CS543/ECE549 Assignment 3

Name: Li-Kai Chuang
NetId: likaikc2

Part 1: Homography estimation

A: Describe your solution, including any interesting parameters or implementation choices for feature extraction, putative matching, RANSAC, etc.

For feature extraction, I'm using SIFT detector and descriptor. After interest points are generated in both left and right image, we start matching them. We're using Euclidean distance to compute the error between two SIFT descriptors.

Something worth mentioning when matching them is that I use the cross comparing. It simply means that only two points are their best match will they be matched together. More specifically, for each interest point in the left image, it gives score to each point in the right image. The point that gets the highest score is the left point's best match. Only if it is the best match of its best match will they be considered as a real match.

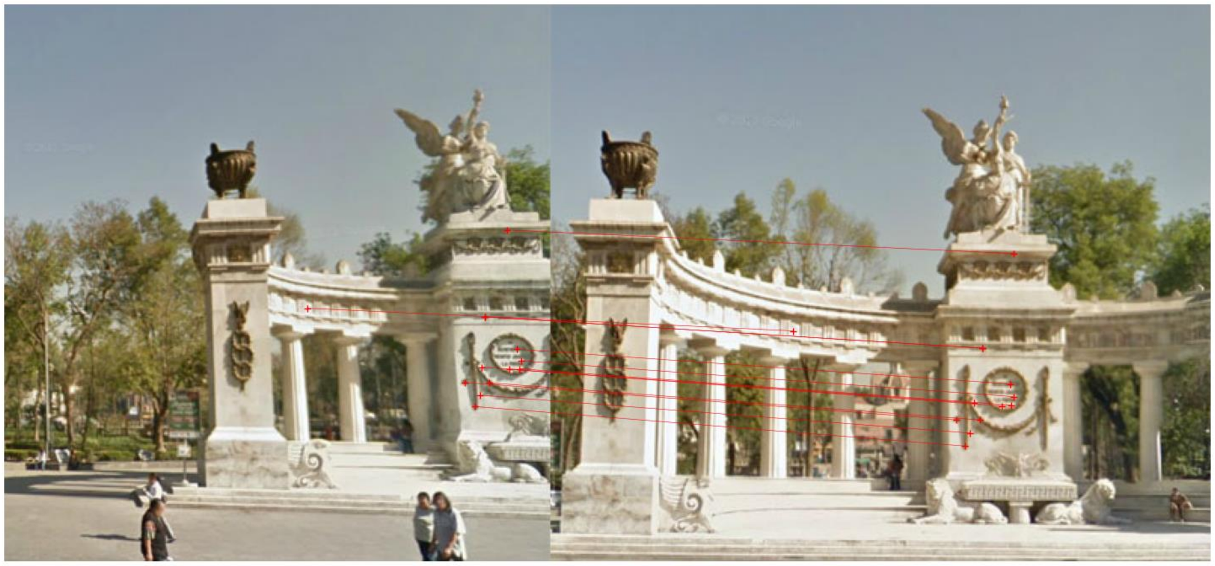
To solve the homography, we first construct the A matrix. Then we can solve it with pseudo-inverse or SVD.

As for RANSAC, we randomly pick 6 within a total of 56 points to compute the homography. Then use the homography to project all the 56 points into the right image. Compare how close the projected point and actual point are on the right image. If they are close enough then we consider the point as an inlier of a certain homography. Our goal is to find as many inliers as possible. So we do the above process 1000 times and pick the best homography.

