Name : Zong Fan

Part-1 : Estimate the albedo and surface normals

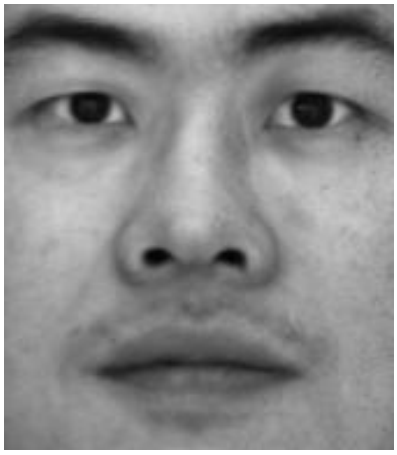
1) Insert the albedo image of your test image here:

YaleB05:

YaleB07

YaleB01

YaleB02



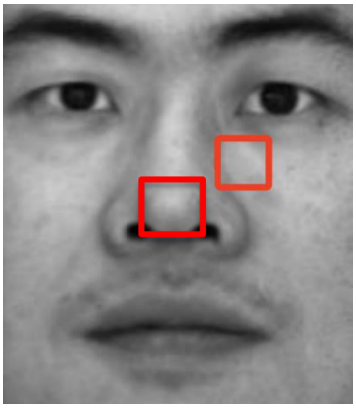
2) What implementation choices did you make? How did it affect the quality and speed of your solution?

First, reshape and transpose the image intensity array into shape (n, h*w), where n is the number of images, h and w is the image height and width of image. Then according to Lambert's law, compute g (equals to albedo * surface normal) by solving the least square solutions of g*v=I. Now g has shape (3, h*w). Transpose and reshape g into shape (h, w, 3). Then compute normal along the last dimension of g to obtain the albedo of each pixels with shape (h, w).

Because it uses numpy matrix operation to compute the whole image at once instead of looping over pixels, the computation is fast. Based on the generated image, the estimated albedo and surface normals have relatively good quality.

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

There are artifacts reflecting the regions which is affected by non-diffusive reflection characteristics. For example, in YaleB02 image, there are bright patch on the nose (marked in red rectangles). This may reflect the highlights caused by specular reflection in this region.



4) Display the surface normal estimation images below:

From left to right: surface normal x => y => z

YaleB05:

YaleB07

YaleB01

YaleB02

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

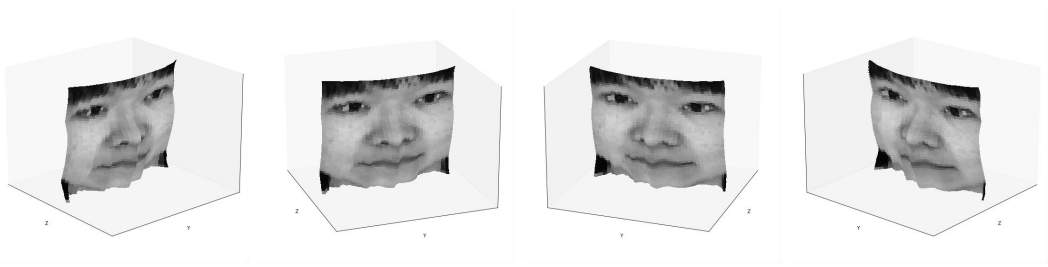
appropriately to keep the file size manageable -- but make sure that the correctness and quality of your output can be clearly and easily judged.

For row/column integration, just as the method shown on lec4 slide 39.

