EECS 442 Computer Vision: HW1

Term: Fall 2017
Instructor: Jason J. Corso, EECS, University of Michigan
Due Date: 9/28 23:59 Eastern Time

Constraints: This assignment may be discussed with other students in the course but must be written independently. Over-the-shoulder Matlab coding is strictly prohibited. **Web/Google-searching for background material is not permitted. Everything you need to solve these equations is presented in the course notes and background materials, which have been provided already.**

Goals: Test background material and ability to implement common tools in Matlab or Python. Deepen the understanding of image as functions material.

Data: You need to download the supplemental data to complete this assignment. All paths in the assignment assume the data is off the local directory. The download has data and code; it is named `hw1.zip`.

Problem 1 (16): Fitting Curves with Linear Least Squares

We discussed the least-square formulation for a slope-intercept model of a line during lecture.

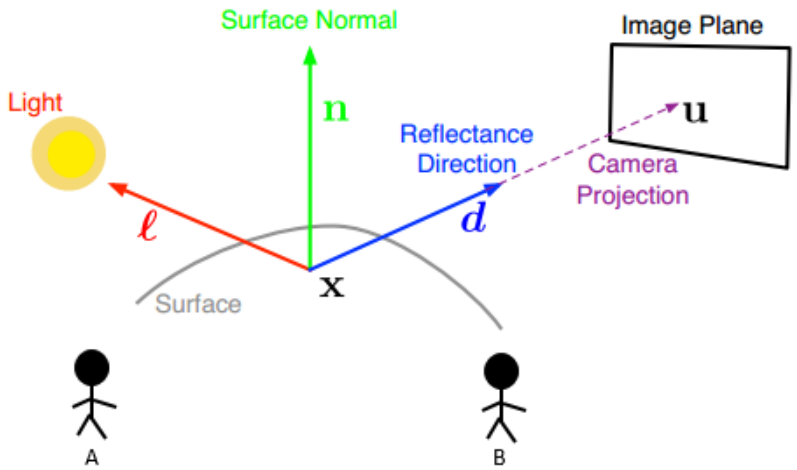
$$E(m,b) = \sum_{i=1}^{n} [y_i - mx_i - b]^2 \tag{1}$$

And we also talked about fitting a quadratic curve using LLS. Now, assume you got some sample points from some experiments, i.e. `hw1_p1.mat` in the problem1 folder.

1. (7) Say you want to fit the data to a cubic curve. Write down the least squares formulation (in a form similar to Eq.1) as well as the matrix formulation of it. Explain why this problem could be solved by using Linear Least Squares despite the fact that the curve is non-linear.
2. (6) Implement the solution of the LLS problem you wrote in (1). Report resulting coefficient values and a plot of the fit; include your code verbatim in your pdf submission. The file named `viz_curve.m` will plot the curves for you. Increase the order of curves, such as 4,5 and 6, and report their resulting plots as well.
3. (3) Does the curve with higher orders fit the data better? Explain what you observe.

Problem 2 (8): Lambertian Model

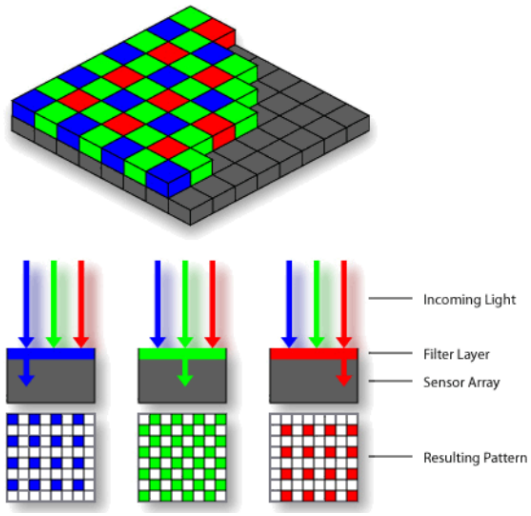
Recall the Lambertian model of reflectance: $R(\mathrm{x}) = \rho \boldsymbol{\ell}(\mathrm{x})^{\mathrm{T}} \mathbf{n}(\mathrm{x})$



1. (4) There are two people standing at location A and B. Who will observe stronger reflection at point x? Why?
2. (4) Tell one drawback of Lambertian model and name a specific object in the real world whose reflectance is not well modeled by the Lambertian model.

Problem 3 (10): Bayer Demosaicking

To digitalize a perfect color image, we overlay the Bayer pattern filter on top of the photo sensors to accept RGB separately. We could call this process mosaicking. Assume that you've got a digital image filtered by the Bayer pattern and want to demosaic it, i.e., reconstruct the original color image. One of the common ways to do this is to use bilinear interpolation. For a green pixel, two red neighbors can be interpolated to yield the red value and two blue neighbors can be interpolated to yield the blue value, and similar for red or blue pixels. The only difference is that for red or blue pixels you need to consider 4 neighbors instead of 2.



In Matlab, the function `demosaic` actually does this job, but now you need to implement this function by your own. Pick any color image you like and use `bayer_filter.m` to generate a Bayer encoded image. Implement the demosaicking algorithm in `mydemosaic.m`. Report the original image, filtered image and reconstructed image; include your `mydemosaic.m` verbatim in your pdf submission with brief explanation. Submit your code together with your report.

Problem 4 (10): Potts Model on Color Images

Recall the Potts Model that was discussed as a mechanism for modeling piecewise constant images and regularizing region boundaries. It was shown in the class that how to implement Potts model with loops. Now you need to implement the Potts model using ideas of **Spatial Range Map** operators. The images we are considering here are discrete color images with RGB three channels and for every channel values range from 0 to 255. You should complete the function `potts.m` to return the $E(\mathbf{I})$ for the given image. The provided code `potts_load.m` will load in the images, call `potts.m` and display the results for you. For all code, use $\beta = 1$.

Run and report output figures of `potts_load.m`; include your `potts.m` verbatim in your pdf submission with brief explanation. Submit your code together with your report.

Hint: the input image is in the format of uint8 and you might need to turn it to double to yield correct answer.

Rubric: 4/10 for correct, running code (that outputs the correct E for both `I1` and `I2`); 10/10 for correct, running code that uses no `for` loops.

Submission Process: NOTE: this may change before the submission. I am considering an alternative process. Submit a single pdf with your answers to these problems, include the code verbatim as text in the pdf (it is only a few lines of Matlab or Python). Include all plots and discussion in the pdf. Submit code together with the pdf if specified in the problems.

Grading and Evaluation: The credit for each problem in this set is given in parentheses at the stated question (sub-question fraction of points is also given at the sub-questions). Partial credit will be given for both paper and Matlab questions. For Matlab questions, if the code does not run, then limited or no credit will be given.