Name : Houhao Liang

Part-1 : Estimate the albedo and surface normals

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

(Subject B01):  (Subject B02): 

(Subject B05):  (Subject B07): 

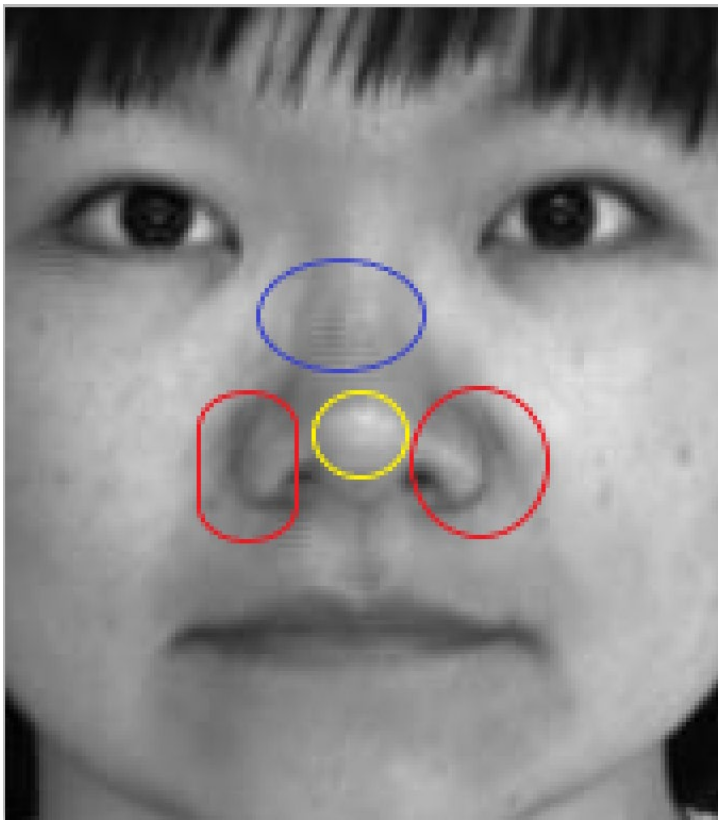
2) What implementation choices did you make? How did it affect the quality and speed of your solution?

For calculating the albedo and surface normal, we follow the steps as below.

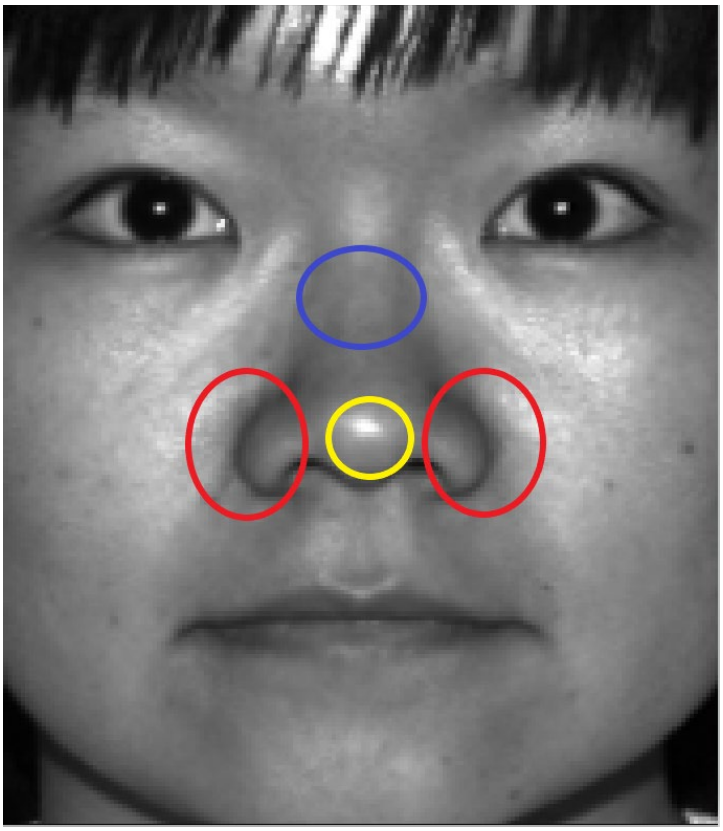
1 Preprocess images: subtract the ambient image, set negative values to zero, normalize the data into (0,1)
2 Reshape the image, (h x w x N) -> (N x h x w) -> (N x (h*w))
3 Use numpy function to compute the lease squares solution given the light source directions, pixels, and the assumption of Lambert model.
4 Use numpy function to compute the magnitude of the least square solution which is the norm value
5 Reshape the computed magnitude into (h x w), which is the albedo image
6 Least square solution is divided by albedo image, which is temp surface normal.
7 Switch the location between pixels and xyz. Reshape the temp surface normal into (h x w x 3), which is surface normal.

