Lab 7

Learning Objectives:

I. **How can I manipulate images in Python?** *How do I interact with images in Python and then display this data?* This lab will focus on using SciKit Image to be able to mathematically manipulate data.

II. **Let me Google that for you.** *How do I use function 'X'?* This lab will require use of many different functions from libraries. One of the most useful skills a programmer can have is the ability to read documentation and understand how to use certain codes. Thus for this lab we want you to look up documentation on functions and figure out how to use them to help you get the results you want.

While you work is about learning to read documentation and apply it to your code.
Part 1 is a warm-up for you to play around with some basic SciKit tools.
Parts 2 helps you practice with more complex functionality in SciKit, especially as used for data analysis.

While You Work: Google Documentation

It's hard to remember every little detail of a function, so it's useful to be able to look them up. When you need to learn or relearn how to use it up, try Google first. If you figure out that certain search terms work better than others, take note of that for future reference.

Part 0: Get the Lab from Github

Today's lab has starter code. Click on the link https://classroom.github.com/a/GFXj365-; the lab is then located at https://github.com/physics91si/spring19-lab07-*username*.

Part 1: Extracting RGB channels

This part is for you to start playing with SciKit Image. Open file lab07.ipynb and look at Part 1. Notice that we already included the relevant import statements so that you can immediately begin to use functions from the NumPy and SciKit libraries.

SciKit Image comes with a small set of images that you can play around with. For this part, we will use our own image. Load the image starry.jpg using io.imread(), and display it. Notice the shape of the image array. The last dimension has a value of 3, corresponding to each to the three RGB channels.

Make 3 copies of the image array using np.copy(). Use index slicing to convert the copies to

color arrays. For example, for the red array, you want to set all pixel values associated with green and blue to 0 (i.e. black). Using the subplot functionality of matplotlib, display the three RGB separated images in one plot. Remember to label the subplots with the color name.

Part 2: Image Analysis

Look at part 2 of lab07.ipynb. Again we have provided the required import statements. Your task here is to detect the stellar objects present in the provided image of the Hubble Deep Field. To do so, carry out the following steps:

1. Convert the given image to grayscale using the rgb2gray function. Display the original and the grayscale side-by-side using subplots.
2. You will observe that the grayscale image isn't actually black and white. Due to technical reasons, the RGB values are weighted somewhat differently than what we would expect. To display the grayscale image in black and white use the cmap option of imshow().
3. We will consider two Blob Detection methods - Laplacian of Gaussian (LoG) and Determinant of Hessian (DoH) provided as blob_dog and blob_doh respectively. Look up how these functions work, i.e. the input parameters and the output. Apply them to the grayscale image to obtain the locations and approximate radii of the blobs on the Hubble Deep Field.
4. Finally, plot the original image with the blobs highlighted. To do so, create a patch using the plt.Circle method. Add this patch to the image on the axis using the add_patch method. You will have to do this for each blob. *Hint: Use a for loop.*
5. Do step 4 for both the LoG and the DoH algorithms and display the detected blobs side by side.

Part 3: Submit Your Lab

In case you forgot -- save your work in the Jupyter notebook, then shut it down and enter the commands (from Terminal/Anaconda Prompt/whichever command line you're using):

1. git commit -m "<insert commit message here>"
2. git push