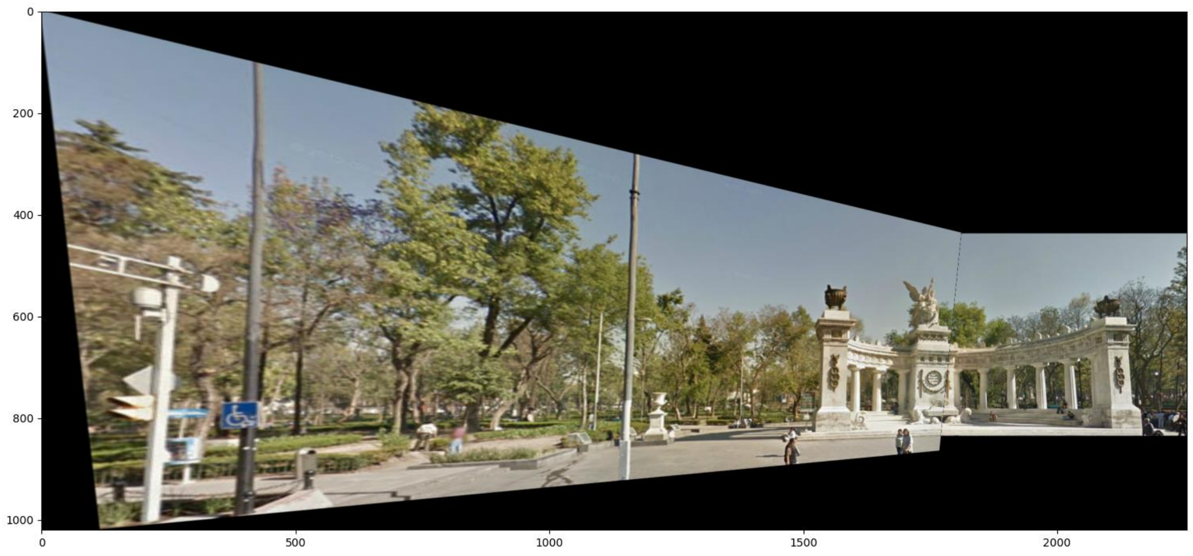
RANSAC config:
Iteration: 1000
Number of points to fit (computer the homography): 6
Error computing method: Euclean Distance
Error threshold: 0.5 (Feature points that are smaller than this are considered inliers)

B: For the image pair provided, report the number of homography inliers and the average residual for the inliers. Also, display the locations of inlier matches in both images.
RANSAC config:
Iteration: 1000
Number of points to fit (computer the homography): 6
Error computing method: Euclean Distance

Error threshold: 0.5 (Feature points that are smaller than this are considered inliers)
========
Result below:
Total feature points: 56
Number of inliers: 13
Average residual error of inliers: 0.223745263234332



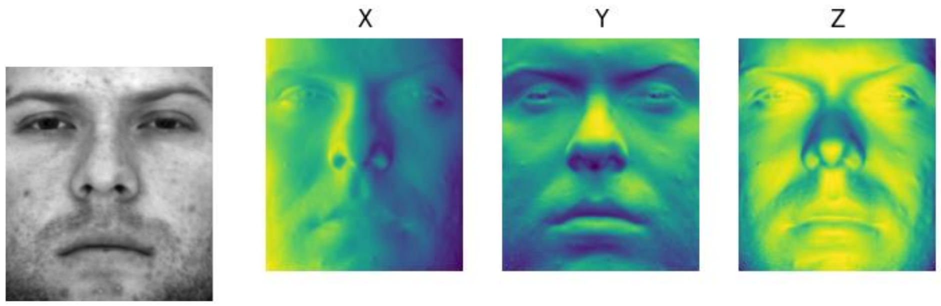
C: Display the final result of your stitching.
First transform the four corners of left image to see the position of transformed corners so that we know the output size of the image. Then transform the left image using homography, and transform the right image an offset so these two images can be stitched together.

