CS543/ECE549 Assignment 3

Name: Emmanuel Gallegos
NetId: eg11

Part 1: Homography estimation

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

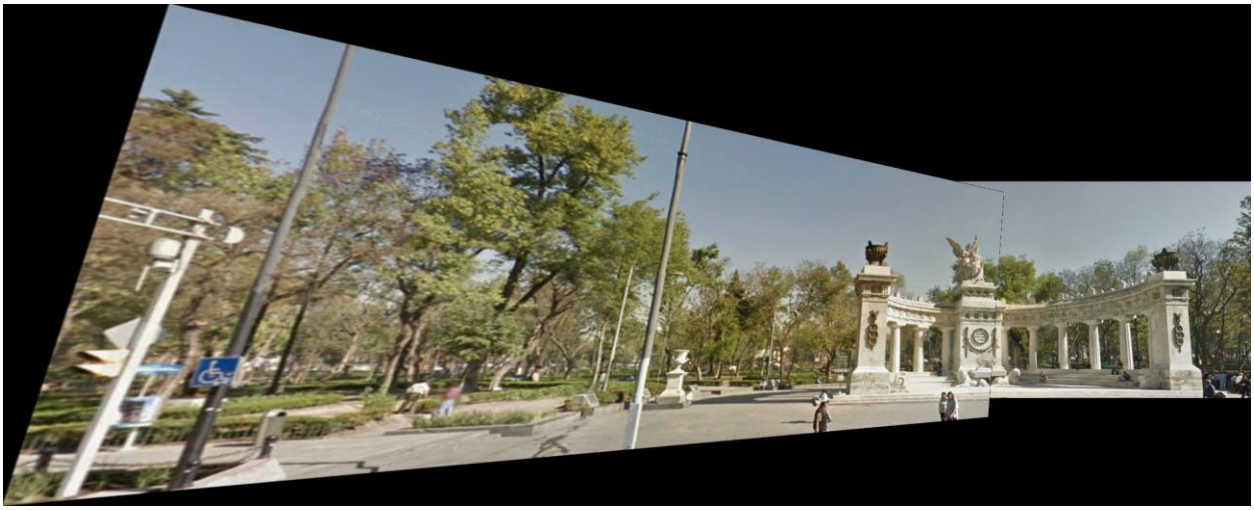
My solution followed the algorithm provided in lecture and in the assignment description. I used OpenCV's SIFT keypoint and descriptor module, along with BFMatcher using L2 Norm to generate putative matches. I followed the RANSAC algorithm without the parameter to stop early if a certain number of inliers were detected. For displaying the image, I copied values from both images to one large numpy array, picking values from the transformed left image when both images had defined pixel values.

My parameters were as follows:
Number of putative matches: 400
Number of points to draw in homography estimation: 4 (minimum)
Number of iterations in RANSAC: 1000
Inlier Threshold: 0.5

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 inliers: 53
Average inlier residual: 0.139

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 chose to calculate albedo and surface normal using the algorithm described in lecture. I did not modify the calculation at all from what I can tell.

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

   This implementation does assume Lambertian objects, or that the faces scatter light equally, and that there are no occlusions or shadows. However, for images with light

coming from the side, in particular, there are often shadows cast across part of the face, which do not allow us to get correct readings in the shadowed areas.

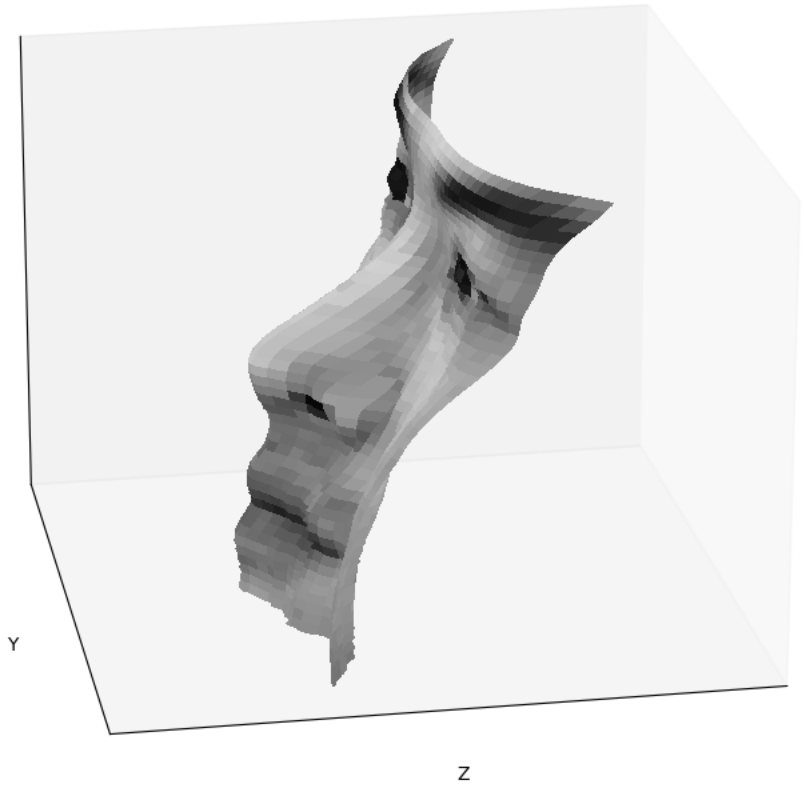
4) Display the surface normal estimation images below:

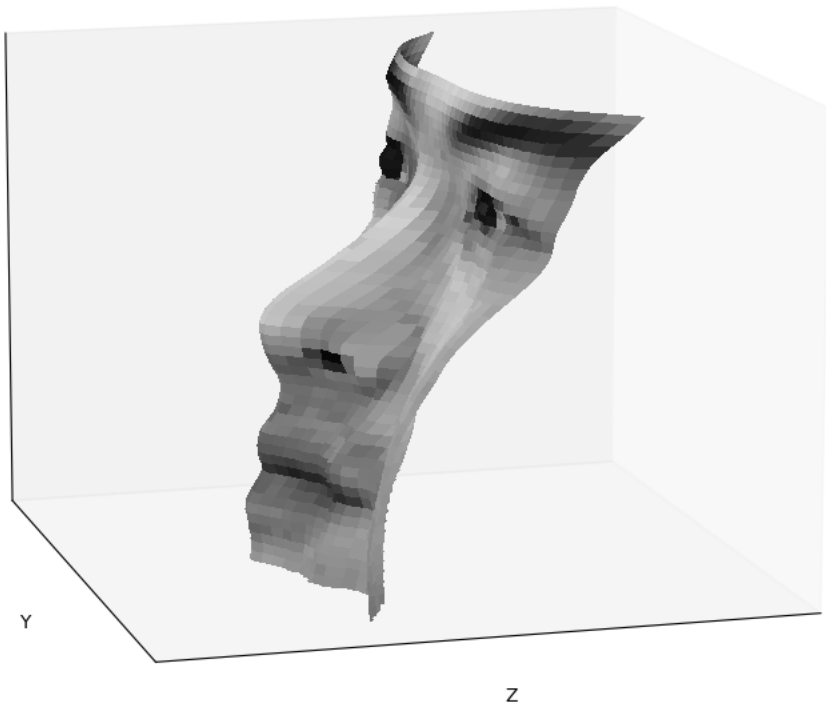
B: Compute Height Map

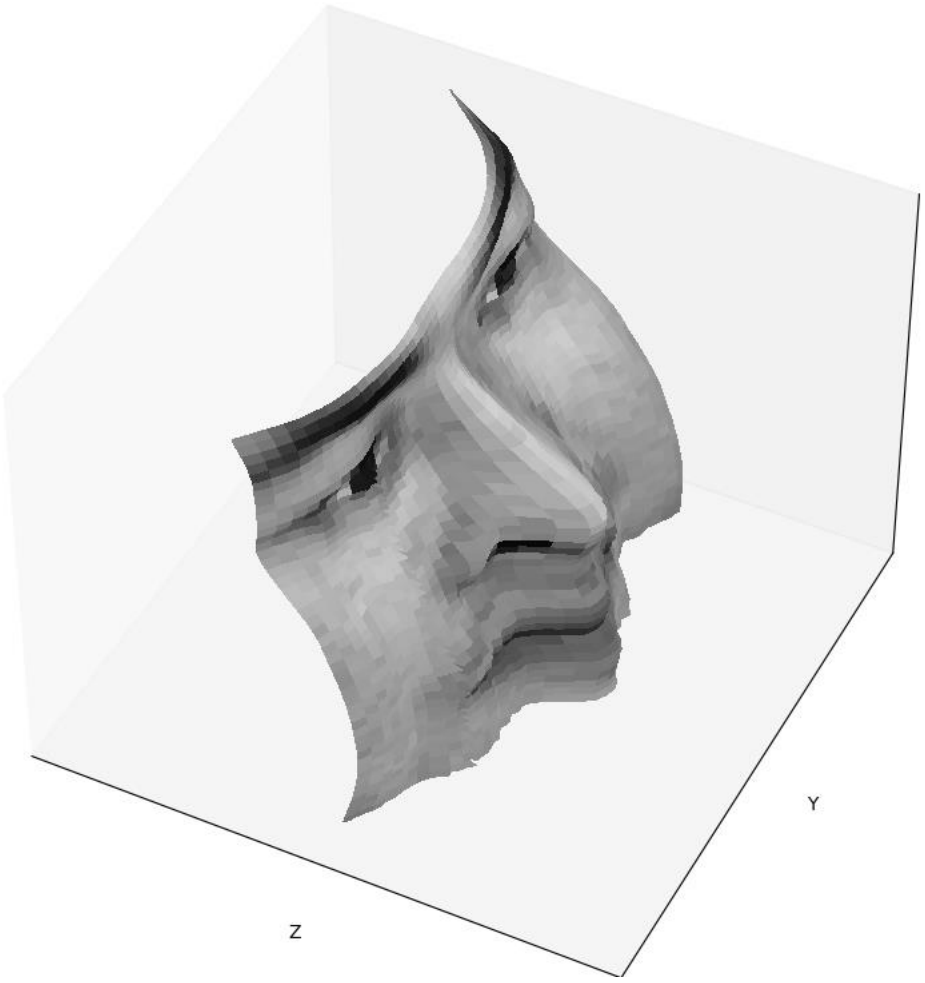
5) For every subject, display the surface height map by integration. Select one subject, list height map images computed using different integration method and from different views; for other subjects, only from different views, using the method that you think performs best. When inserting results images 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.

