

**Name :** Yan Zhang

**Part-1 : 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?

**1. reshape the original 3D image array (h x w x n) into 2D array (n x hw) as below:**

$$\begin{bmatrix} I_1(x_1, y_1) & \cdots & I_1(x_{hw}, y_{hw}) \\ \vdots & \ddots & \vdots \\ I_n(x_1, y_1) & \cdots & I_n(x_{hw}, y_{hw}) \end{bmatrix}$$

, where each column corresponds to the pixel value vectors of N projection images at the location of that pixel.

Code:

```
y=imarray.transpose(2,0,1).reshape(Nimages,-1)
```

**2. getting all the solutions with a single call to numpy solver:**

Code:

```
G=np.linalg.lstsq(light_dirs, y, rcond=None)[0]
```

Here, we get g vectors for every pixel into a 3 x N<sub>pix</sub> matrix

**3. restore the obtained 2D g vectors into expected 3D array (h x w x 3)**

Code:

```
G=G.reshape((3,shape[0],shape[1])).transpose(1,2,0)
```

**4. derive albedo by magnitude calculation:**

Code:

```
albedo_image=np.sqrt(np.square(G[:, :, 0]) + np.square(G[:, :, 1]) + np.square(G[:, :, 2]))
```

**5. get surface normals by normalizing G:**

Code:

```
for i in range(3):  
    G[:, :, i] = G[:, :, i] / albedo_image  
surface_normals = G
```

*By getting all the solutions with a single call to numpy solver instead of nested loops, the program would be much faster to run, since python implemented some fairly efficient algorithms for doing matrix manipulation, while the accuracy of the solution is preserved.*

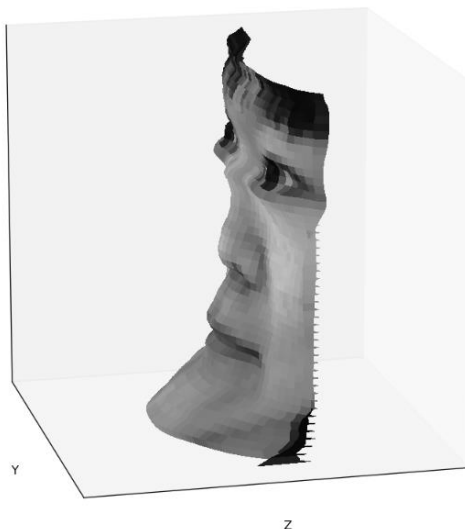
- 3) What are some artifacts and/or limitations of your implementation, and what are possible reasons for them?

*First, in principle, the albedo image should be a fairly constant color for human skin. However, the estimated image as below shows different colors for nose and face, and we can even speculate 3D structures like the nose in the albedo image.*



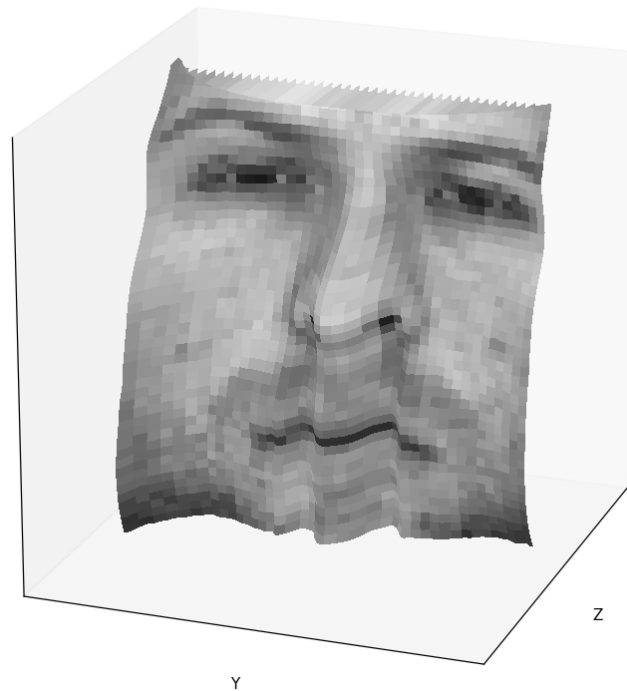
*The possible reason might lie in the fact that the assumption of photometric stereo is not strictly met in the real case. Elaborately, the set of pictures is hardly obtained in exactly the same camera/object configuration. Besides, orthographic projection is not feasibly guaranteed everywhere. All these factors lead to the errors in the albedo image estimation.*

