CS 543

Assignment 1: Shape from Shading

Chia-Hao Hsieh (chsieh17)

1. Briefly describe your implemented solution, focusing especially on the more "non-trivial" or interesting parts of the solution. What implementation choices did you make, and how did they affect the quality of the result and the speed of computation? What are some artifacts and/or limitations of your implementation, and what are possible reasons for them?

In `photometric_stereo.m`, to solve the least squares problems for all pixels at once, the formula in slide 20 of lecture 4 has to be modified as follows:

In slide 20 of Lecture4,

$$\begin{bmatrix} I_1(x,\ y) \\ I_2(x,\ y) \\ \dots \\ I_n(x,\ y) \end{bmatrix} = \begin{bmatrix} V_1^T \\ V_2^T \\ \dots \\ V_n^T \end{bmatrix} g(x,\ y)$$

$(n\ x\ 1) \quad (n\ x\ 3)\ (3\ x\ 1)$

To solve this for all p pixels,

$$\begin{bmatrix} I_{11}(x,\ y)\ I_{21}(x,\ y) \dots I_{p1}(x,\ y) \\ I_{12}(x,\ y)\ I_{22}(x,\ y) \dots I_{p2}(x,\ y) \\ \dots \quad \dots \quad \dots \quad \dots \\ I_{1n}(x,\ y)\ I_{2n}(x,\ y) \dots I_{pn}(x,\ y) \end{bmatrix} = \begin{bmatrix} V_1^T \\ V_2^T \\ \dots \\ V_n^T \end{bmatrix} [g_1(x,\ y)\ g_2(x,\ y) \dots g_p(x,\ y)]$$

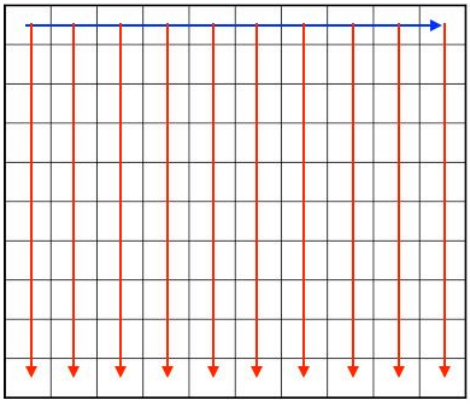
$(n\ x\ p) \qquad (n\ x\ 3) \qquad (3\ x\ p)$

… in which p = h x w is the number of all pixels. This modified formula translates to the following line of code in my program:

```
gpix = light_dirs \ ipix;
```

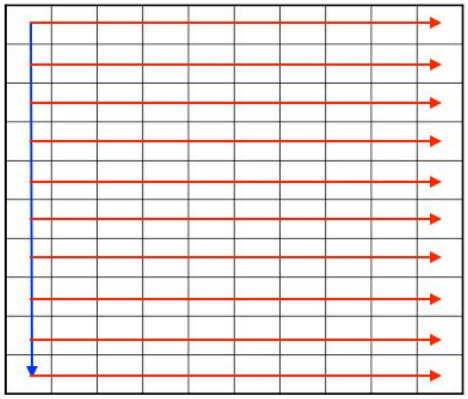
In `get_surface.m`, I use dynamic programming to update the returned height map.

For the column case, the returned height map is updated in the following order:



First go along the blue arrow, then the red ones

For the row case, the returned height map is updated in the following order:



First go along the blue arrow, then the red ones

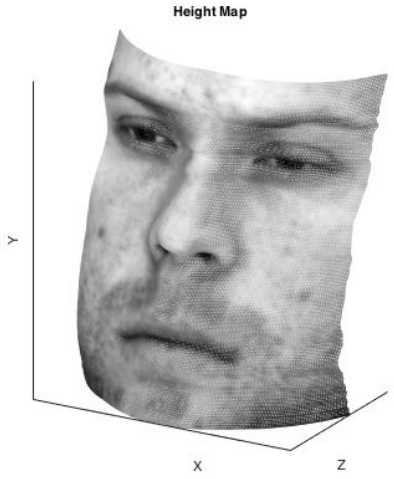
For the random case, every pixel height is computed individually by a random path. This is implemented by a recursive function, walk. Since walk is recursive, there are a lot of function calls generated for computing one single pixel height. The performance is therefore pretty bad.