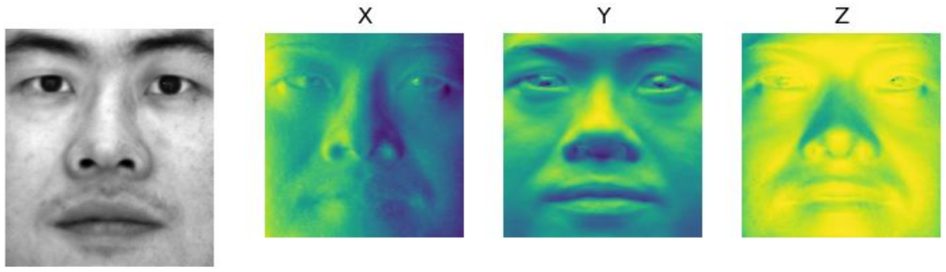


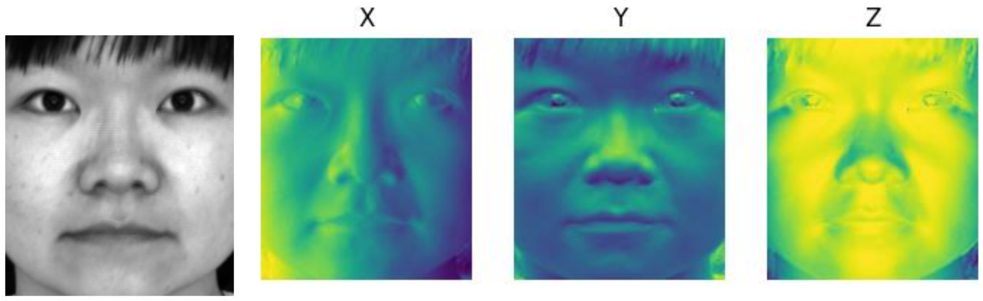
Part 2: Shape from shading

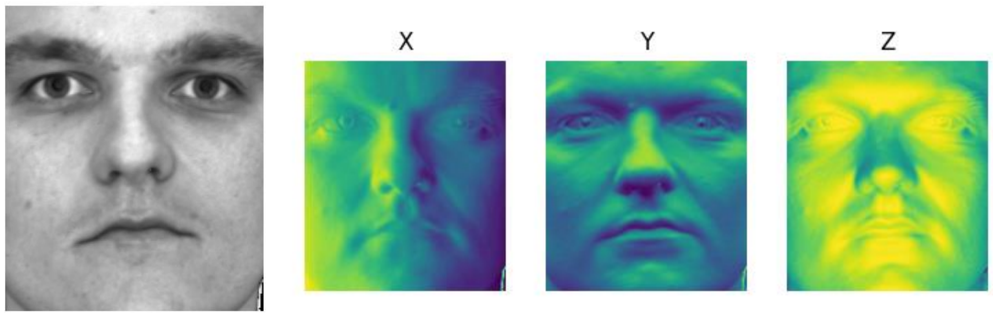
A: Estimate the albedo and surface normals

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?
3) What are some artifacts and/or limitations of your implementation, and what are possible reasons for them?
4) Display the surface normal estimation images below:

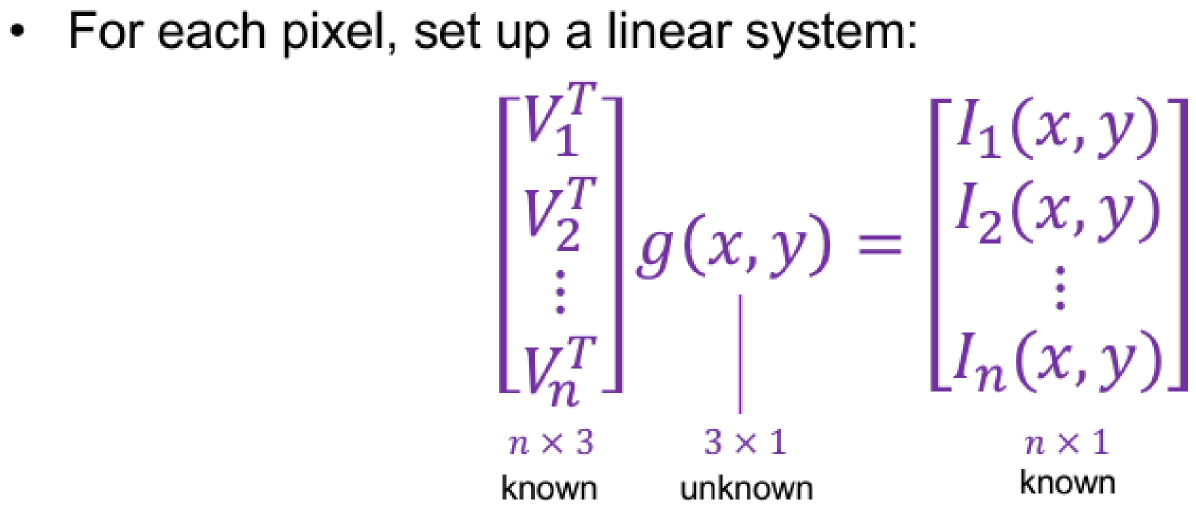








To find the surface normal and albedo of each pixel, we set up a linear system for each pixel. V is the light source vector and I is the pixel value. Since it’s an overdetermined system we can get a least-square solution of it. We can use pseudo-inverse to solve it.



- For each pixel, set up a linear system:

$$\underbrace{\begin{bmatrix} V_1^T \\ V_2^T \\ \vdots \\ V_n^T \end{bmatrix}}_{\substack{n \times 3 \\ \text{known}}} \underbrace{g(x,y)}_{\substack{3 \times 1 \\ \text{unknown}}} = \underbrace{\begin{bmatrix} I_1(x,y) \\ I_2(x,y) \\ \vdots \\ I_n(x,y) \end{bmatrix}}_{\substack{n \times 1 \\ \text{known}}}$$

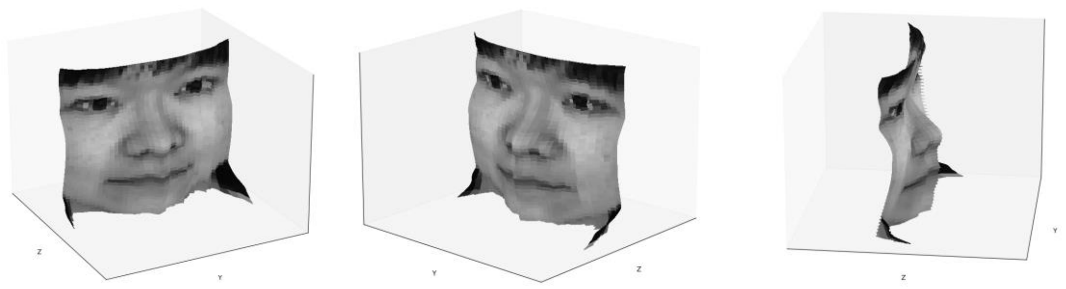
After that, for every pixel we have a 3 x 1 vector. The magnitude of it is the albedo value. We can turn this vector into and unit vector, which is the surface normal of the pixel

The limitation of this implementation is that we need at least two light source to solve the surface normal. Thankfully the least-square solution has closed-form solution and so it somewhat reduces the running time. It still takes a O(number of pixels) time complexity to compute the surface normal.

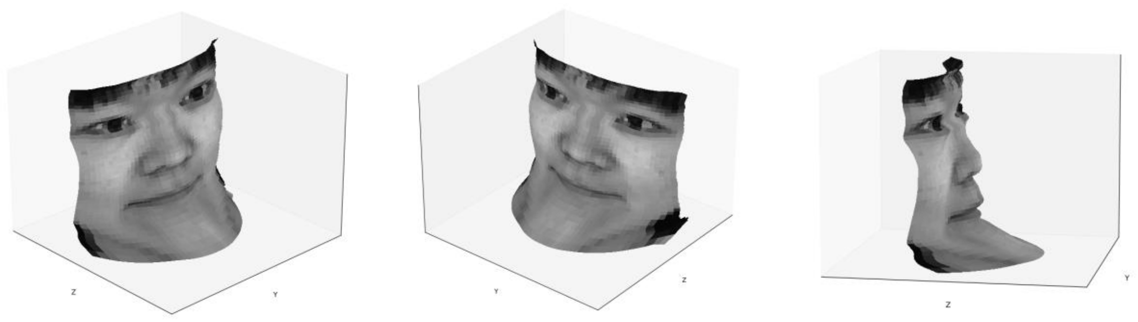
B: Compute Height Map

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.