Selected Subject:

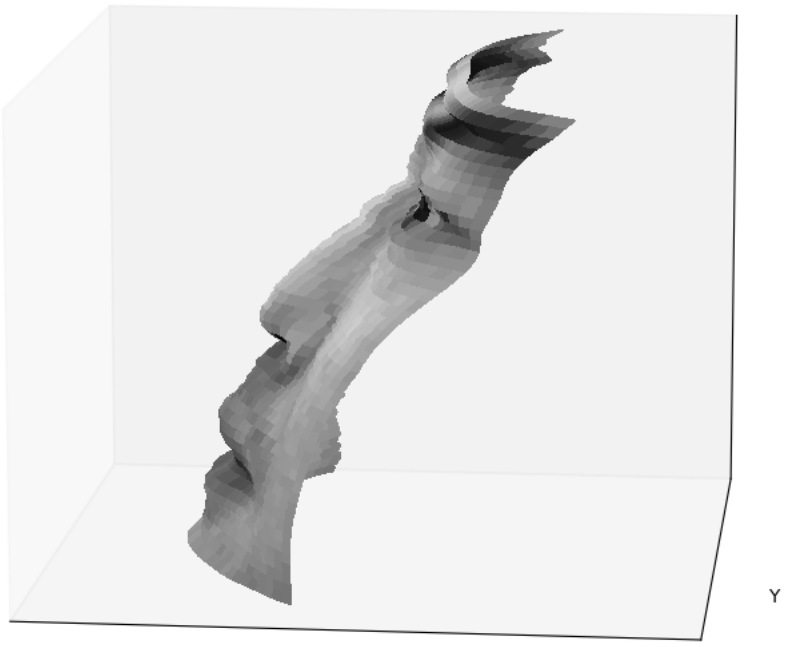
Column Method:

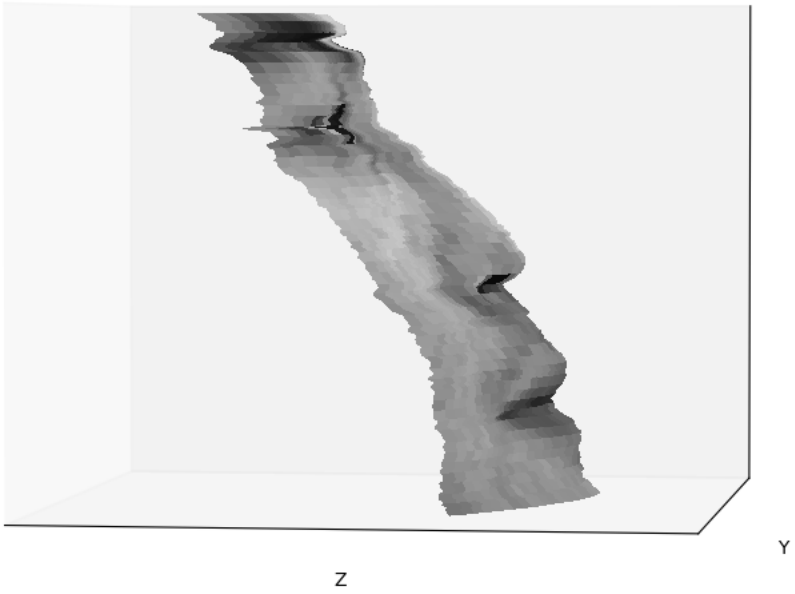

Y
Z


Y
Z

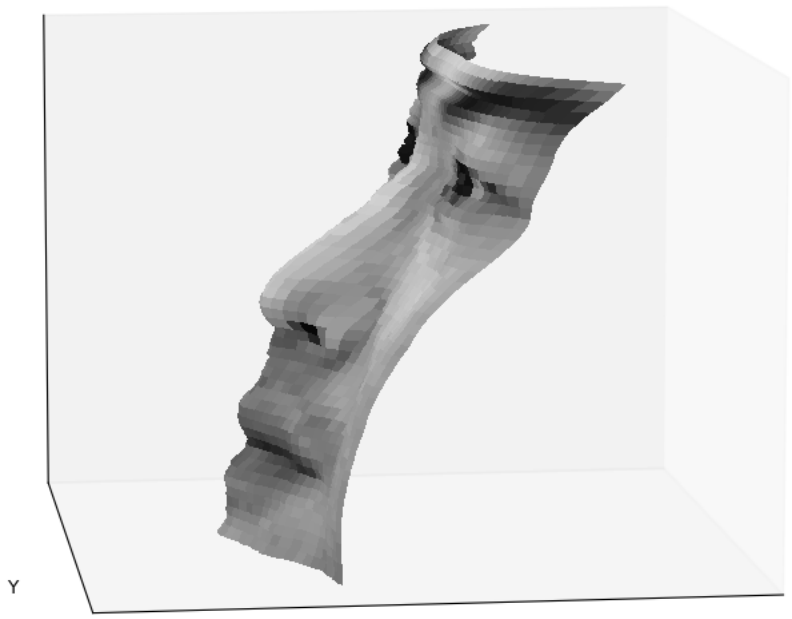


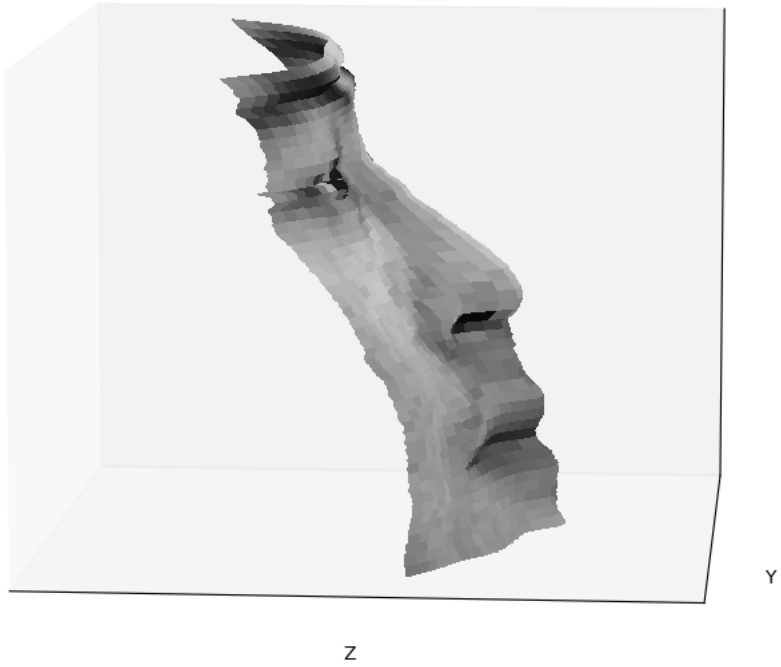