*Another artifact I found in terms of 3D reconstructions is as below:*



*When adopting column-first integration method, based on the estimated albedos, the chin was found to be protrude significantly once it gets to the hair/shoulder. This might be due to the fact that the column-first method starts so far off from  $z=0$ , which means that it's not suitable to this dataset, and also the reason why averaging over multiple paths should be recommended.*

*Third, some skewed patterns are observed. For example, as for the 'column-first' method on yaleB01 data, the 3D reconstructed lips seem wave-shaped. This is possibly because the assumptions of the shape-from-shading method is violated in this region, in that the mustache and skin make each point on a surface receive light not only from sources visible at that point, but also influenced by shadows and interreflecion across from different parts of the face.*


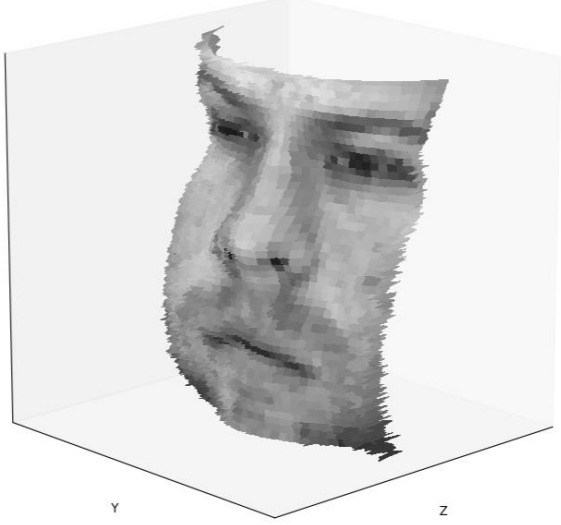
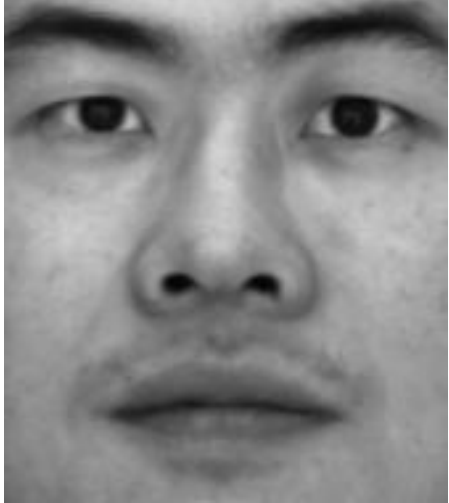
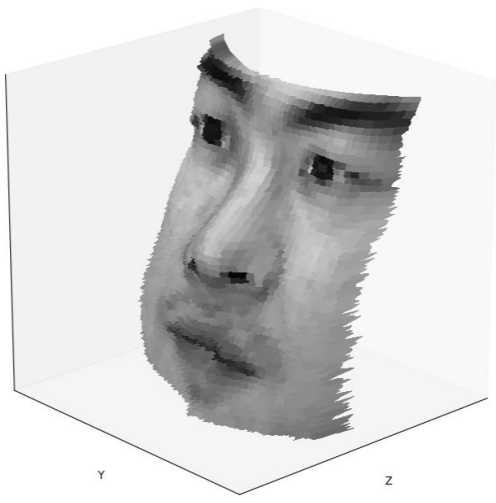


