CS543/ECE549 Assignment 3

Name: ZihanHu
NetId: zihanhu2

Part 1: Homography estimation

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

To better align two images, I use the SIFT detector to find interesting points. Then I match the point from two different images by finding the minimum square of Euclidean distance. The first parameter I have for Euclidean distance matching is 8000. When the pairs of point distance are below this threshold, I will keep them as my input for the RANSAC. I got 40 matching pairs when I used this threshold.

Then I implement my RANSAC to find the best homography matrix. I set my iteration to 20000 in order to find the best matrix that includes the most inliers. The second threshold I set for maximum error for the inliers is 300 (my images have a pixel value range from 0-255). If the error of the point is less than 300, it can be counted as an inlier. As a result, I got 30 inliers for my best homography matrix.

Then, I use my homography matrix to align two images, I choose to use one image pixel value when there is an overlap. And the result is pretty good.

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.

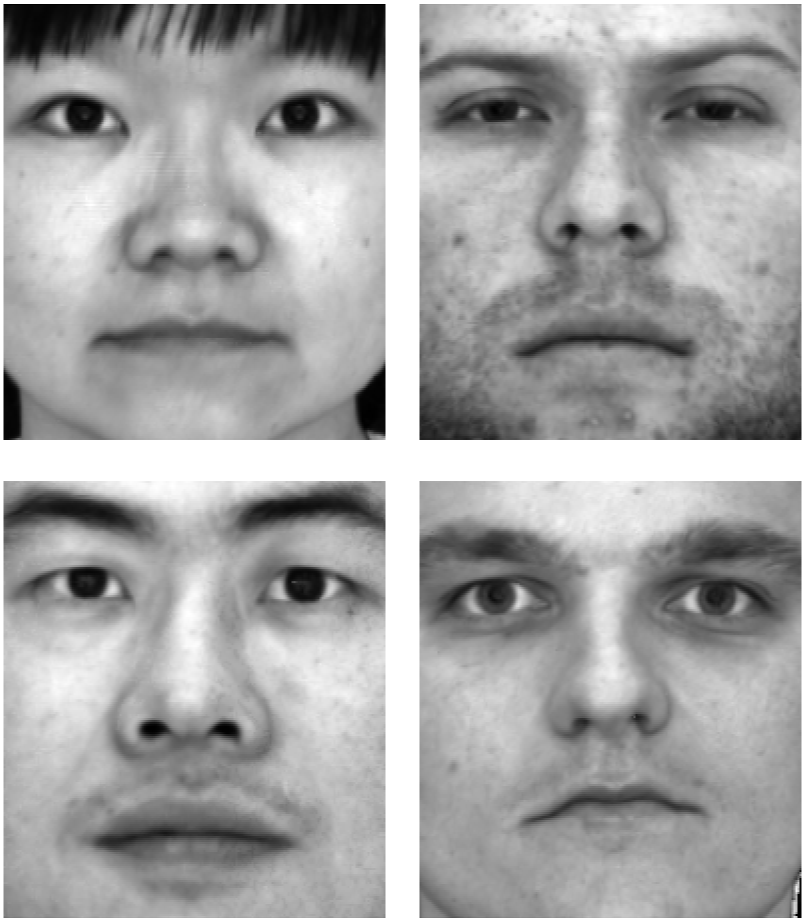
The number of **homography inliers is 30. The average residual for the inliers is** (my images have a pixel value range from 0-255) `52.39602556880587 it is 0.2 if the pixel range from 0-1`

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?