2. Discuss the differences between the different integration methods you have implemented for #5 above. Specifically, you should choose one subject, display the outputs for all of a-d (be sure to choose viewpoints that make the differences especially visible), and discuss which method produces the best results and why. You should also compare the running times of the different approaches. For timing, you can use tic and toc functions. For the remaining subjects (see below), it is sufficient to simply show the output of your best method, and it is not necessary to give running times.

Case column:

Elapsed time is 0.003530 seconds.

It takes basically no time. However, there are some weird wrinkles around the subject's left eye and his nose. Those wrinkles are parallel to columns direction, which means they are there because the cumulative differences between columns become noticeable when x is large enough. This explains why the subject's right face looks smoother than the left one.

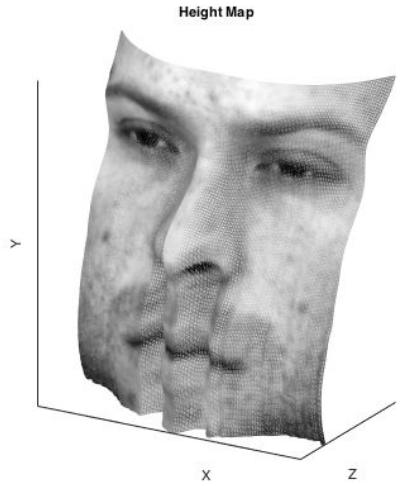


Case row:

Elapsed time is 0.003189 seconds.

It takes about the same amount of time as the column case. But the cumulative differences between rows are even bigger, which causes the subject's weird mouth. Those two gaps in his mouth appear after his nose. So it's probably because our model handles the surface normals around his nose poorly.

Also, his nose looks too sharp.

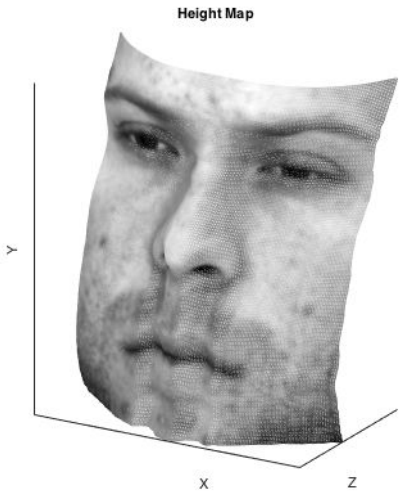


Case average:

Elapsed time is 0.007769 seconds.

This takes twice running time as the previous cases do since it simply calls both of them and takes average of their results.

The height map seems better than the column one but the subject's mouth still seems very unnatural.



Case random:

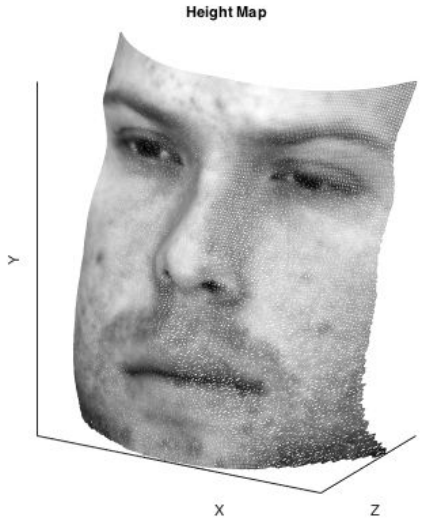
Elapsed time is 197.488877 seconds.

(I use the average of 5 random paths of all pixels.)

This takes a lot more time than other cases since recursive calls are expensive.

