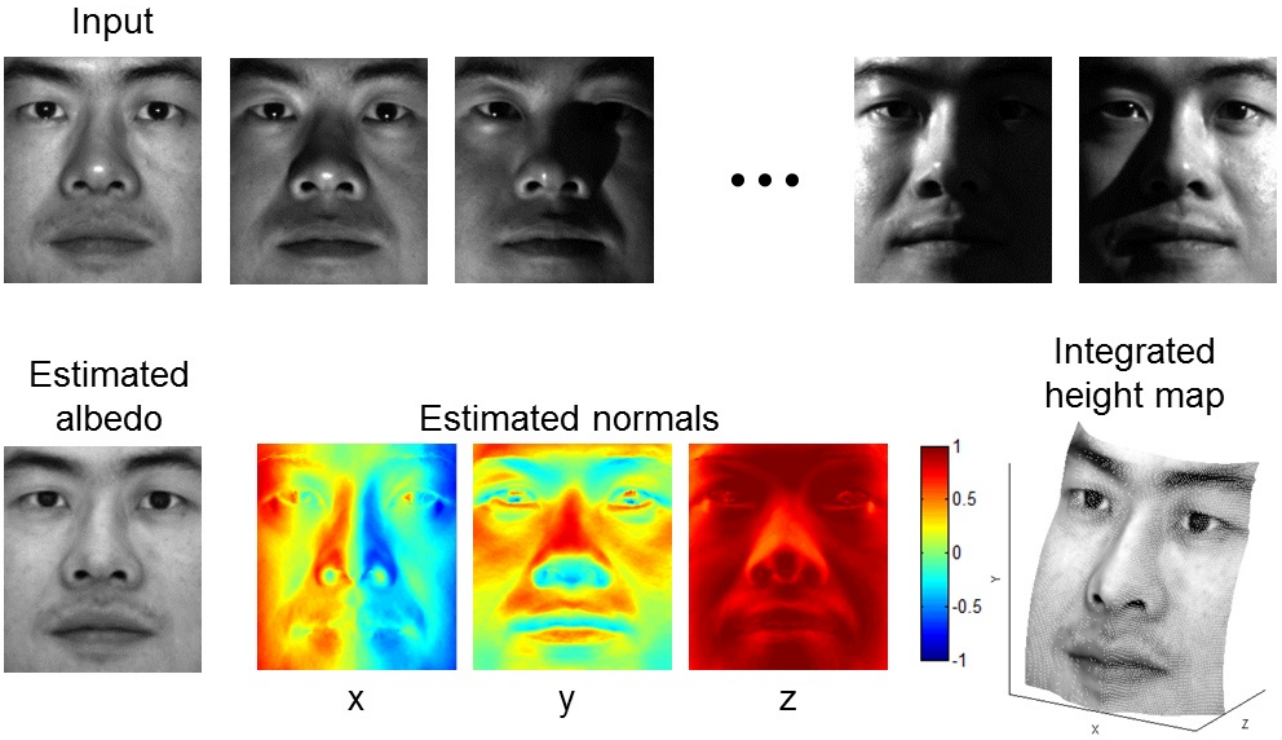
EECS 442: Computer Vision, Winter 2016

Homework 2: Photometric Stereo

Due date: February 3 by 11:59PM



Instructions

The goal of this assignment is to implement shape from shading as covered in lecture. This is also described in shape from shading section (Sec 2.2) in Forsyth and Ponce book (pdf link for this section).

1. The data 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 code consists of several .m functions. Your task will be to fill in the code in `prepareData.m`, `photometricStereo.m`, and `getSurface.m` as explained below. The remaining files are utilities to load the input data and display the output.

   When testing your code you can change parameters in `evalCode.m` to change which image set and surface method is used. You can start by setting the `subjectName='debug'` which creates images from a toy scene. You can debug your code against this before you try the faces.

2. For each subject (subdirectory in data), read in the images and light source directions. This is accomplished by the function `loadFaceImages.m` which is provided for you. `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).

3. 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 MATLAB's `bsxfun` function.

4. Estimate the albedo and surface normals. For this, you need to fill in code in `photometricStereo.m`, 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 each of the three channels corresponds to the x-, y-, or z-component of the normals. To solve for the albedo and the normals, you will need to set up a linear system as shown in the lecture slides.

   Hints:
   - To get the least-squares solution of a linear system, use MATLAB's backslash operator. That is, the solution to `Ax = b` is given by `x = A\b`.
   - If you directly implement the formulation in the slides, 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 npixels` matrix and getting all the solutions with a single application of the backslash operator. (You do not have to do it this way, it is just something to consider)
   - You will most likely need to reshape your data in various ways before and after solving the linear system. Useful MATLAB functions for this include `reshape` and `cat`.
   - You may also need to use element-wise operations. For example, for two equal-size matrices X and Y, `X .* Y` multiplies corresponding elements, and `X.^2` squares every element. As before, `bsxfun` can also be a very useful function here.

5. Compute the surface height map by integration. The method is shown in the slides, 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 `getSurface.m` file. 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. Integrate across multiple random paths, taking the average to get a final result.

6. Display the results using the included code `displayOutput` and `plotSurfaceNormals`. To make things easier, running `evalCode.m` will call everything automatically.

Extra Credit

On this assignment, there are not too many opportunities for "easy" extra credit. That 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 `displayOutput` and `plotSurfaceNormals`). 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).

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

Turn the following files into CTools:

- prepareData.m
- photometricStereo.m

- getSurface.m
- report.pdf

Also include additional code (e.g. for extra credit) and explain it in the report what each file does.

Acknowledgements

This homework is based on a similar one made by Lana Lazebnik at UIUC.