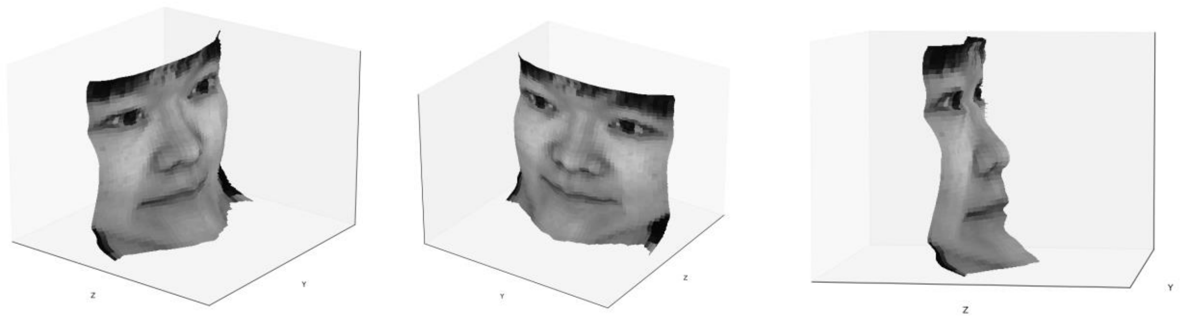
Row



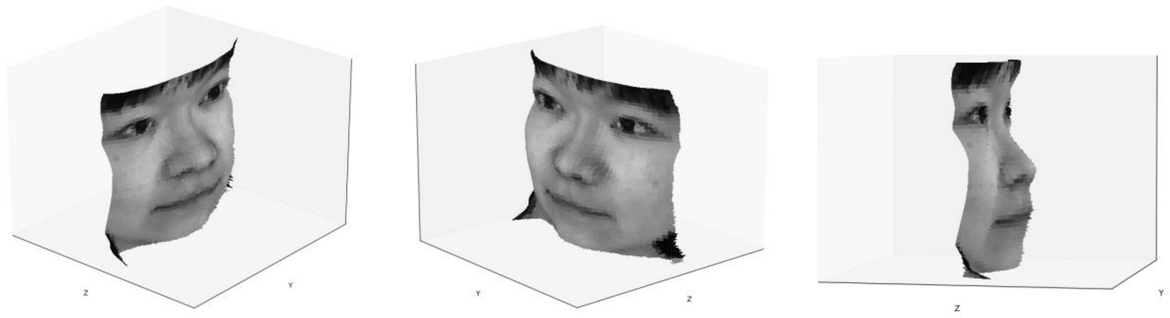
Column



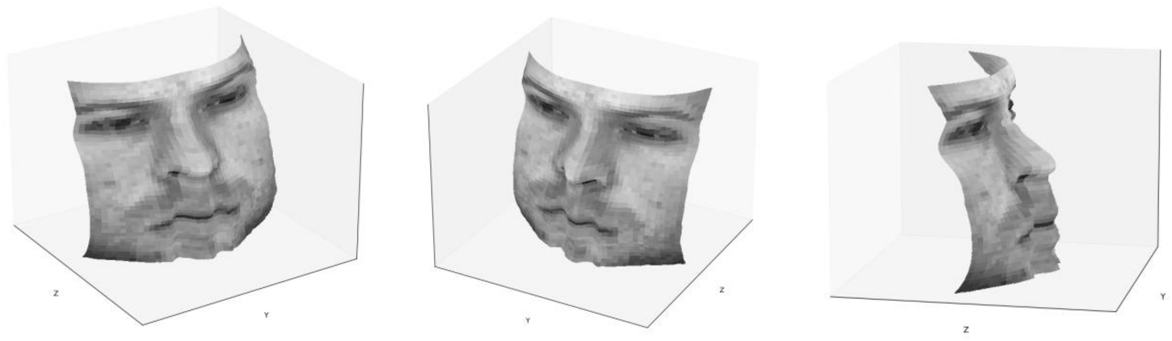
Average

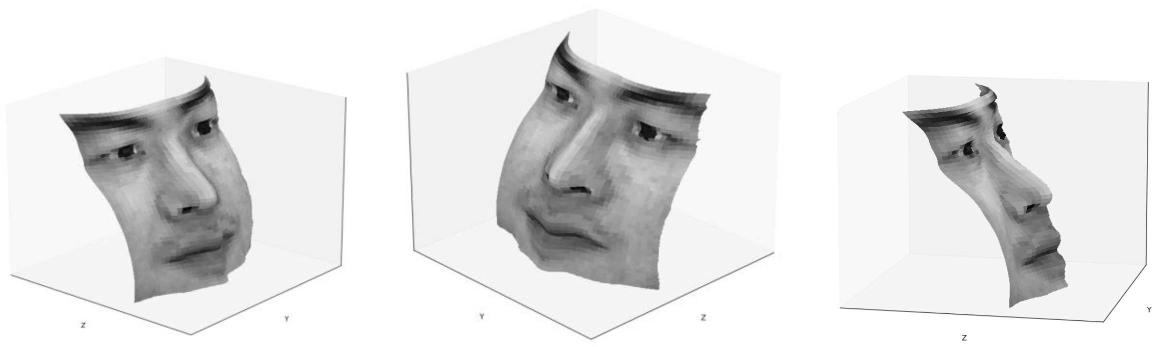