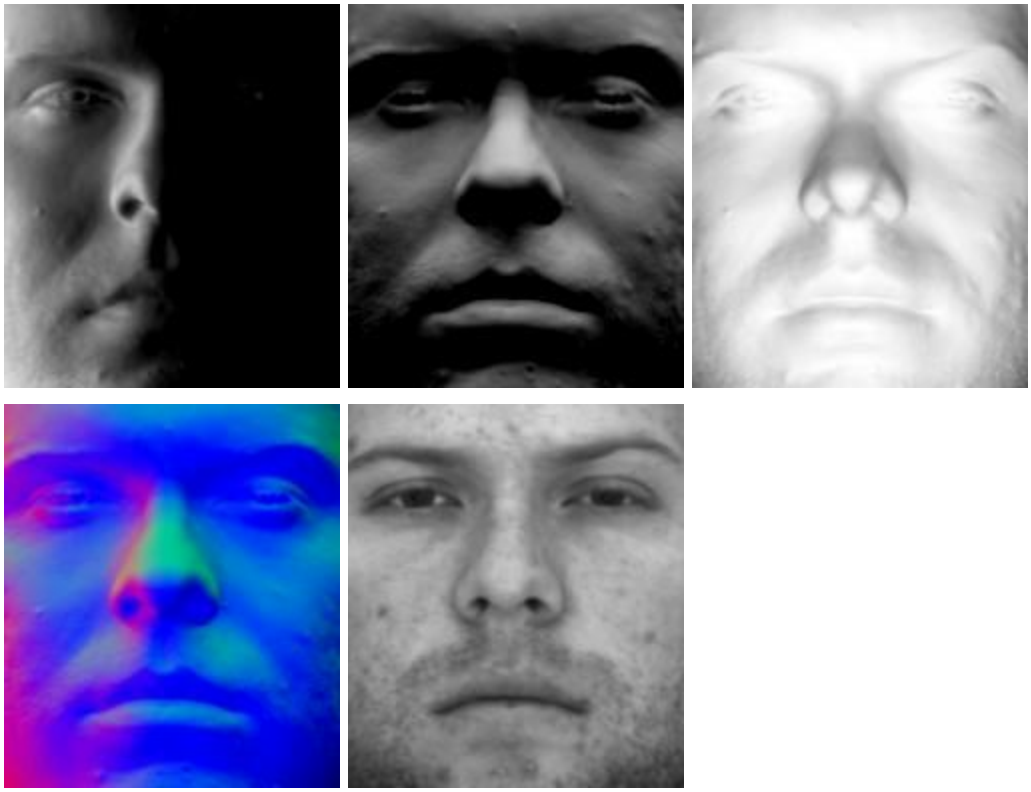
Each pixel goes through different random paths. So adjacent pixels have noncontinuous height. This explains those noisy white dots on his face.

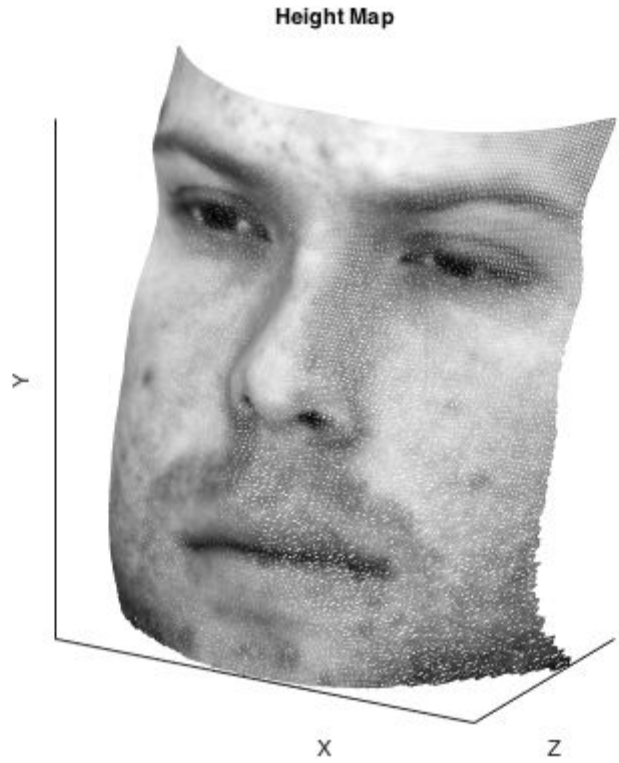
This is the best method. Because the noise here can be smoothed out easily by further processing the height map while it is hard to remove noise in the previous cases.



