

CS543/ECE549 Assignment 3

Name: Kaylin Chen

NetId: tc49

Part 1: Homography estimation

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

I use `sift.detectAndCompute` to find the feature points and descriptions for both pictures, and then use `distance.cd` to find the euclidean distance, after that, I loop through every pair of the distance to find those that are under the threshold `des_distance.min()*15` as the matches.

For RANSAC, I follow the steps described on the assignment pages to construct `A` and find the homography by `np.linalg.svd`. I use a threshold=2 to find the inliers from the squared distance.

Finally, I use a matrix to shift both images in the same way, by dotting it with the homography matrix from RANSAC, I can keep the projective transform and the shifting at the same time, and then form the output by adding two shifted images pixel by pixel.

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.

number of homography inliers: 62

average residual: 0.52



C: Display the final result of your stitching.



Part 2: Shape from shading

A: Estimate the albedo and surface normals

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

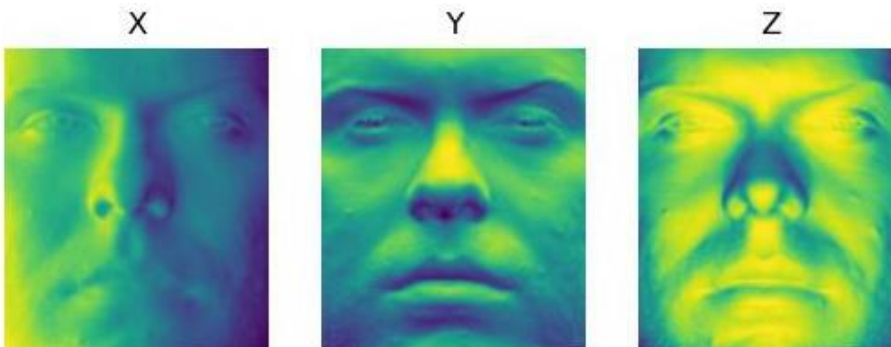


- 2) What implementation choices did you make? How did it affect the quality and speed of your solution?
- 3) What are some artifacts and/or limitations of your implementation, and what are possible reasons for them?
- 4) Display the surface normal estimation images below:

For the implementation of photometric_stereo, I obtain the stacked g which is a $3 \times \text{pixel}$ matrix by reshaping the input imarray from $h \times w \times N$ to $N \times (h \times w)$. The way I do this is transpose the array into $N \times h \times w$ first and then do the reshape. After reshaping, I use `numpy.linalg.lstsq` and `np.linalg.norm` to obtain g and albedo image. I also do reshape and stack after getting surface_normals from $(g / \text{albedo_image})$ to get the required shape of the normals.

I believe by using this method, the speed is faster than looping through all pixels and the quality looks good enough without any conspicuous artifact for the following steps.

yaleB01:



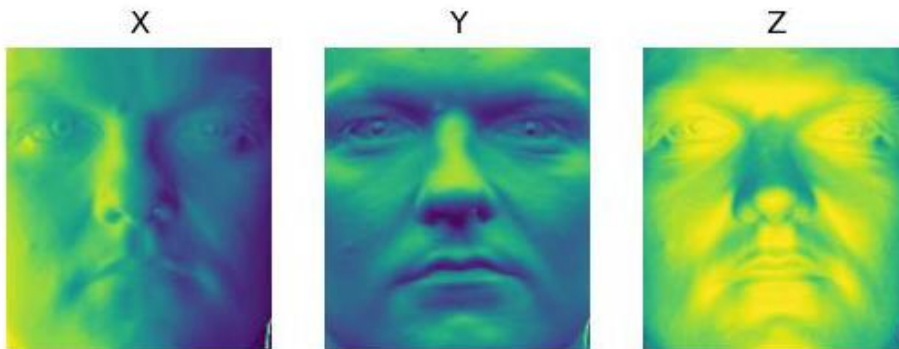
yaleB02:



yaleB05:



yaleB07:



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.