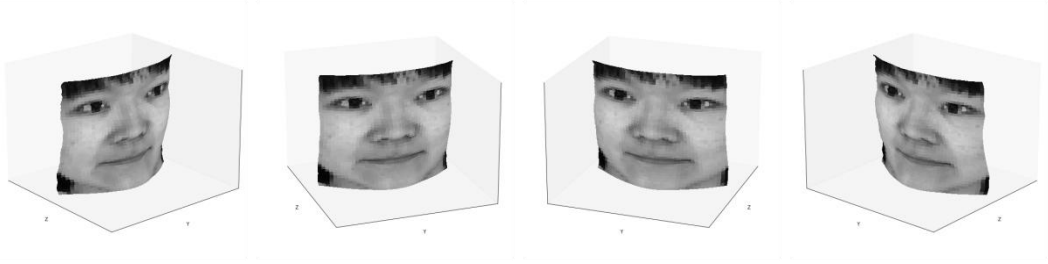
For random integration, use random walk for each pixels on the image. For pixel (x, y), there is x+y moving right/down steps from origin to this point. We random shuffle the x+y steps and lead the path to the pixel along this shuffle route. To compute all pixels. We employed nest looping for simplicity. The speed is slow. However, it could be processed parallelly because the route to each pixel is independent.

For YaleB05, from left to right: elevation angle=20, azimuth angle=-40, -20, 20, 40

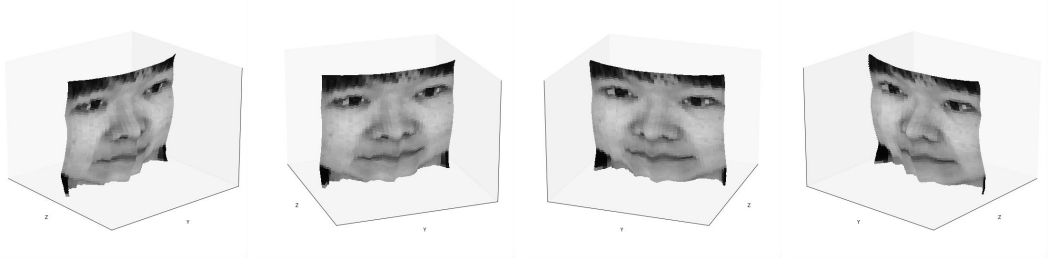
A. Integration by 'Column' method:



B. Integration by 'Row' method:

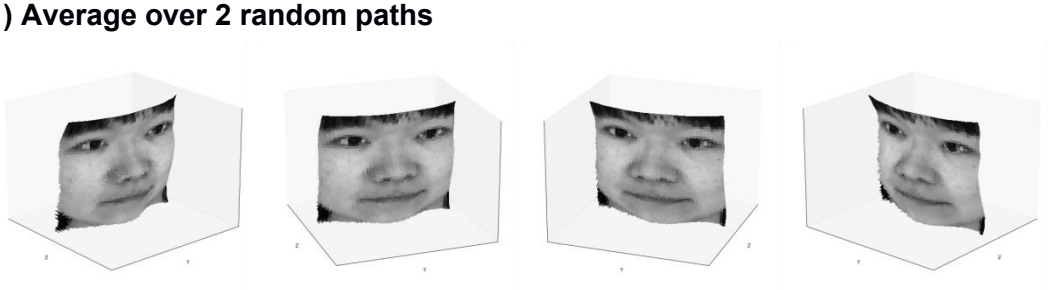


C. Integration by 'Average' method:

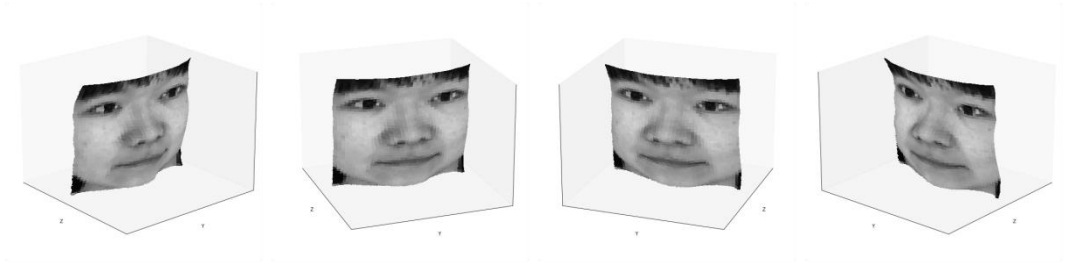


D. Integration by 'Random method':

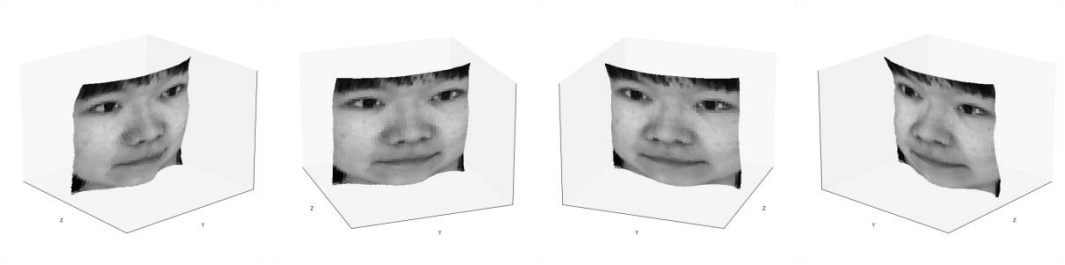
1) Average over 2 random paths



2) Average over 5 random paths

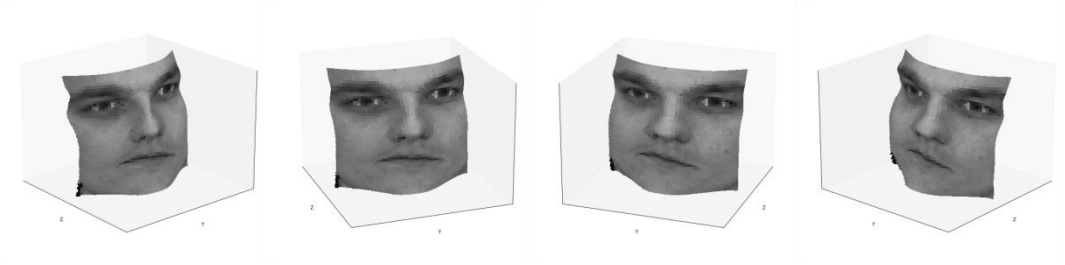


3) Average over 10 random paths



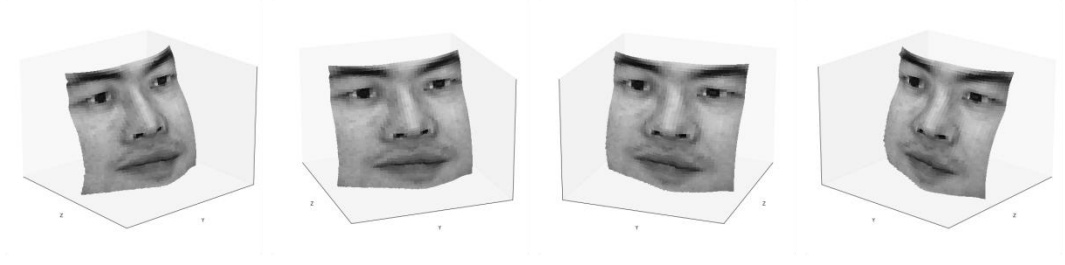
As we can see, random integration averaged on 10 paths achieves best height map estimation. The edges are smooth and the 3D depths looks reasonable.
For comparison, random integration averaged on 5 paths also achieves comparable performance while its speed is much faster than previous one, since the computation of each path is serial.

YaleB07



YaleB01

YaleB02



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

Random integration averaged on 10 paths.

Because this method could estimate the height map on the entire surface of the normal image rather than only using horizontal and vertical edges. It could alleviate the disturbances caused by horizontal or vertical artifacts.

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	2.8s/path
average	0.1s/path
row	0.1s
column	0.1s

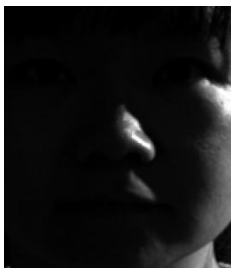
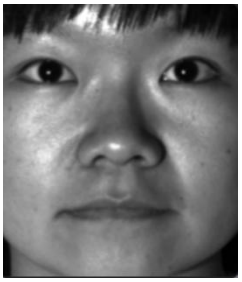
Test on 1.4 GHz Quad-Core Intel Core i5.