Row Method:



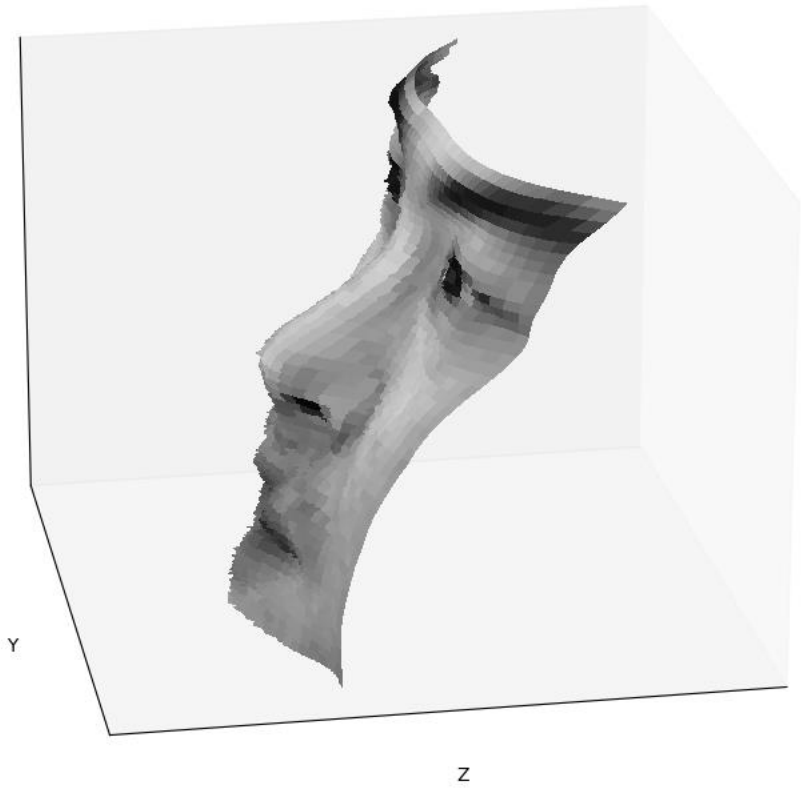


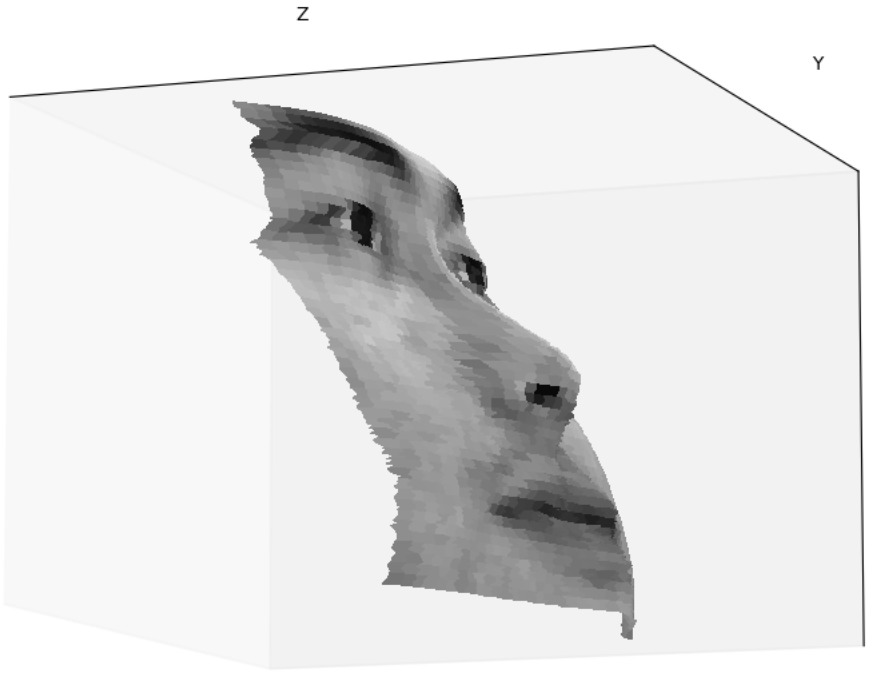
Average method:


Y
Z


Y
Z

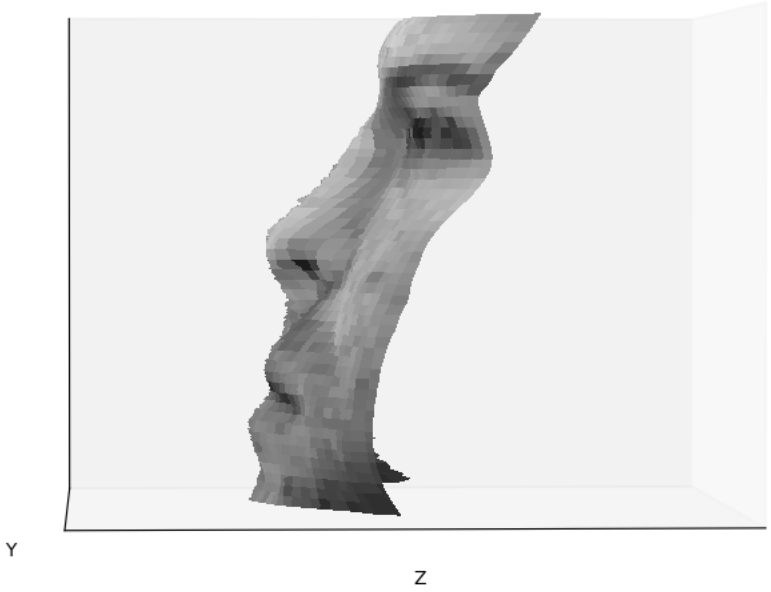
Random Walk Method:

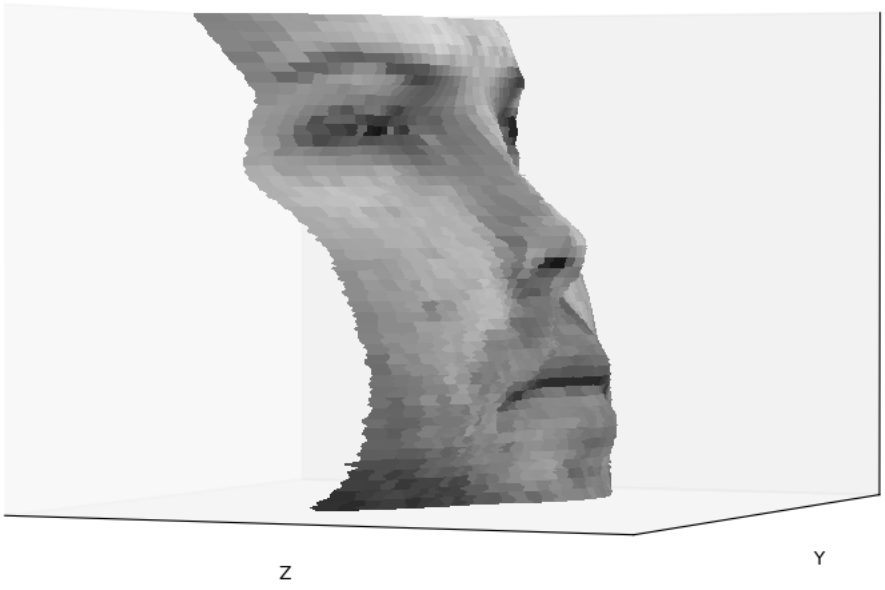

Y
Z


Z
Y

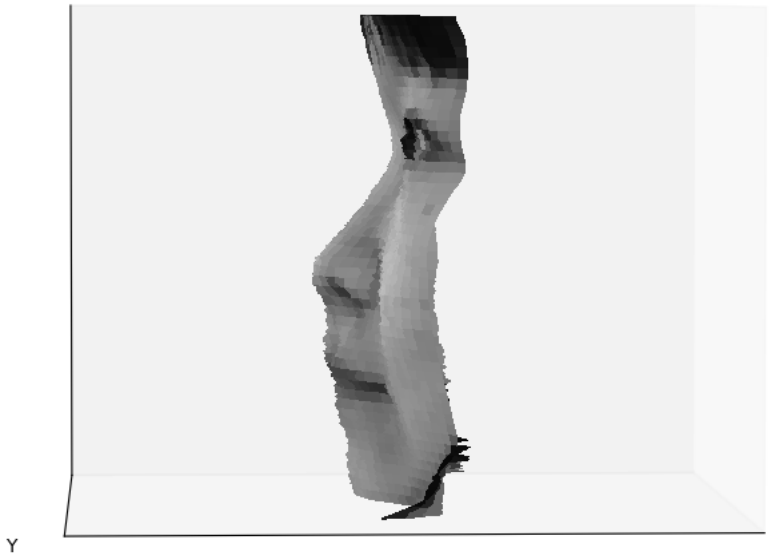
Other Subjects:

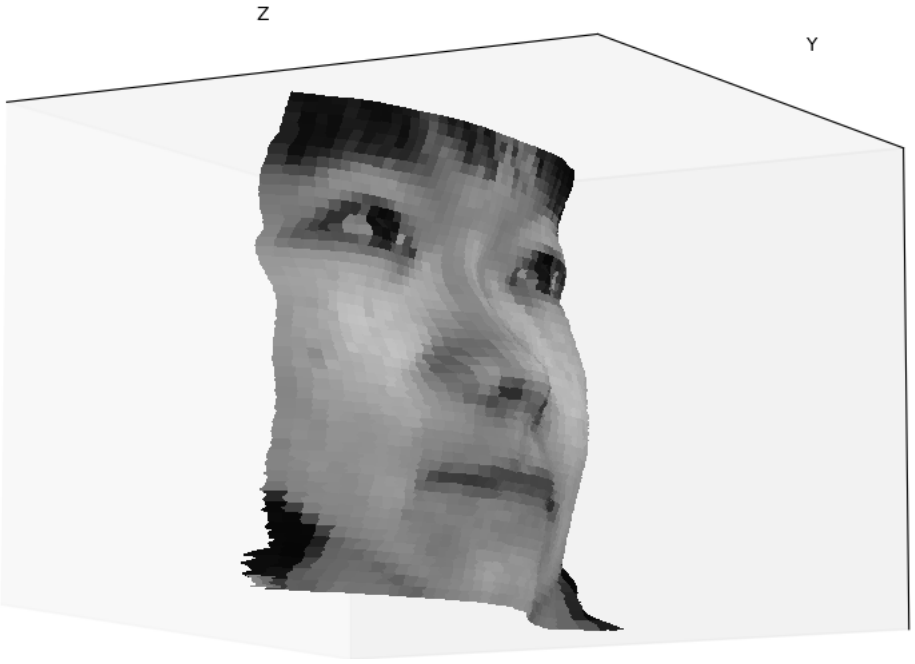
Subject 1: Random Walk Method

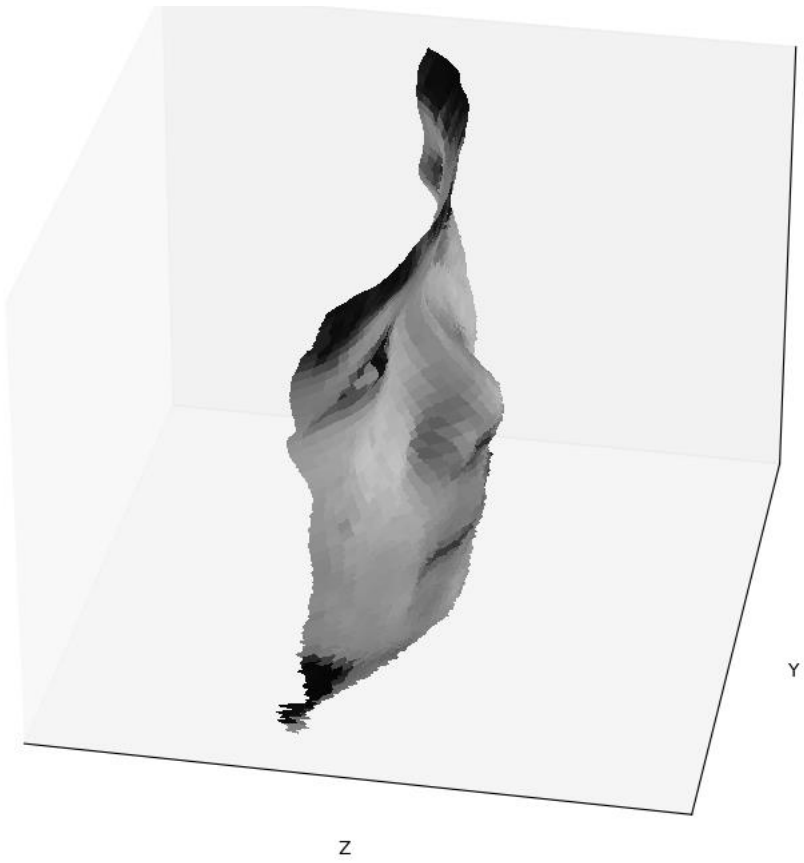




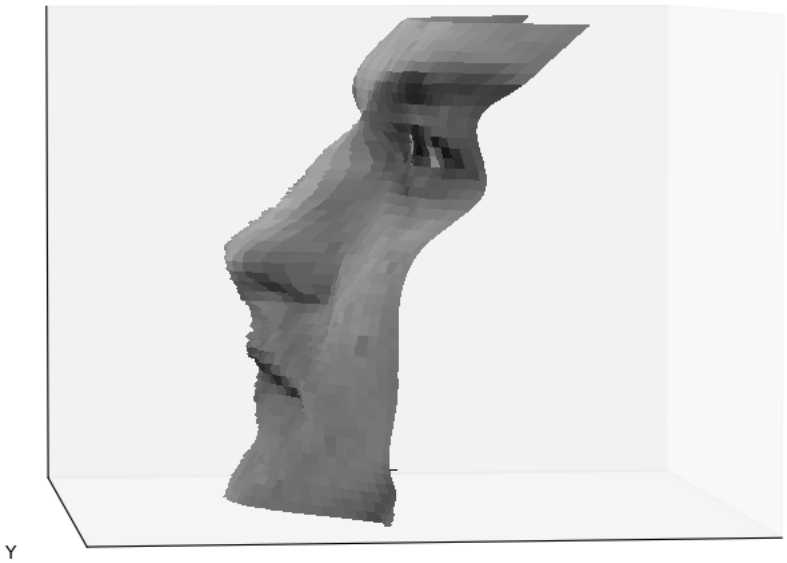
Subject 2: Random Walk Method


Y
Z

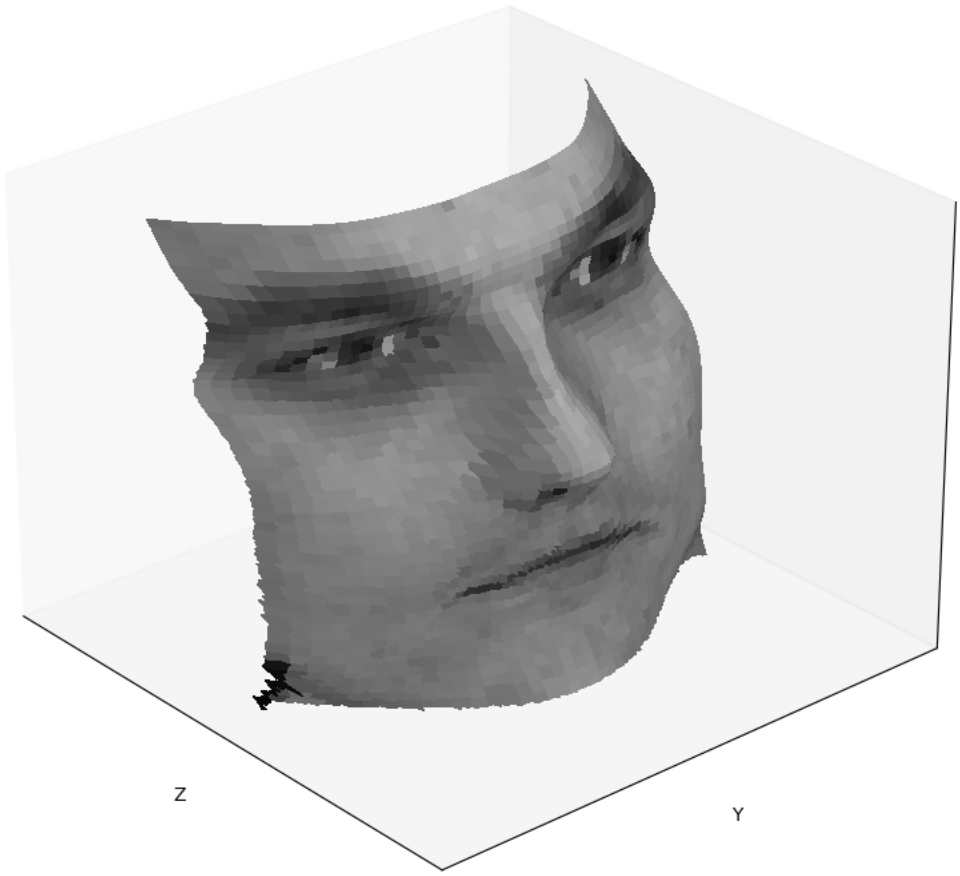

Z
Y



Subject 3: Random Walk Method


Y
Z


Z
Y



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

   The random integration seemed to work best for surface reconstruction. I believe this is because the derivatives we calculate (fx and fy) are only approximations, and thus walking down them in different orders to compute integration gives different results. Thus, by randomly walking down the two derivatives as we sum and averaging the results, we get a better result that accounts for our approximations best.

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	55.1419s
average	0.0009187s
row	0.0009997s
column	0.0s