I use numpy function and stack all unknown g vectors for every pixel into a (3 x all pixel) matrix, avoiding use loop and speed up the calculating process. For example, use np.linalg.lstsq to calculate least square solutions, and np.linalg.norm to calculate magnitude.

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



(Albedo image)



(Original image)

1 Red circle: Albedo image should be a constant color, but we can see some dark color in the albedo image and the structure of the nose.
2 Yellow circle: The shiny area is blended in the albedo image
3 Blue circle: The pixel in the albedo image is different from the original image.

4) Display the surface normal estimation images below:
(Subject B01):

(Normal - x) (Normal – y) (Normal – z)

(Subject B02):

(Normal - x) (Normal – y) (Normal – z)

(Subject B05):

(Normal - x) (Normal – y) (Normal – z)

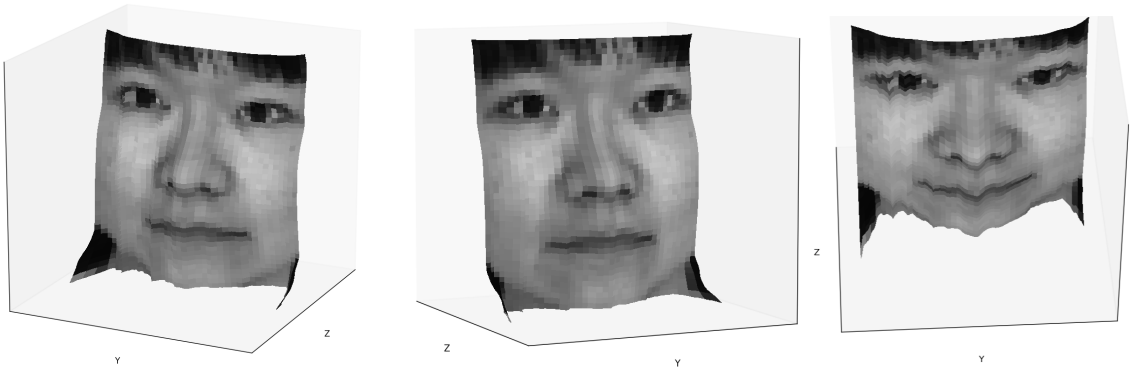
(Subject B07):

(Normal - x) (Normal – y) (Normal – z)

Part-2 : Compute Height Map

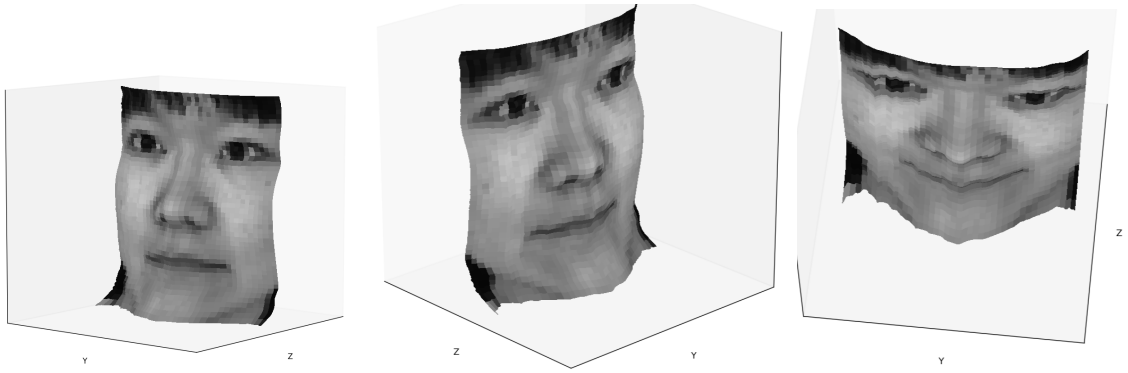
5) For every subject, display the surface height map by integration and display. Select one subject, list height map images computed using different integration method and from different views; for other subjects, only from different views, using the method that you think performs best. When inserting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

Select Subject B05:
For row method:

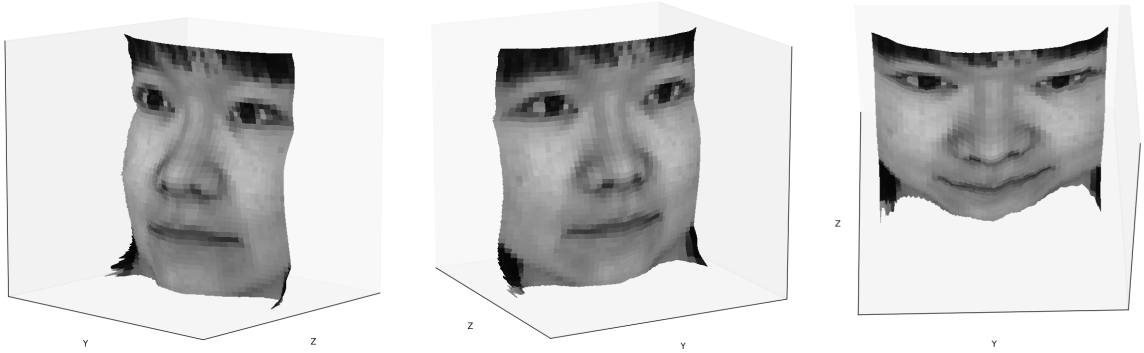


For column method:

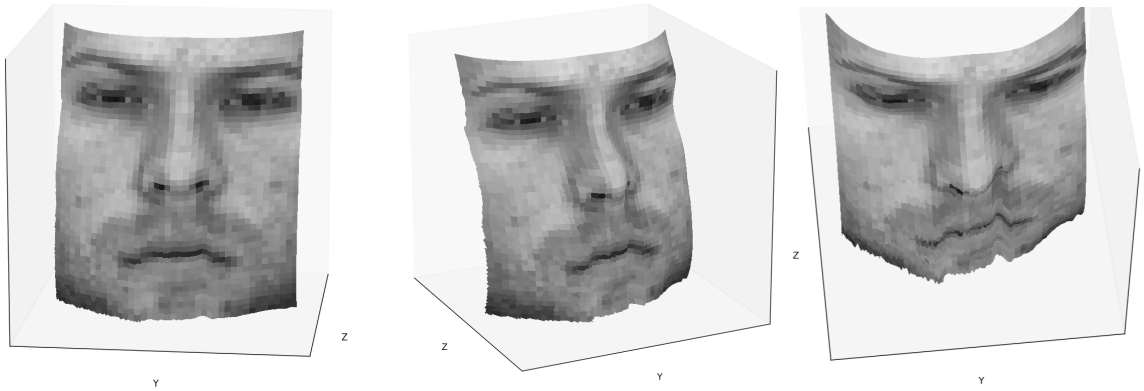
For average method:



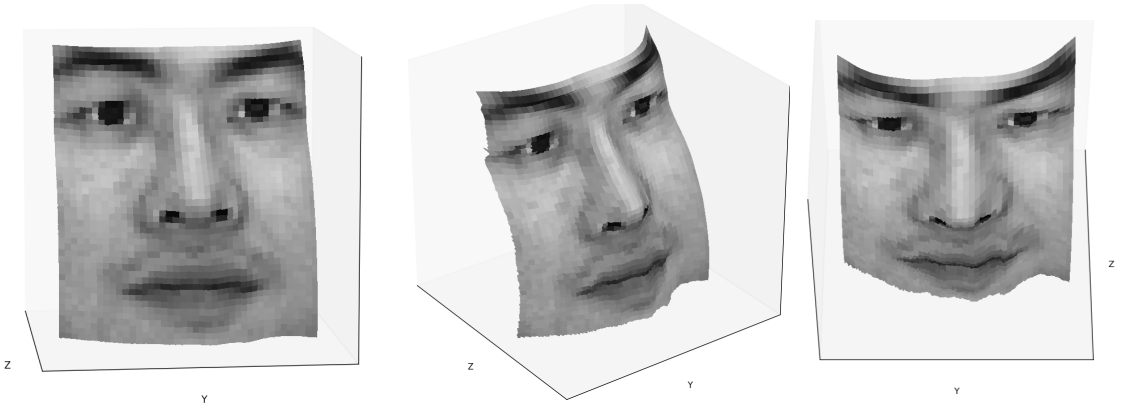
For random method:



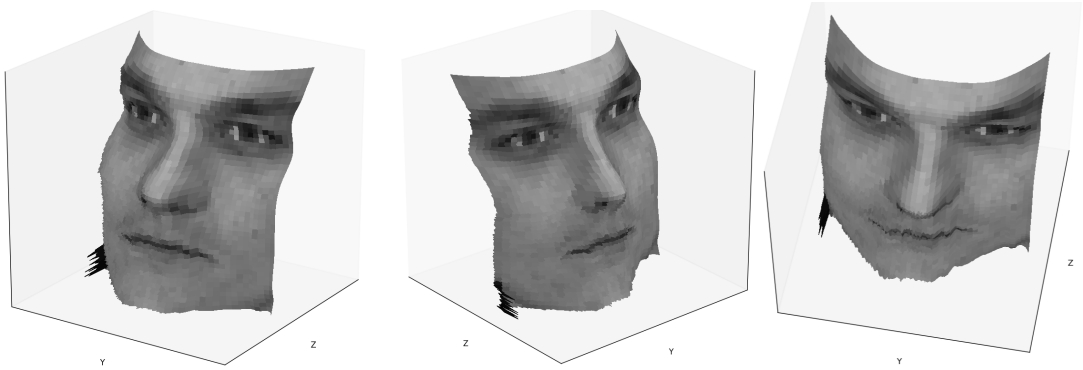
For the rest of image, use random method which is the best.
Subject B01:



Subject B02:



Subject B07:



6) Which integration method produces the best result and why?
   The random method is better.
   For row and column method, you integrate from one path and one direction. For average, you take the average the column and row method, and the error is average. For random method, your pixel gets more integration paths and this method take integration from different equally direction. Then take average, so your error is less than the other method.

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	87.29040380000015 seconds
average	0.0012922999999318563 seconds
row	0.001140099999929589 seconds
column	0.0007708000000548054 seconds

Analysis: The execution of random method needs more time than other methods. It's reasonable since there are more paths and more integrations. Its execution time also depends on the number of random path and the step size. If you would like to get better result, you can set more paths. However, it the number of paths exceeds a limit, the execution time will be much longer and the edge will be fuzzy.

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.

   Assumptions of shape-from-shading:

   • A Lambertian object (Lambert's law, diffuse reflection)

   • A local shading model (each point on a surface receives light only from sources visible at that point)

   • A set of known light source directions

   • A set of pictures of an object, obtained in exactly the same camera/object configuration but using different sources

   • Orthographic projection

   Violation:

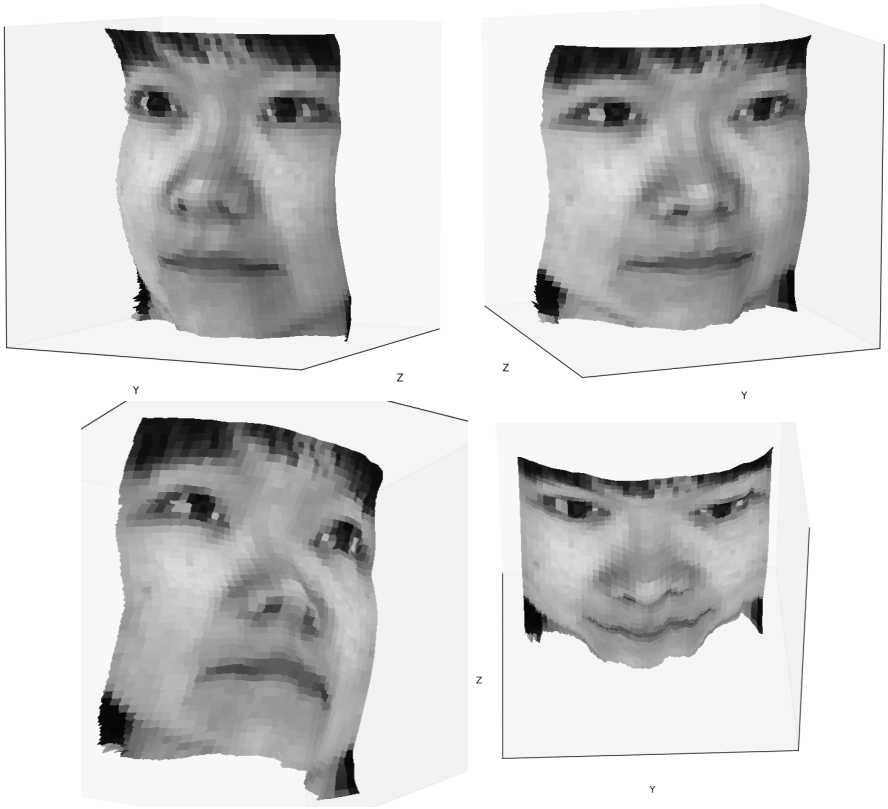
   • Shadow in the original image dataset. Consider the equation, I = k * albedo * N * S, if light is zero, the albedo should be zero.

   • The surface of Yale Face is not dull and matte, which means it violate the diffuse reflection.

   • The Image data doesn't obey the assumption of diffuse reflection. Most of them have shiny area which is due to specular reflection.

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.
After removing some images:
(Albedo Image)

(3D reconstruction):



10) Discuss whether you were able to get any improvement over a reconstruction computed from all the viewpoints.
After removing the images with lots of shadow, the reconstruction results get better. The model is more close to a Lambert model, which obeys our method assumption. But since the number of images in this dataset is not sufficient enough, we would get worse result if we remove too many images.

Part-3 : Bonus

Post any extra credit details/images/references used here.
Probably can set a threshold for shadow area.