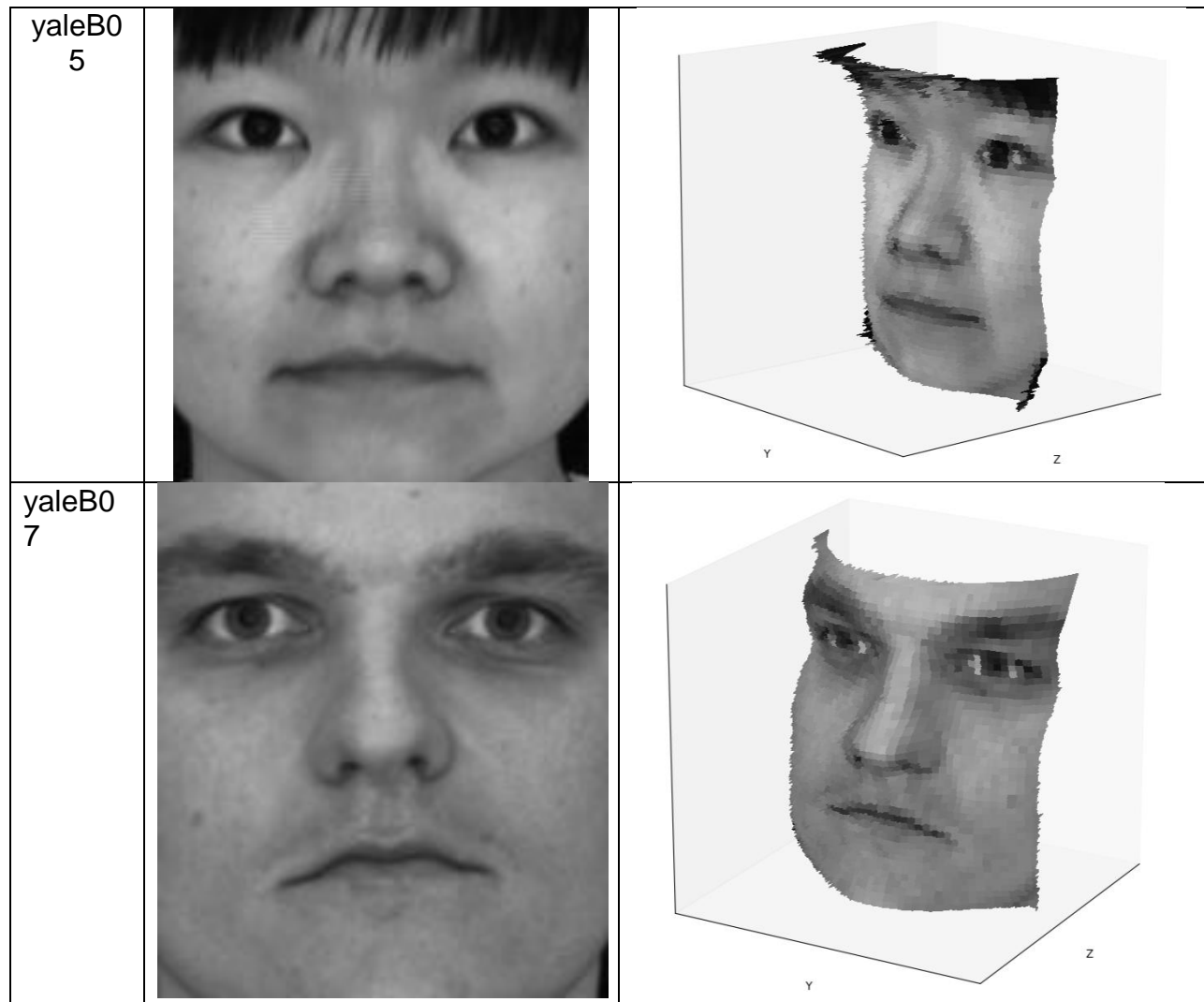
4) Display the surface normal estimation images below:



## Part-2 : Compute Height Map

- 5) For every subject, display your **estimated albedo maps** and **screenshots of height maps** (use `display_output` and `plot_surface_normals`). 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:

Subject	estimated albedo maps	screenshots of height maps
yaleB0 1		
yaleB0 2		

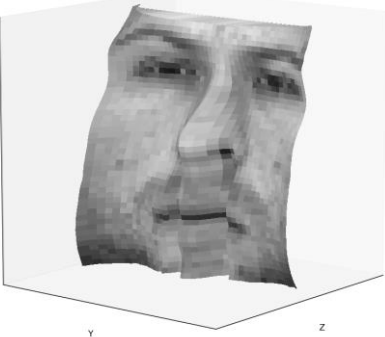
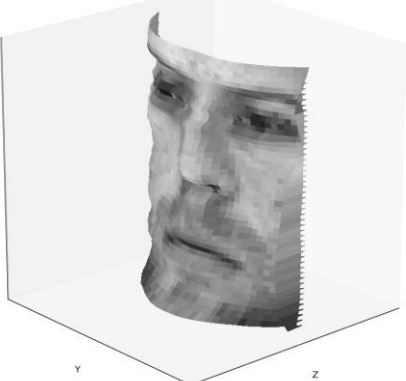
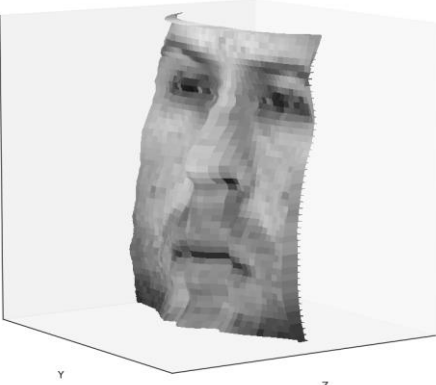
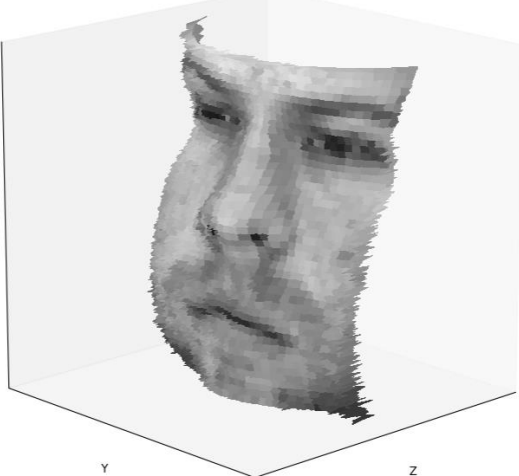


