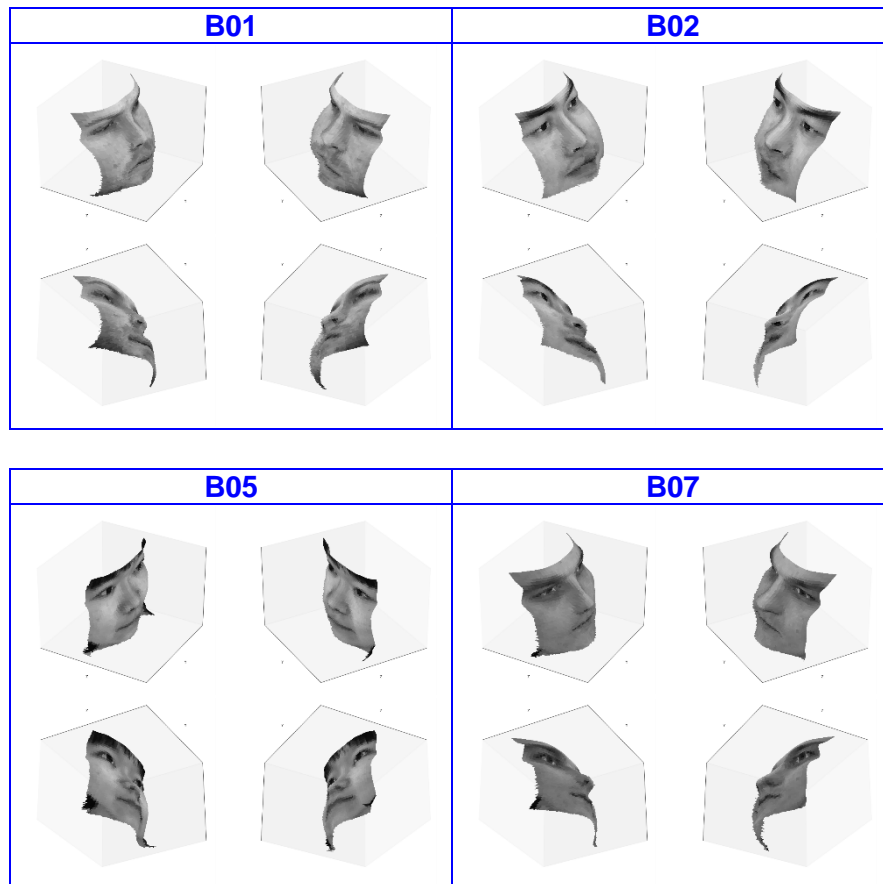
- 5) For every subject, display the surface height map by integration. Select one subject, list height map images computed using different integration method and from different views; for other subjects, only from different views, using the method that you think performs best. When inserting results images into your report, you should resize/compress them appropriately to keep the file size manageable -- but make sure that the correctness and quality of your output can be clearly and easily judged.
- I choose to use B07 to demo different integration methods. Viewpoints are organized by following azimuths and elevations:

az -60, el 30		az 60, el 30
	front	
az -60, el -30		az 60, el -30



Average	  	  	  
Random (n=5)	  	  	  

I choose to use random method (with $n=5$) for all subjects. Only diagonal viewpoints from are shown here since direct side views cannot illustrate the surface topography.



6) Which integration method produces the best result and why?

Based on my observation, I would say random works best in terms of visual result. It successfully removes awkward lip distortion and abnormal chin size.

My implementation of random creates a random combination of horizontal-vertical movement from (0, 0) to target pixel, and average out accumulated results (in this report, using $n=5$). While the image may violate integrability constraint, multiple random walk paths can smooth out the impact of discontinuity along any given path.

- 7) Compare the average execution time (only on your selected subject, “average” here means you should repeat the execution for several times to reduce random error) with each integration method, and analyze the cause of what you’ve observed:
These are averaged over 10 repeats using subject B07 to get measurable delay, timed by Python `time.time()` function, which provides processor time in floating point.

Integration method	Execution time
Column	0.30 ms
Row	0.32 ms
Average	0.79 ms
Random	12.4 s

For the slight difference between column/row, I would assume it is because of row/column major, causing some overhead during array copy, in my implementation, column method has

```
height_map[1:, :] = f_y[1:, :]
```

while row method has

```
height_map[:, 1:] = f_x[:, 1:]
```

which probably generates some noticeable delay after lots of copy (with 10 repeats, these operations are repeated for almost 2000 times).

Average is simply combining column and row method, which is roughly double their time with some overhead for averaging over their results.

Random method uses in-place shuffle to generate random walk sequence and iterate over *all* pixels using a nested for-loop. This is bound to be much slower than other vectorized operation. Therefore, the computation cost for random method is the highest.