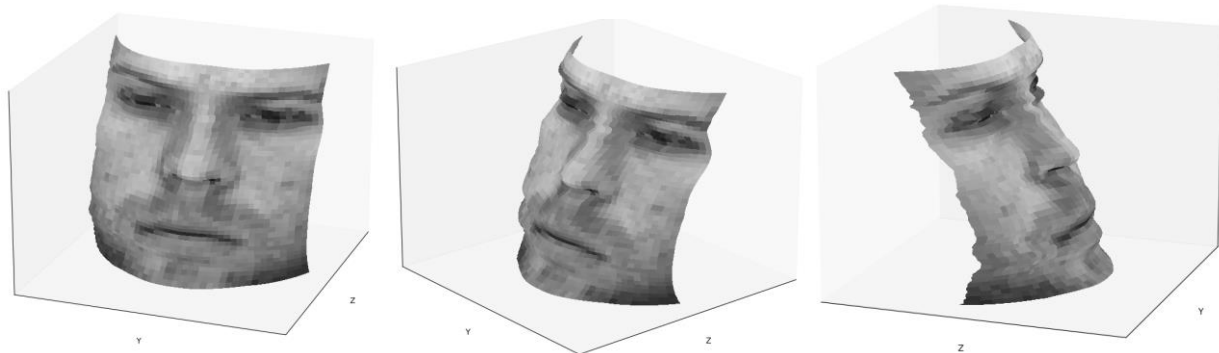
- 6) Which integration method produces the best result and why?
- 7) Compare the average execution time (only on your selected subject, “average” here means you should repeat the execution for several times to reduce random error) with each integration method, and analyze the cause of what you’ve observed:

I believe the random method produces the best result because averaging through multiple paths enable each pixel to grab the normal information from different directions.

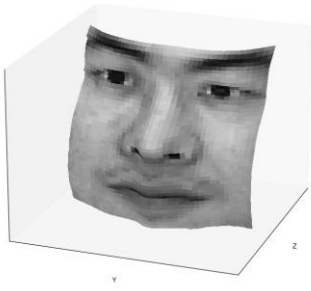
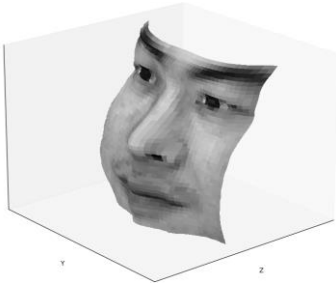


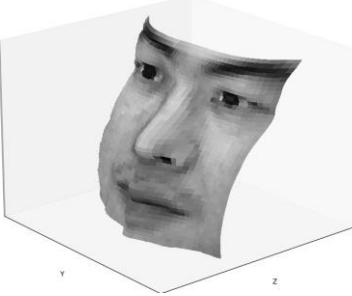
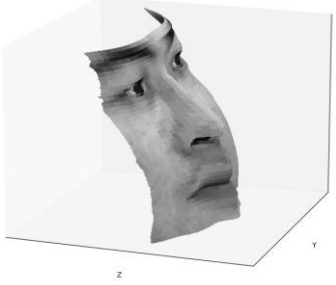
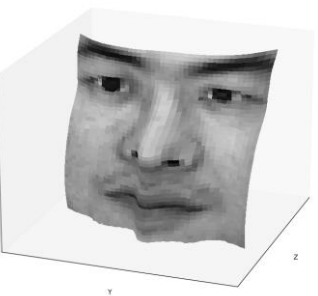
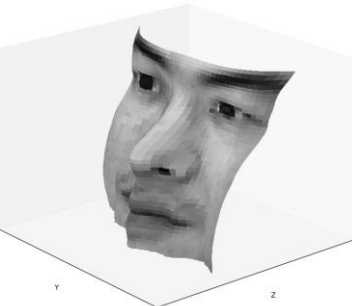
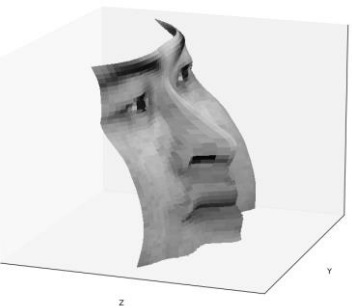

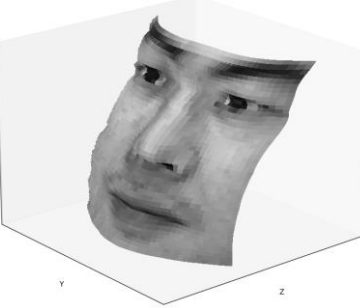
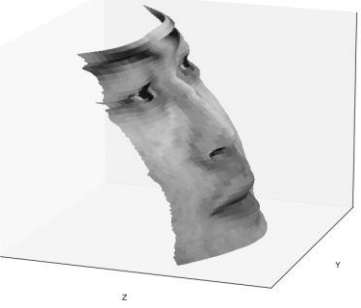
The random method consumes the most execution time because it needs to traverse multiple paths, whereas the row and column method only follows a single path. My method involves running 1000 paths, so the execution time is approximately 1000 times that of the row and column method. The average method, on the other hand, combines both row and column methods, resulting in a runtime roughly double that of each individual method.