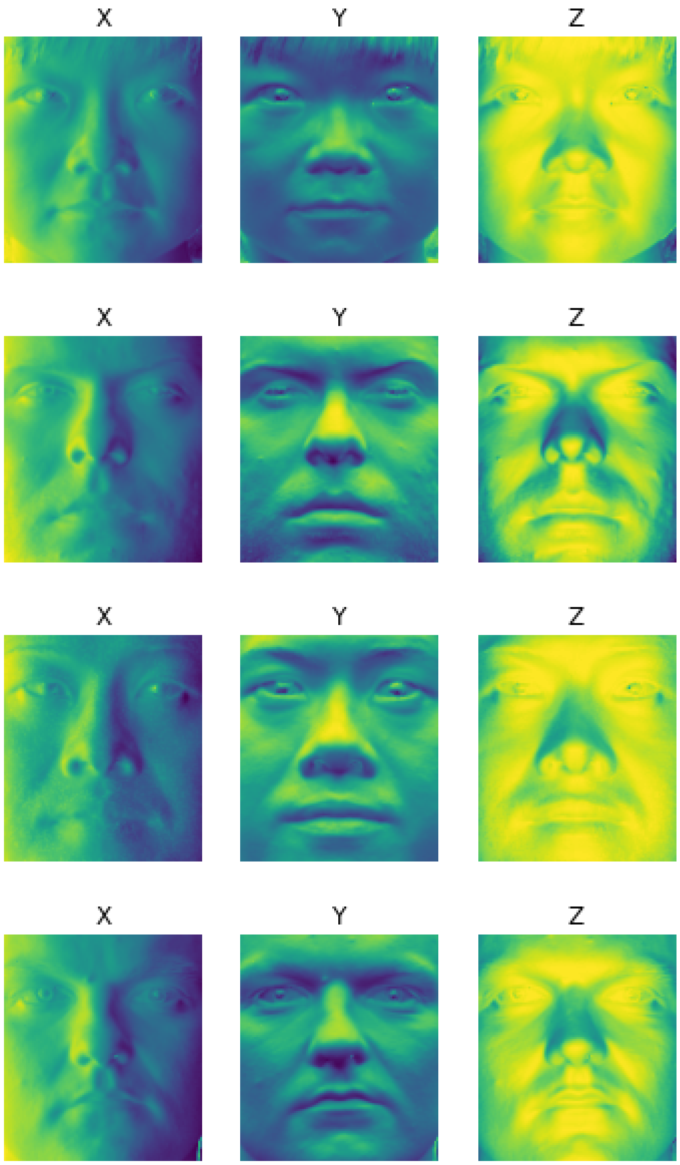
I preprocess the images by subtracting ambient images. Thresholding values below zero to zeros to avoid negative pixel values. Then I use a smart way to construct my matrix, which I just need one call of numpy.linalg.lstsq to get my solution. I stacked the unknown **g** vectors for every pixel into a 3 x npix matrix and got all the solutions with a single call to the NumPy solver. With all values in the valid range and subtracting ambient images, my output will have a higher quality. The way I solve the linear equation can increase the speed.

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

There are some artifacts. Faces are not exact Lambertian objects. Meanwhile, faces are not local shading models, which do not receives light only from sources visible at that point, they may also receive light from the reflection of the face itself. In addition, faces may not be captured by exactly the same camera/object configuration, which can also cause artifacts.

4) Display the surface normal estimation images below:

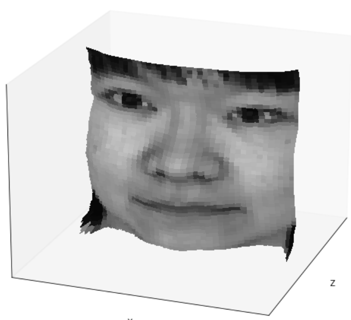
   Images are on the next page.

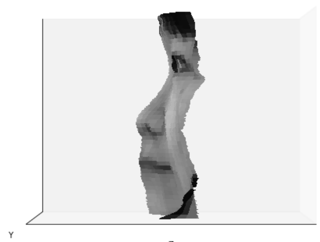


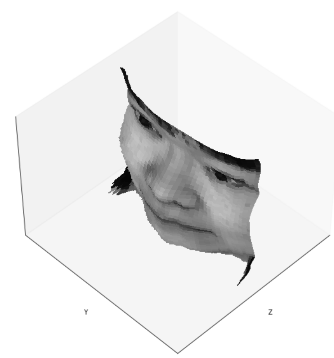
B: Compute Height Map

5) For every subject, display the surface height map by integration. Select one subject, and list height map images computed using different integration methods and from different views; for other subjects, only from different views, use 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.