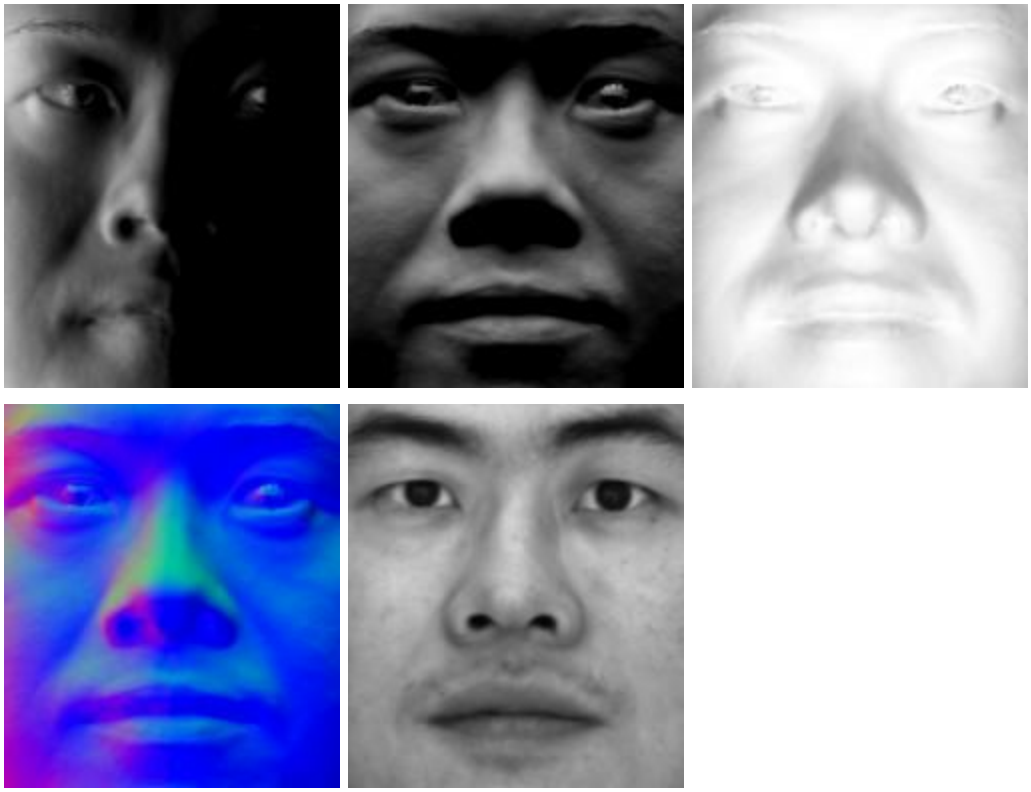
3. For every subject, display your estimated albedo maps and screenshots of height maps (use display_output and plot_surface_normals). When inserting results images into your report, you should resize/compress them appropriately to keep the file size manageable -- but make sure that the correctness and quality of your output can be clearly and easily judged. For the 3D screenshots, be sure to choose a viewpoint that makes the structure as clear as possible (and/or feel free to include screenshots from multiple viewpoints). You will not receive credit for any results you have obtained, but failed to include directly in the report PDF file.

yaleB01:

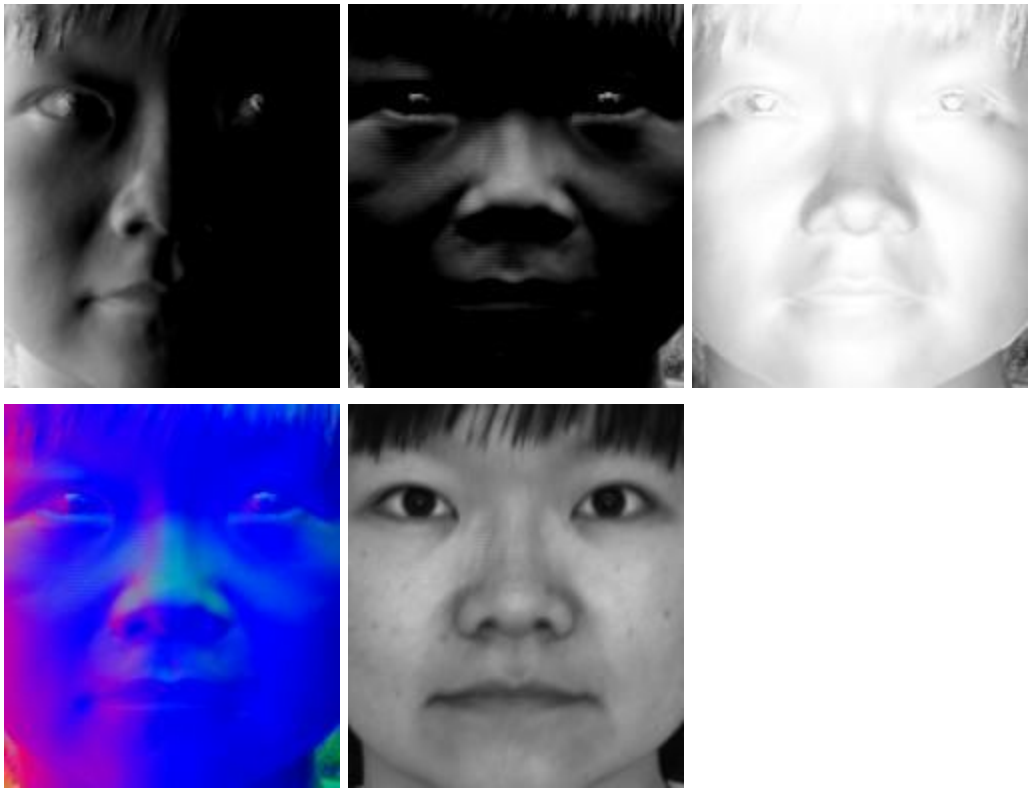




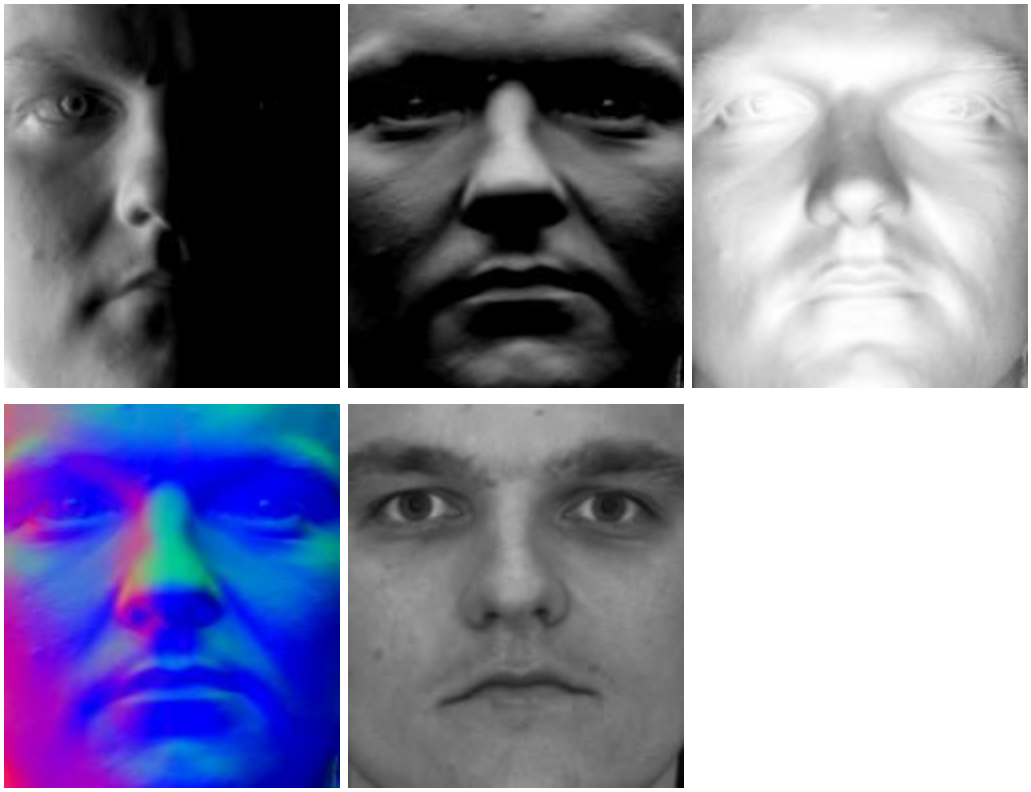
yaleB02:



yaleB05:



yaleB07:



4. Discuss how the Yale Face data violate the assumptions of the shape-from-shading method covered in the slides. What features of the data can contribute to errors in the results? Feel free to include specific input images to illustrate your points. 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 and discuss whether you were able to get any improvement over a reconstruction computed from all the viewpoints.

- Human skin, eyebrows, eyes do not exhibit Lambertian reflectance.
- Photos are taken in slightly different angle and location since it is hard for the subjects to hold their heads still the whole time.
- Light sources may not be ideal single light sources.
- Some parts of face cast shadows to other parts.