Prof. Raghuveer Parthasarathy
University of Oregon; Fall 2024

Physics 410/510 Image Analysis: Homework 2

Due date: Wednesday, October 9 by **11:59 pm,** submitted through Canvas. You'll see in "Assignments" a place to submit a **PDF**.

Readings (linked in the "[1] Digital Images; Thresholding; Noise" Canvas Page):

- (Optional) "Gonzales Woods 10p3 Thresholding partial.pdf" – on Thresholding, most of Section 10.3 of Gonzales and Woods.
- "Gonzales Woods 10p4 Adaptive Thresholding.pdf" – on Adaptive (local) thresholding, from Section 10.4 of Gonzales and Woods. Short and fairly obvious; you can instead look up other things on local thresholds if you want, e.g. https://scikit-image.org/docs/0.12.x/auto_examples/segmentation/plot_threshold_adaptive.html

Image Files: Images you'll need for the assignment are in the "Images" folder in \Files on Canvas.

1 Thresholding. (6 pts.) Use Otsu's method (`threshold_otsu` in Python's `skimage` library; `graythresh` in MATLAB) to determine the threshold values for the images "robert-mapplethrope-calla-lily-1984.png" and "istanbul_arch_museum_gray_crop.png". Plot the histogram of pixel values for each image[1] and indicate the threshold value on the histogram plot. Submit the histograms along with the binary thresholded images – paste these into your PDF. Comment on the results. Similarly, threshold an inset to the Calla Lily photo: "robert-mapplethrope-calla-lily-1984_CROP.png" – why does this look different than the thresholded whole image?

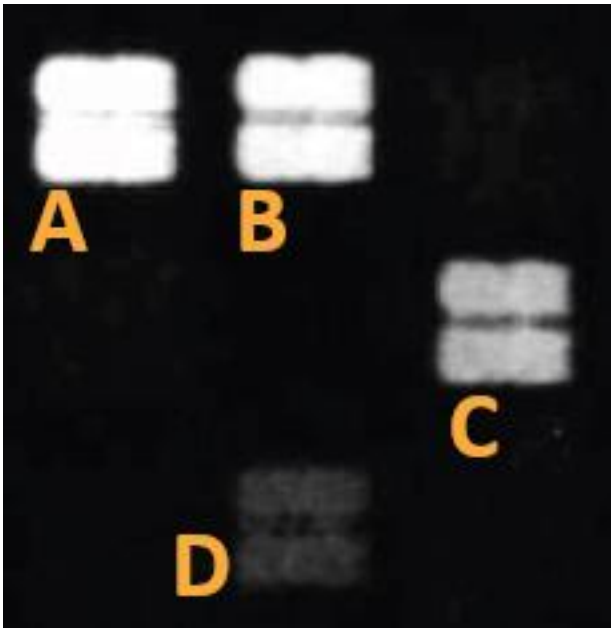
2 Thresholding, again. (6 pts.) Practice thresholding, both global and adaptive (or "local"), as described in the posted excerpt from the Gonzalez *et al.* book. Find some images; try things. You can use standard toolbox functions, such as `threshold_local` in Python's `skimage` library (use `offset=0`) or `adaptthresh` together with `imbinarize` in MATLAB or. For one image, submit the original and the results of applying a global threshold and a local -threshold, noting what block size you used. Be ready to share your examples them with the class.

[1] Be aware of "`flatten()`" and "`ravel()`" to flatten numpy arrays in Python; and (:) in MATLAB. The histogram shouldn't be a rainbow!

3 Protein density. (12 pts.) Download the image "microarray_crop.png", which shows just a few fields of peptides, roughly rectangular in shape, out of over 200,000 patterned onto a chip. (Source: this paper[2].) The different intensity levels correspond to different densities of proteins bound to these peptide patches, indicating different binding strengths. Intensity is proportional to protein density. Use ImageJ (drawing boxes and selecting "Analyze \ Measure") or figure out how to draw/analyze regions in Python or MATLAB to measure the average intensity in different regions.