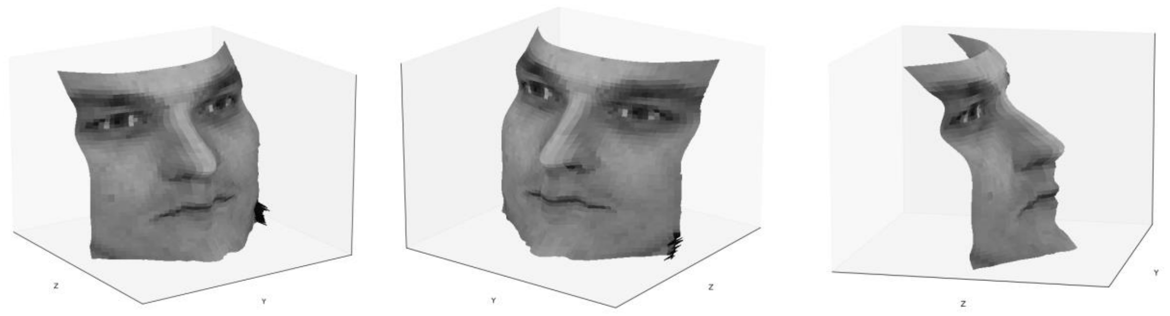
Random



Average

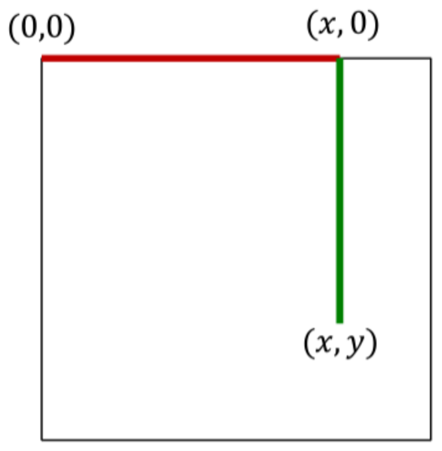


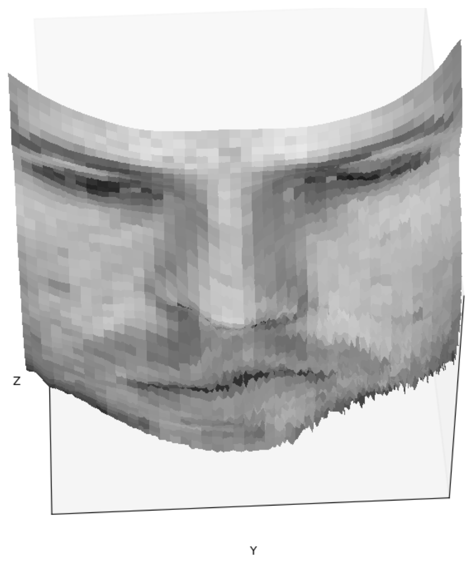




6) Which integration method produces the best result and why?