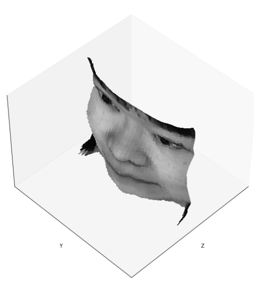
Random:(05)

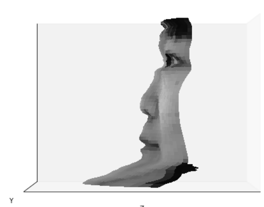


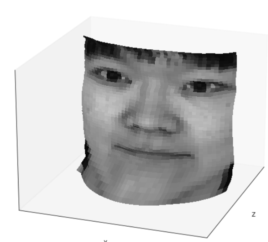




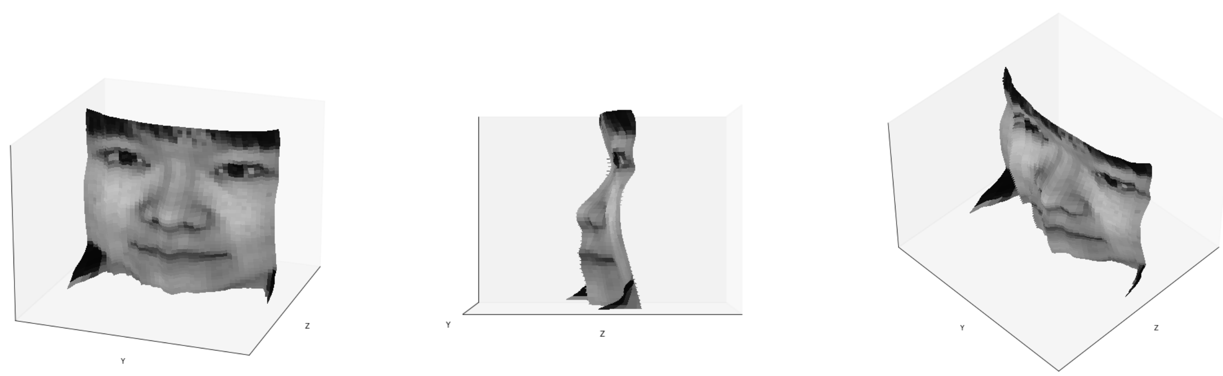
row:



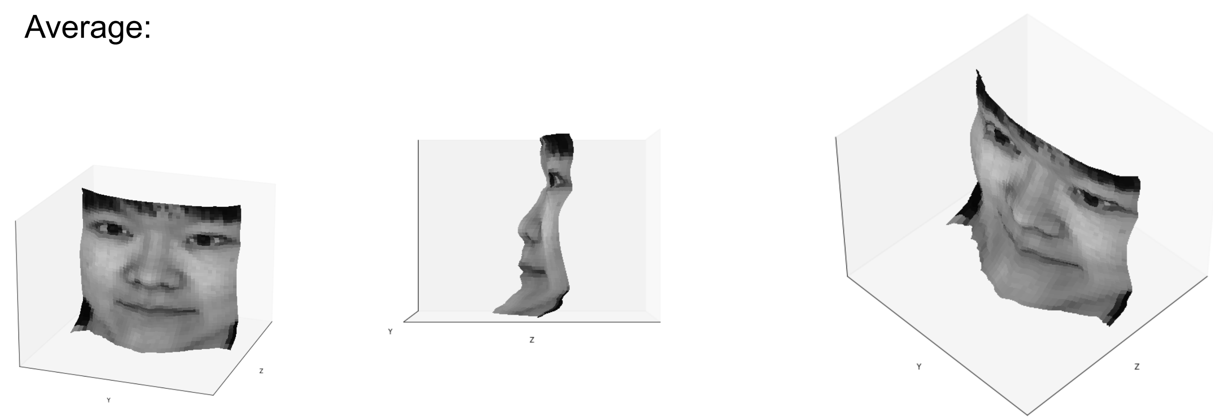




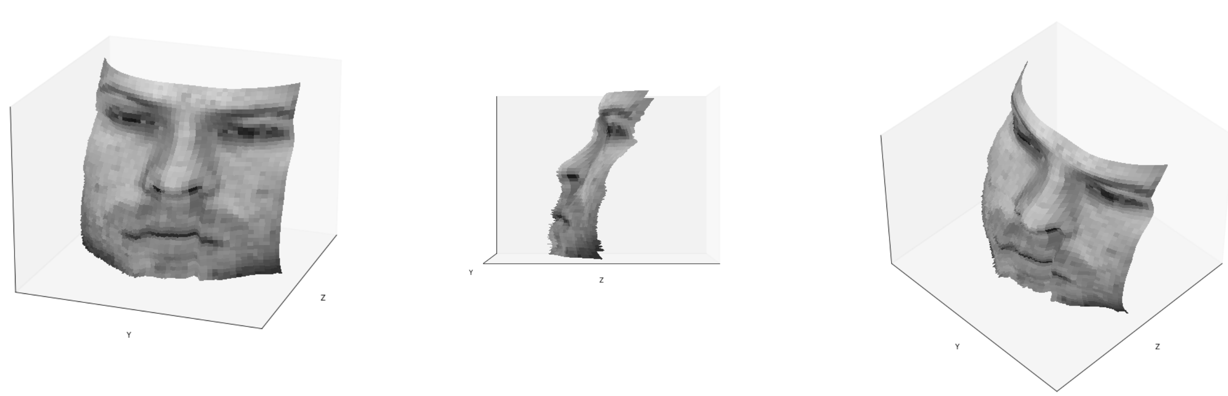
Column:



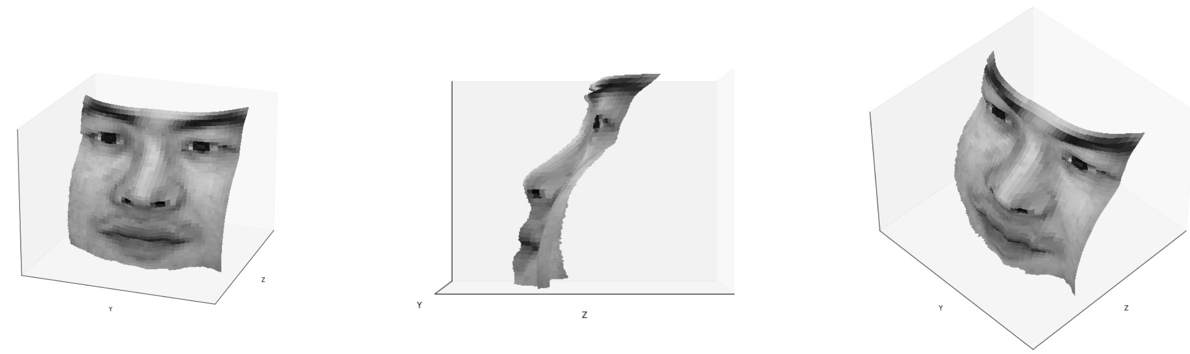
Average:



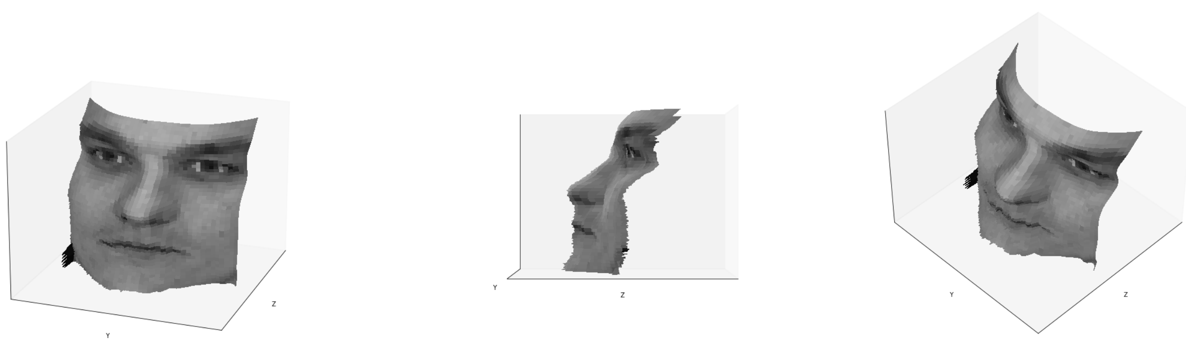
Random:(01)



Random(02)



random (07)



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

   From my result, I think the random way will give me the best result because it takes multiple different paths to calculate the height map, which makes it more robust to outliers when taking the average of random paths compared to just only rows or columns.

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

Integration method	Execution time
random	80.71656918525696
average	0.7949188232421875
rows	0.7283170223236084
columns	0.7694311141967773

For row, column, and average integration method, the height map can be easily calculated by only one or two paths. However, for my random method, I average 40 different paths to calculate the height map, which will make the execution time long.