(a) (8 pts.) The protein density in field C is _____ X that of field D. Fill in the blank and describe your process. Pay attention to the black regions! *Note:* you needn't calculate uncertainties.

(b) (4 pts.) What can you say about the protein density in field A relative to D?



4 Non-uniform background subtraction. (9 pts.) Images often have a non-uniform background that one wants to subtract, for example from non-uniform illumination of a sample. Consider the file "GUV24_bead_crop.png", an image of a fluorescent ellipsoidal particle. As always, we can think of the image as some intensity I that's a function of coordinates x and y. In other words: each pixel has a row and column position (y and x, respectively), and $I = I(x, y)$. Use `numpy.meshgrid` (Python) or `meshgrid` (MATLAB) to get 2D grids of x and y values of each pixel. For example, if I were a 3x3 image, x would be

```
0 1 2
0 1 2
0 1 2 (in Python)
```

[2] Hansen LB, Buus S, Schafer-Nielsen C (2013) Identification and Mapping of Linear Antibody Epitopes in Human Serum Albumin Using High-Density Peptide Arrays. PLoS ONE 8(7): e68902. https://doi.org/10.1371/journal.pone.0068902

Note that the image may be imported as a 3D array, since it's a PNG – see Homework 1. Deal with this however you want so that you use a 2D image array for the problem.

(a, 1 pt.**)** Plot the image as a surface; i.e. I is the height of a surface at each xy position. (Caution: make sure I is 2-dimensional; what can you do if it isn't?)

In Python,

```
ax = plt.axes(projection ='3d')
ax.plot_surface(x, y, I)
```

In MATLAB:

```
surf(I); shading interp
```

To make it look nicer, you may want to change the colormap from Python's default.

Submit an image of your surface plot; a screen capture is fine.

(b, 2 pts.**)** Make a graph of I vs. x for every pixel, and a graph of I vs. y. (In other words: each pixel has an x value and an I value. Make a plot of I vs. x; same for y.) Submit both graphs.

(c for 410 students, 5 pts.**)** Fit a plane to $I(x, y)$ and subtract this from the image. In more detail: the general equation for a plane $z(x, y)$ is $z(x, y) = p_2x + p_1y + p_0$; you want to find the p_2 , p_1, and p_0 that best fit the data. Remind yourself (or look up, or ask) how to calculate linear regression coefficients using linear algebra. Subtract the best-fit plane from the image, I, and display the resulting image. Submit the image, the values of the p's, **and** your code.

If you are not able to do this: For 3 points fit a line to either I vs. x or I vs. y, whichever seems more appropriate based on (b). You can use numpy's or MATLAB's `polyfit` to fit a line. In other words, you're determining the p_1 and p_0 for the equation $z = p_1x + p_0$ that best-fits the data. Subtract this z from the image, I, and display the resulting image. Submit the image, the values of p_1 and p_0, **and** your code.

(c for 510 students, 5 pts.**)** Fit a quadratic function to $I(x, y)$ and subtract this from the image. In more detail: the general equation for a quadratic function of two variables $z(x, y)$ is $z(x, y) = p_5x^2 + p_4xy + p_3y^2 + p_2x + p_1y + p_0$; you want to find the p's that best fit the data. Remind yourself (or look up, or ask) how to calculate linear regression coefficients using linear algebra. Subtract the best-fit quadratic from the image, I, and display the resulting image. Submit the image, the values of the p's, **and** your code.

(d, 2 pts.**)** What are some of the limitations of the above method for subtracting a non-uniform background? Describe images for which it would not work well. Think of at least one improvement that could be made and describe the procedure. You don't have to code the improved method, though you can if you want to.

5 Comments (2 pts.) Roughly how long did this assignment, not including the readings, take? Which parts took the most time?