6) Which method produces the best result and why?

*For subject yaleB01, the outputs of all 4 methods are listed as below:*

*By comparison, we can see that the random integration method provides best facial profiles, i.e. best reconstruction performance.*

*The underlying mechanism lies in that the second order partial derivatives are not strictly equal in real case, so the integration result is path-dependent. Instead of single path, the random integration method takes multiple paths and average the results, thus enables the result to be more accurate.*

integration methods	screenshots of height maps
Row	 <p>A 3D visualization of a face's height map generated using the Row integration method. The face is shown in a three-quarter view, with the Y-axis pointing downwards and the Z-axis pointing to the right. The surface appears relatively smooth but has some visible blockiness or pixelation.</p>
Column	 <p>A 3D visualization of a face's height map generated using the Column integration method. The face is shown in a three-quarter view, with the Y-axis pointing downwards and the Z-axis pointing to the right. The surface shows significant vertical banding or streaking artifacts, particularly along the right side of the face.</p>
Average	 <p>A 3D visualization of a face's height map generated using the Average integration method. The face is shown in a three-quarter view, with the Y-axis pointing downwards and the Z-axis pointing to the right. The surface shows some vertical streaking, similar to the Column method, but it appears slightly less pronounced.</p>
Random	 <p>A 3D visualization of a face's height map generated using the Random integration method. The face is shown in a three-quarter view, with the Y-axis pointing downwards and the Z-axis pointing to the right. The surface is highly noisy and irregular, with many sharp, jagged edges and a very rough texture.</p>

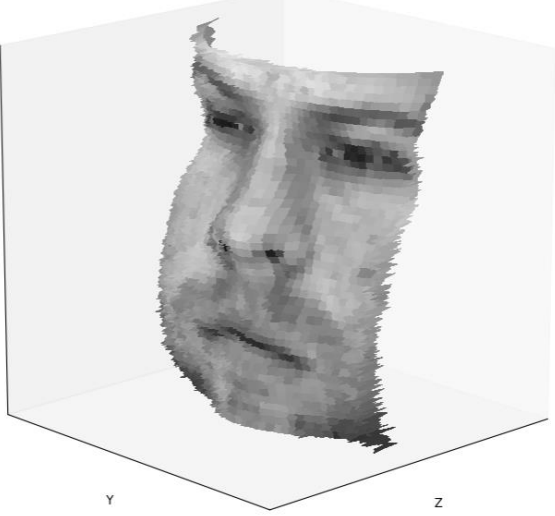

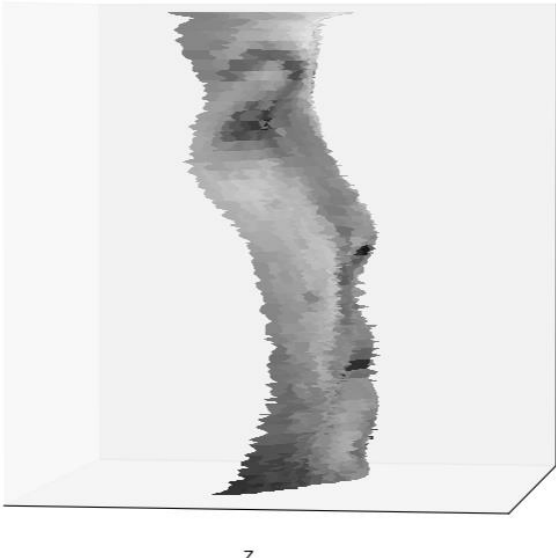
- 7) Compare the average execution time with each integration method, and analyze the cause of what you've observed:

Integration method	Execution time (s)
random	402.25882918373327
average	0.002040243503415695
row	0.00043759802747445065
column	0.00045135010100239015

*The running time for row-first and column-first integration methods are similar, while average method is a bit slower, because it deals with both row-first and column-first calculation in a single run. The random method features with much longer running time, since it iteratively generates random path (5 iterations in my case). Besides, nested loops are used instead of numpy for implementation of random method, so the code efficiency of my random method is compromised, compared with numpy, on dealing with matrix manipulation.*

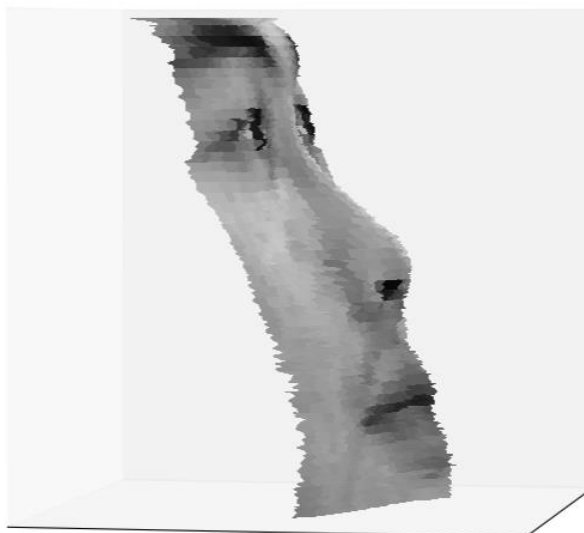
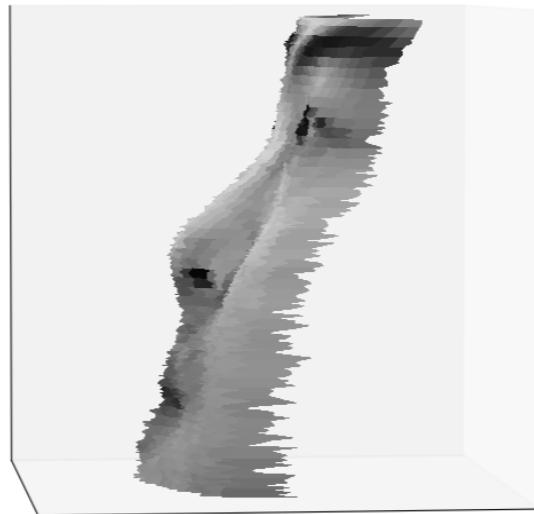
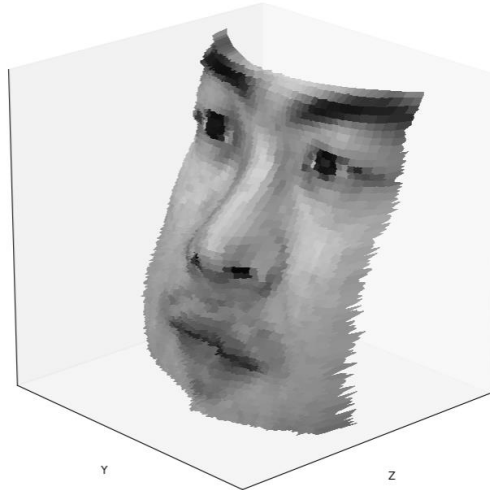
Analysis:

- 8) Post the 3D screenshots of every subjective below, select several different perspectives:

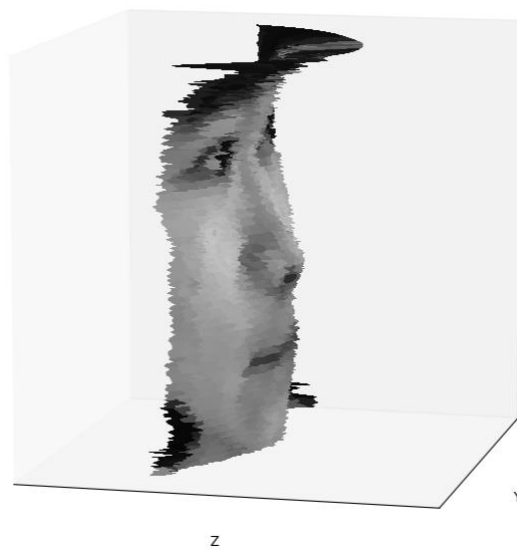
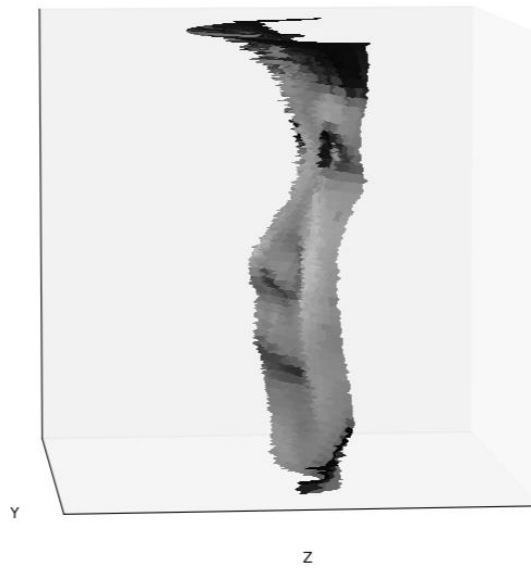
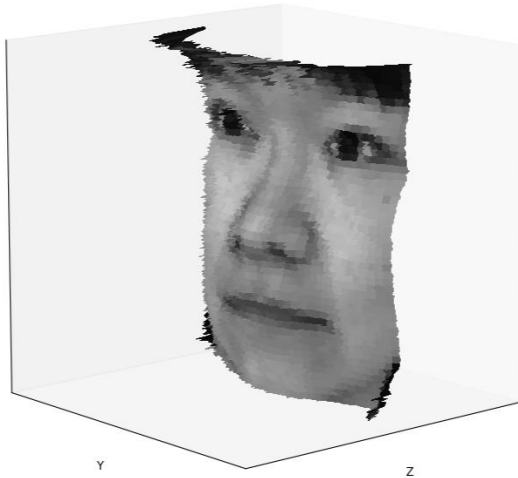
Subject	screenshots of height maps
yaleB01	  



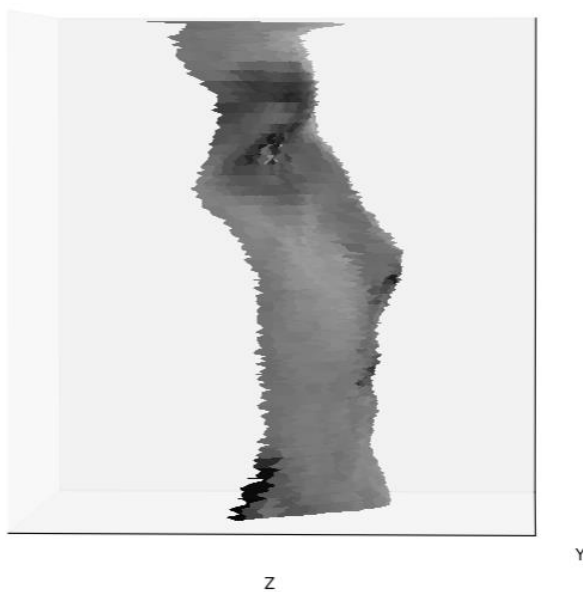
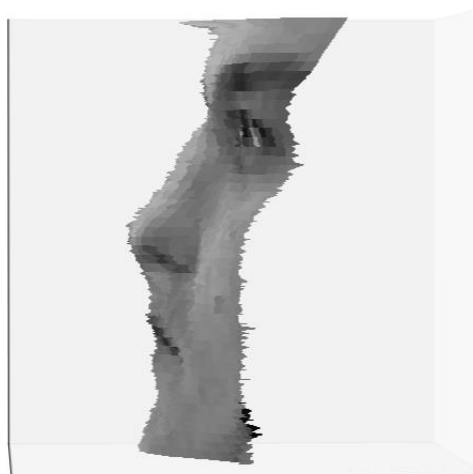
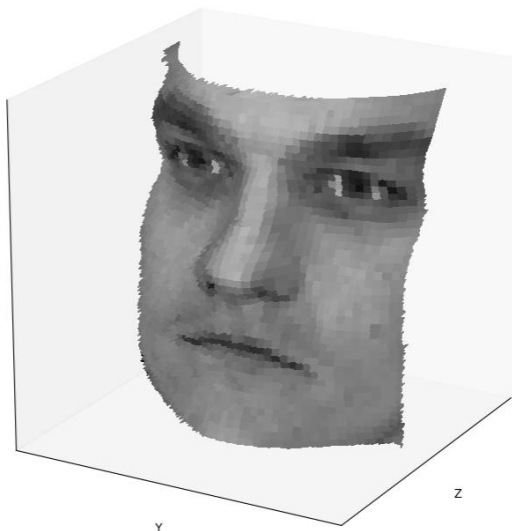
yaleB02



