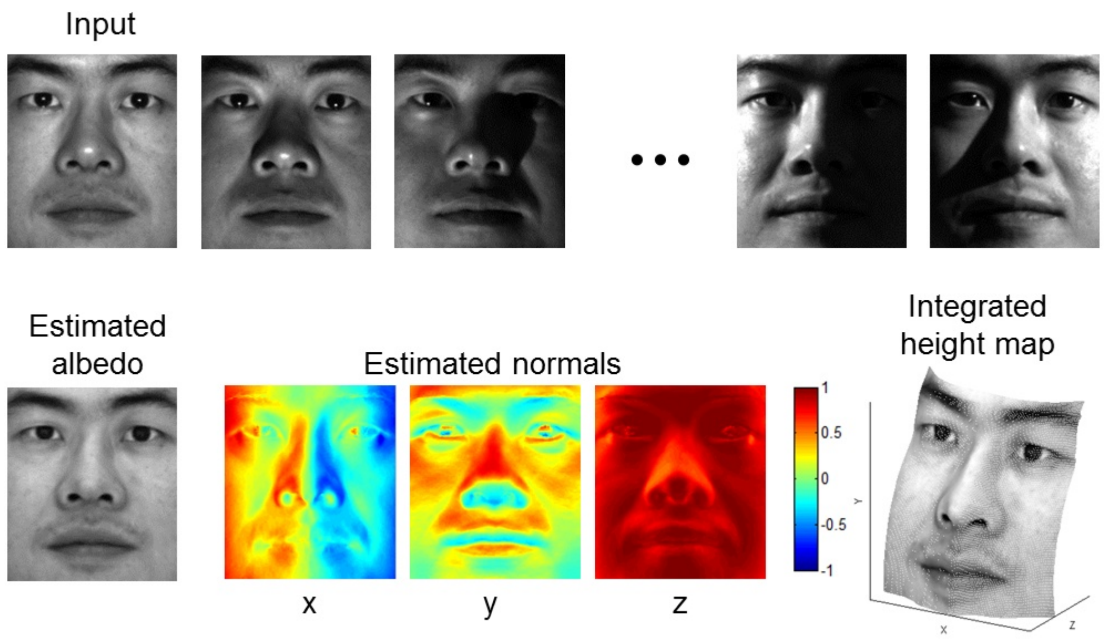
Spring 2018 CS543/ECE549

Assignment 1: Shape from Shading

Due date: Monday, February 19, 11:59:59 PM



Python Instructions -- see here for MATLAB

Adapted by Wei Han

The goal of this assignment is to implement shape from shading as described in Lecture 4 (see also Section 2.2.4 of Forsyth & Ponce 2nd edition), and to start becoming comfortable with python image and matrix processing and display functions.

1. You need the following python libraries: `numpy, matplotlib, jupyter, Pillow`.

2. Download the **data** and instead of the MATLAB code in that archive use this python notebook. The data, in the `croppedyale` directory, consists of 64 images each of four subjects from the Yale Face database. The light source directions are encoded in the file names. The provided code is in a ipython notebook. Your task will be to implement the functions `preprocess`, `photometric_stereo` and `get_surface`, as explained below. We have provided utilities to load the input data and display the output.

3. For each subject (subdirectory in croppedyale), read in the images and light source directions. This is accomplished by the function `LoadFaceImages`. `LoadFaceImages` returns the images for the 64 light source directions and an *ambient* image (i.e., image taken with all the light sources turned off).

4. Preprocess the data: subtract the ambient image from each image in the light source stack, set any negative values to zero, rescale the resulting intensities to between 0 and 1 (they are originally between 0 and 255).

   Hint: these operations can be done without using any loops whatsoever. You may want to look into the concept of array broadcasting in Numpy.