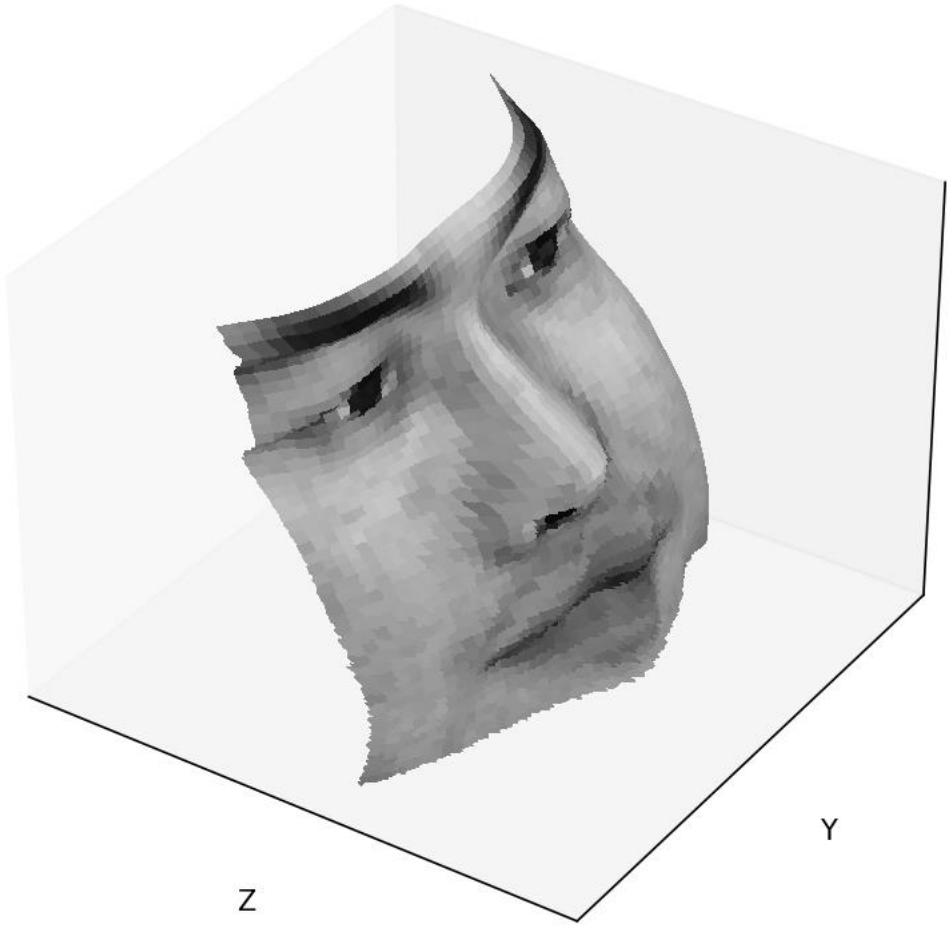
I believe the reason that random took so long is because I had to manually perform nested loops that, for each pixel location, performs several random walks in order to calculate its final value. These random walks may be short for the pixel at (0,0), but takes X+Y steps for the pixel in the opposite corner. So our complexity is O(X*Y*N_RANDOM_WALKS*(X+Y)). Assuming a square image, this would be O(X^3 * N_RANDOM_WALKS)

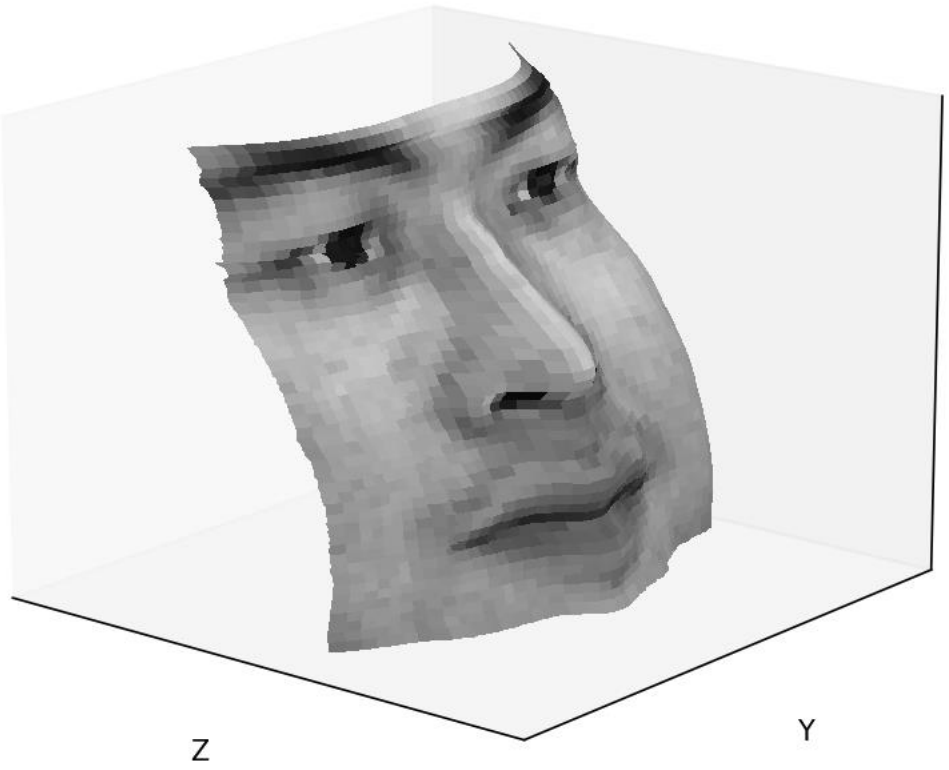
C: Violation of the assumptions

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

   One of the assumptions is a Lambertian object. Human faces are not Lambertian, however, and their reflective properties (such as wet eyes, or oil on the face, as well as different facial textures) violate this assumption. There is also the assumption of a local shading model, but occluding features (like nose or even lips) can cast shadows on the face that don't let us recover correct intensity data in the shadowed regions.

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.

    I didn't really see too much improvement, honestly, though perhaps the face is a bit less right skewed than without the improvements.