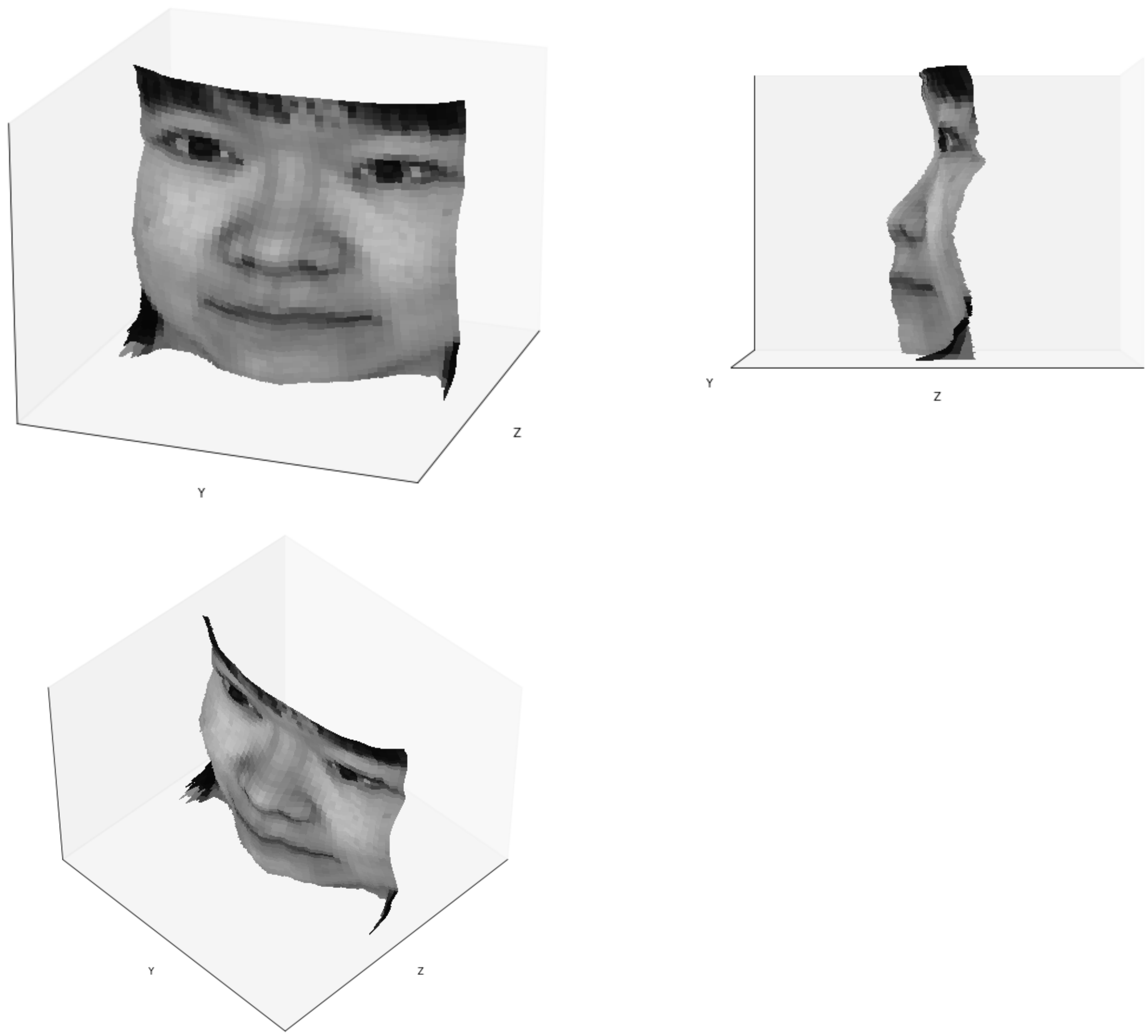
C: Violation of the assumptions

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

   We can clearly see that the face are not Lambertian objects. In the provided data set, we can see the specular reflection at the nose and some other places. Those pure white surfaces violate the assumptions of the shape-from-shading. Meanwhile, we cannot guarantee that faces are not moving, which may also violate the assumptions of shape-from-shading.

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.

   I choose to delete several images that have strong specular reflection effects. Meanwhile, I run my random method several times to ensure the average quality of my output image.


Y
Z
Y
Z
Y
Z

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

I think I am able to improve the result a bit, but the improvement is not that obvious depending on which images I delete. However, it looks like the place having a specular effect has some improvement.