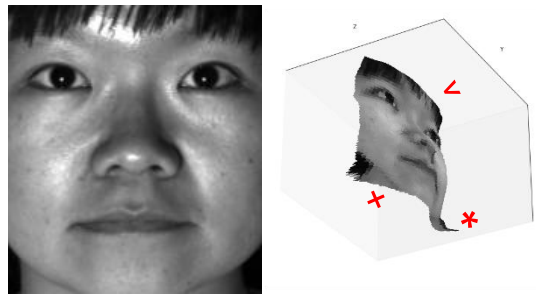
Part-3 : Violation of the assumptions

- 8) Discuss how the Yale Face data violate the assumptions of the shape-from-shading method covered in the slides.

(Using assumptions listed in lecture slide 4.)

Lambert's law

In this assignment, we assume that Lambert's law holds true for all subjects. However, this is not the case for specular surface like eyeball and oily skin areas.



These regions would have abnormal high reflectance along all directions, presenting itself as a blob of flat region. Subject B05 (red arrow) shows flat facial feature around its forehead, nose ridge, and cheekbones.

Local shading model

This assumes each point on the surface receives light only from sources visible at that point.

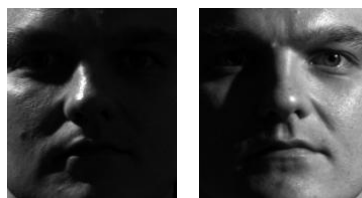
Using the same subject, knowing the fact that there should only be a single light source, area below jawline still shows portion of the neck, indicates these regions are illuminated by other means, probably through environment. The algorithm cannot accommodate this type of surface (red plus), which fail to interpret the depth.

Shadow

Asides from neck, the height map has outliers around shadowed/non-illuminated area (red asterisk), since there is no/not enough intensity information to perform the least square fitting.

- 9) Choose one subject and attempt to select a subset of all viewpoints that better match the assumptions of the method. Show your results for that subset.

Using subject B07, I remove series of images with significant shadowing like these



And using the residual from the least square fit, we can figure out the Pearson's R of each pixel by

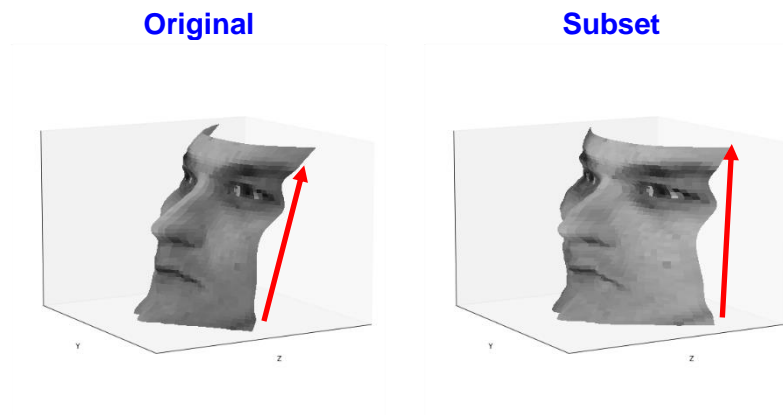
$$r^2 = 1 - \frac{\eta}{N\sigma^2}$$

where η is the residual of the LS fitting, N is number of samples, σ^2 is the variance of all sample points. By taking the median of all r^2 , we can have a rough idea how consistence current dataset is.

After removing the following image series (ordered numerically by illumination angle),

3, 30, 31, 32, 33, 34, 50, 51, 52, 53,
54, 55, 56, 57, 58, 59, 60, 61, 62, 63

median of the r^2 rises from 0.4224 to 0.7589. For the sake of simplicity, I choose to use average method for height map reconstruction,



We can easily find out that the surface is no longer skewed.

- 10) Discuss whether you were able to get any improvement over a reconstruction computed from all the viewpoints.

Removed images are mainly those with shadows that cover up other facial features. Those shadow would cause lower facial features, e.g. lips, to become darker than usual, equivalent to uniform, flatter region, so they can consistently get blocked. Subset result shows more natural (lower) facial feature that is consistent with what we would expect to appear on a human.

Part-4 : Bonus

Post any extra credit details/images/references used here.

Due to limited assignment time, I decided to emphasize my bonus work on satisfying the integrability constraint. The method I choose to improve this is the work from Frankot and Chellappa, following the MATLAB implementation from <https://github.com/karan9nov/shape-from-shading/blob/master/frankotchellappa.m>

Frankot-Chellappa uses discrete Fourier transform to represent our surface of interest as a finite set of basis functions, which implicitly enforced the integrability constraint, since the surface is now projected onto a subspace that span across a set of integrable slopes.

From here, we can see that discontinuities of subject B05 using random walk are now resolved. However, shadow and local shading issue originated from the dataset is still not resolved, therefore, flat forehead is still present.