Integration method	Execution time
random	0.287
average	0.00066
row	0.00019
column	0.00023

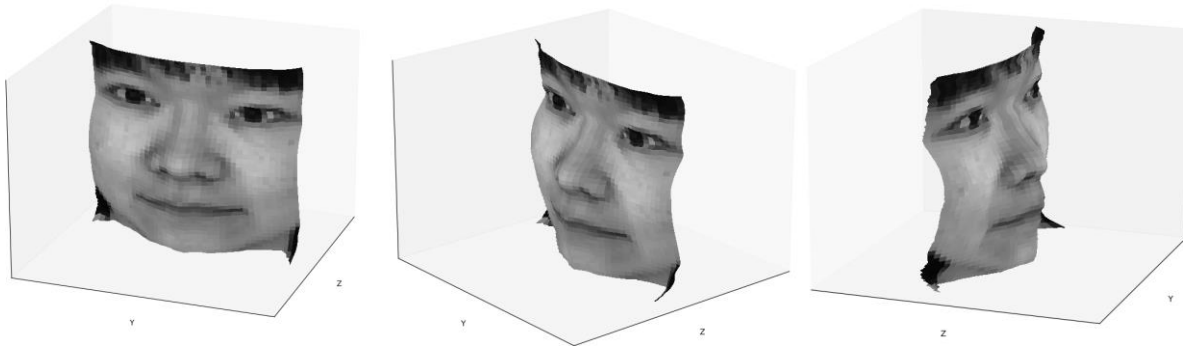
yaleB01:



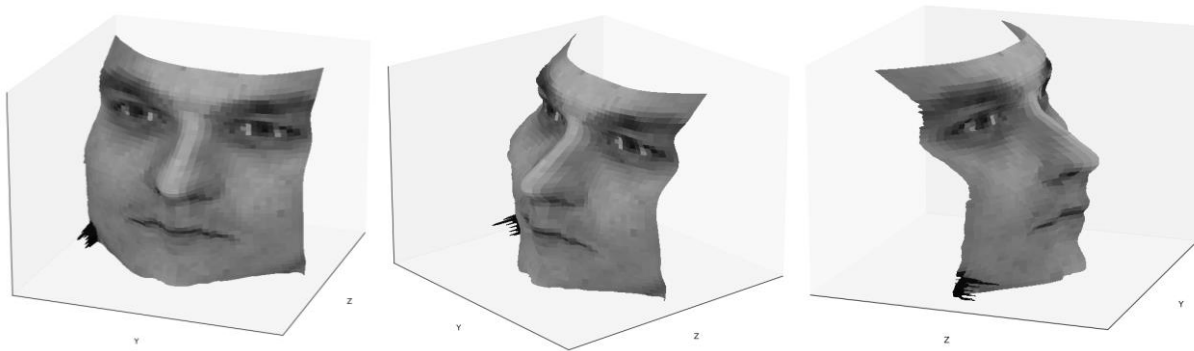
yaleB02:

Integration method	Output		
random			
average			
row			
column			

yaleB05:



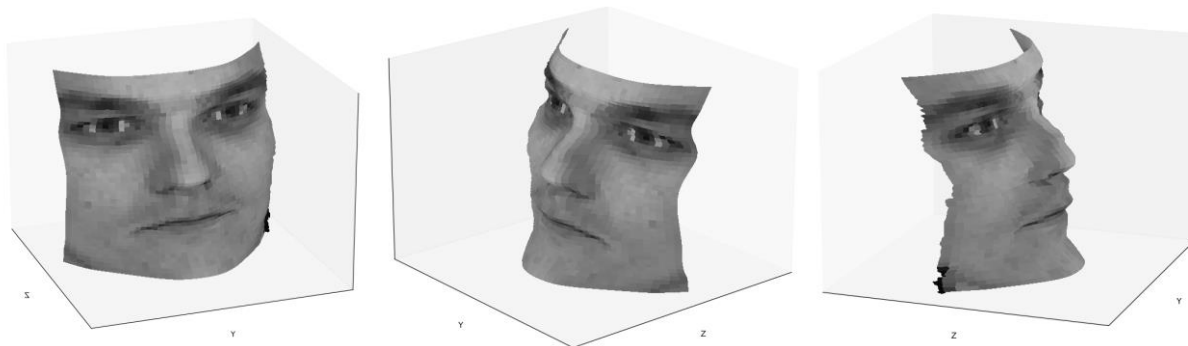
yaleB07:



C: Violation of the assumptions

- 8) Discuss how the violate the assumptions of the shape-from-shading method covered in the slides.
- 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.
- 10) Discuss whether you were able to get any improvement over a reconstruction computed from all the viewpoints.

The method covered in the slides assume that the object is a Lambertian object, however, in the dataset there are some pictures having shadow of the face that violate this assumption. I choose yaleB07 and improve the output by filtering out the images that have a mean value less than 30:



Part 3: Extra Credit

Post any extra credit for parts 1 or 2 here. Don't forget to include references, an explanation, and outputs to receive credit. Refer to the assignment for suggested outputs.