5. Estimate the albedo and surface normals. For this, you need to fill in code in `photometric_stereo`, which is a function taking as input the image stack corresponding to the different light source directions and the matrix of of the light source directions, and returning an albedo image and surface normal estimates. The latter should be stored in a three-dimensional matrix. That is, if your original image dimensions are `h x w`, the surface normal matrix should be `h x w x 3`, where the third dimension corresponds to the x-, y-, and z-components of the normals. To solve for the albedo

and the normals, you will need to set up a linear system as shown in slide 20 of Lecture 4.

Hints:

- To get the least-squares solution of a linear system, use `numpy.linalg.lstsq` function.
- If you directly implement the formulation of slide 20 of the lecture, you will have to loop over every image pixel and separately solve a linear system in each iteration. There is a way to get all the solutions at once by stacking the unknown **g** vectors for every pixel into a `3 x npix` matrix and getting all the solutions with a single call to numpy solver.
- You will most likely need to reshape your data in various ways before and after solving the linear system. Useful numpy functions for this include `reshape`, `expand_dims` and `stack`.
- Compute the surface height map by integration. The method is shown in slide 23 of Lecture 4, except that instead of continuous integration of the partial derivatives over a path, you will simply be summing their discrete values. Your code implementing the integration should go in the `get_surface` function. As stated in the slide, to get the best results, you should compute integrals over multiple paths and average the results. You should implement the following variants of integration:
    a. Integrating first the rows, then the columns. That is, your path first goes along the same row as the pixel along the top, and then goes vertically down to the pixel. It is possible to implement this without nested loops using the `cumsum` function.
    b. Integrating first along the columns, then the rows.
    c. Average of the first two options.
    d. Average of multiple random paths. For this, it is fine to use nested loops. You should determine the number of paths experimentally.

- Display the results using functions `display_output` and `plot_surface_normals` included in the notebook.

Extra Credit

On this assignment, there are not too many opportunities for "easy" extra credit. This said, here are some ideas for exploration:

- Generate synthetic input data using a 3D model and a graphics renderer and run your method on this data. Do you get better results than on the face data? How close do you get to the ground truth (i.e., the true surface shape and albedo)?
- Investigate more advanced methods for shape from shading or surface reconstruction from normal fields.
- Try to detect and/or correct misalignment problems in the initial images and see if you can improve the solution.
- Using your initial solution, try to detect areas of the original images that do not meet the assumptions of the method (shadows, specularities, etc.). Then try to recompute the solution without that data and see if you can improve the quality of the solution.

If you complete any work for extra credit, be sure to clearly mark that work in your report.

Grading Checklist

You should turn in both your **code** and a **report** discussing your solution and results. For full credit, your report should include a section for each of the following questions:

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?

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.

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.**

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.

Instructions for Submitting the Assignment

Your submission should consist of the following:

- All your python code **in a single ipython notebook, with output cleared**. The filename should be **lastname_firstname_a1.ipynb**.
- Your report **in a single PDF file** with all your results (including images) and discussion. The filename should be **lastname_firstname_a1.pdf**.

The files will be submitted through **Compass 2g**. Upload instructions:

1. Log into **https://compass2g.illinois.edu** and go to **Spring 2016 CS543/ECE549**.
2. Select **Course Content** from the left column.
3. Select **Assignment 1** from the list.
4. Upload **(1) your PDF report** and **(2) the zip file containing your code** as two attachments.
5. Hit **Submit**.

Multiple attempts will be allowed but only your last submission before the deadline will be graded. **We reserve the right to take off points for not following directions.**

Late policy: You lose 25% of the points for every day the assignment is late. If you have a compelling reason for not being able to submit the assignment on time and would like to make a special arrangement, you must send me email **at least a week before the due date** (any genuine emergency situations will be handled on an individual basis).

Academic integrity: Feel free to discuss the assignment with each other in general terms, and to search the Web for general guidance (not for complete solutions). Coding should be done individually. If you make substantial use of some code snippets or information from outside sources, be sure to acknowledge the sources in your report. At the first instance of cheating (copying from other students or unacknowledged sources on the Web), a grade of zero will be given for the assignment. At the second instance, you will automatically receive an F for the entire course.