Since the integration accumulates the error, it's not a good idea just use a single path. A better way might be to average different integral path to the certain pixel, for example the random method. However, there's one special artifact that I want to point out. That is, even though we are averaging the path, the pixels that are close to the bottom right still get noisy. Since they are really far away from the origin, the averaged result is just the average of many noisy data, which could also be noisy. We can see this artifact at the screenshot below that pixels that are close the top left are smooth, whereas the bottom left pixels are noisy.





That being said, the random methods still performs better than methods that use single path. A workaround for the above artifact could be using a sliding window method. More specifically, we don't always integrate the surface normals from (0, 0). Instead, we integrate from the past N value, which is (i-N, j-N). This way we're using the computed result as the foundation, and reduces the uncertainty that we might have had if we integrate all the way from (0, 0).

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:

Integration method	Execution time
random	68.230 sec
average	0.148 sec
row	0.083 sec
column	0.069 sec

The row and column method integrate from the origin, and the value is always computed by the adjacent value. The idea is similar to Dynamic Programming, where we reuse the computed result. The time complexity would be simply O(n^2). The random method, however, is a O(k*n^3) algorithm, assuming it's a n x n image and k is the number of path to be averaged. Since for every pixel value we compute k paths, and it requires O(n) for each path.

C: Violation of the assumptions

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

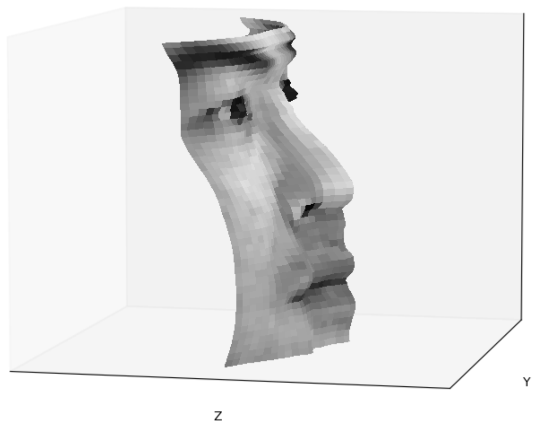
First, the image is not necessarily Lambertian object. Especially for some part of the face, it's so smooth that we can even see a light source reflected on the surface.

Second, the images should have the same objects across all source lights. That is, the expression of the face in each image should be exactly the same, which is not true for some dataset where some eyes are close while others aren't.

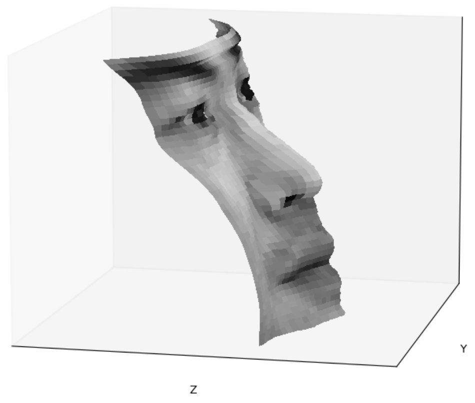
Also, orthographic model might not be true since it's using pinhole camera.

Lastly, we assume no shadows, which is often not true in most photos.

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.



(improved result)



(original reuslt)

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

For this dataset, I removed the viewpoints that contain too much shadow, and removed the viewpoints where the expressions are significantly different than others. We can see the improved result is brighter. Also, the improved result look more realistic and natural in terms of the contour and tilt angle of the face. It should be a straight face rather than a tilted face.