Random integration is slowest, since it compute height value pixel by pixel via 2 nested looping. The other 3 methods compute the height map directly via matrix calculation which is accelerated via parallel computation.

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.

Some images are totally occluded which lose too much information. Some regions contain specular reflection highlight patches. Like the following figures in YaleB05.


(yaleB05_P00A-110E+15)

(yaleB05_P00A+020E-10)

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.

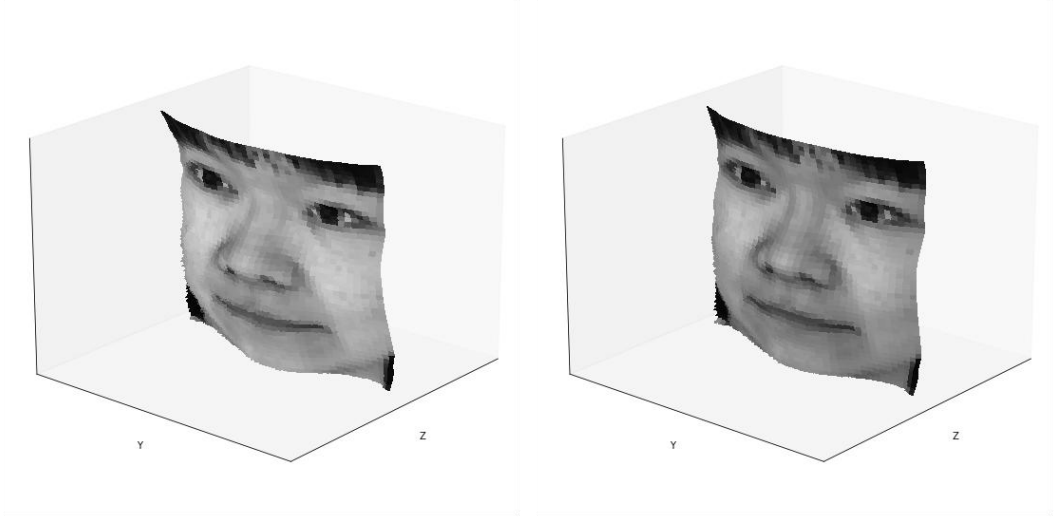
For YaleB05, removing these images to make a new subset:

"yaleB05_P00A-095E+00.pgm",

"yaleB05_P00A-110E+15.pgm",

"yaleB05_P00A+110E-20.pgm",

"yaleB05_P00A+000E+90.pgm",
"yaleB05_P00A+110E+40.pgm",
"yaleB05_P00A-130E+20.pgm",
"yaleB05_P00A-110E+65.pgm",
"yaleB05_P00A+120E+00.pgm",
"yaleB05_P00A-110E+40.pgm",
"yaleB05_P00A-120E+00.pgm",
"yaleB05_P00A+110E+65.pgm",
"yaleB05_P00A+130E+20.pgm",
"yaleB05_P00A+110E+15.pgm",
"yaleB05_P00A-110E-20.pgm",
The following left image is the height map using filtered subset and the right image is obtained without filtering.



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

As we can see, the artifacts around the nose shadow are alleviated. It shows that filtering out unmatched images could improve the shape estimation performance.

Part-4 : Bonus

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

1. Test current method on Phantom 3D stereo dataset.

We employed the DiLiGenT dataset proposed by Shi et al. at 2017 to test the method. The dataset provides ground-truth surface normal data, which could be used to quantitatively estimate the performance of the photometric stereo method.
(Paper link: http://alumni.media.mit.edu/~shiboxin/files/Shi_TPAMI19.pdf
Database link: https://sites.google.com/site/photometricstereodata/single)

We use the cat phantom with 96 images simulated in 96 different light sources. Since its a phantom data, we just use pure black as its ambient background image. The square error of the

predicted normal and GT normal is 6921.34, which seems to be not a trivial error for image with size 310×300. Perhaps due to the object is still non-Lambertian object as the database describes.
The obtained albedo image looks like as following image:

Height map is not very clear, indicating the potential problem about the setting of the database or implementation.