yaleB05



yaleB07



- 9) Analyze the results of different methods above in terms of quality of the albedos and height maps etc.

*As stated in question 6, the second order partial derivatives are not strictly equal in real case, that's the reason why we should take integrals over many different paths and average the results for robustness. The average integration method basically takes 2 paths into effect, so improvement is observed regarding height maps as in question 6, compared with row and column method where only 1 path is adopted.*

*On the other hand, my random method is implemented by looping through all pixels one by one. For a certain pixel, starting from (0,0), a random direction is picked at each step and step size is also randomized, either x or y. In this way, random path is iteratively determined and average out, so, in principle, it would generate more robust height map with smooth profiles.*

*Since the estimation of albedos is nothing related to integration method, so all methods share the same albedos images.*

### **Part-3 : Violation of the assumptions**

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

*First, not all facial areas satisfy the Lambertian object model. For example, the reflection on some smooth areas, e.g. eye region, is close to specular reflection, so the Lambertian model is invalid there. In this case, the pixel intensity no longer a matter of simple linear relation with light source directions.*

*Second, local shading model is hardly guaranteed. Because of shadows and the interreflection across from different parts of the face, each point on a surface receives light not only from sources visible at that point. Example: note the shadow by the nose*



*Third, the set of pictures is hardly obtained in exactly the same camera/object configuration, in case there's any minor facial movement between 2 frames.*

*Besides, orthographic projection is hardly possible to achieve on all facial area.*

*Accordingly, the pixel value matrix in the dataset is obtained by violating the assumptions of the shape-from-shading method. Also, the adoption of linear equation*

*system is based on the simplistic Lambertian object model, which is not the appropriate case in real world. All these features would lead to errors in the results.*

2. 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.

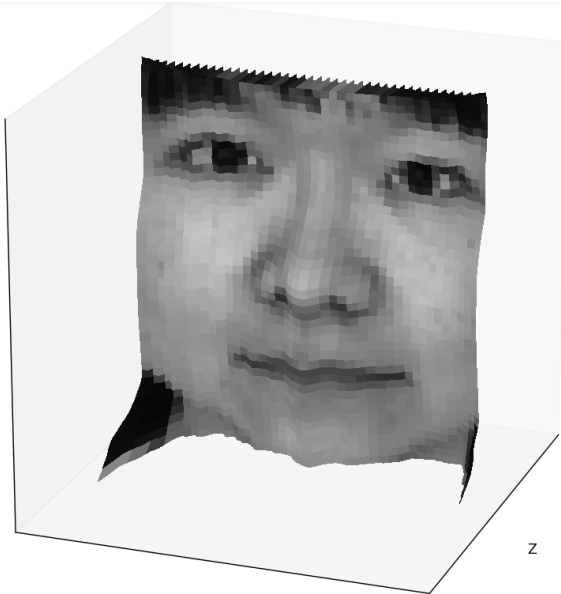
*24 out of 64 viewpoints are selected for albedo image estimation, as shown below compared with albedo image estimation from 64 viewpoints:*



*albedo image from 64 viewpoints*



*albedo image from 24 viewpoints*



*3D reconstruction from 64 viewpoints*



*3D reconstruction from 24 viewpoints*

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

*By comparison, it can be found that the selected viewpoints provide albedo image with more uniform color, e.g. the nose region, because the assumptions of the method were matched better with the selected viewpoints. But the resolution is deteriorated.*

*So, it might help to improve if we do bootstrap sampling on all viewpoints, say 30 out of 60 viewpoints in a single run, and implement the shape-from-shading method on only these selected 30 images. Such operation is iteratively implemented and finally averaging all the results from each run.*

#### **Part-4 : Bonus**

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