PHY331 2019 Tutorial Exercise 5

October 21, 2019

HAND IN: A hard copy of the answers to questions in 2 (in bold). Due Oct 30th in tutorial.

Provided for this week are:

1. `img_example.py`: this is just an example code that shows you how to read images of different file format.

2. `cluster.py`: this code contains the Hoshen-Koppelman functions, plus a short code which shows you how to call the HK routine. You don't need to touch any of these functions, and you don't need to dive into the inner-workings of the code unless you want to. When you write the analysis for the thermal motion experiment in the future, you might want to copy and paste the functions `uf_makeset`, `uf_union`, `uf_find` and `hoshen_koppelman` into your own code.

3. Images: in the folder `test_dataset` you are given some images taken previously for the thermal motion lab (.jpg format), as well as some online images for cytoskeletal transport (.tif format). There is also a .png file that you can try loading with the code.

1. Pixel Threshold Choice: You can use `img_example.py` to try and play around with the output, by changing the threshold intensity to see what gives you the best output images. You're going to find that a single threshold value does not work so well for every frame of the .tif images provided in the `virus_transport` folder, but for the thermal motion data, the background noise is very low and the focal plane is very stable so a single value works well.

2. Running the `cluster.py` code gives you: 1) thresholded images, 2) labeled images and 3) the labeled images in text files, meaning just a matrix of 0's and 1, 2, ... etc. corresponding to the labeled clusters. The conversion from pixel to distance is 0.1155μm per pixel. Write your own code to compute and output the centre of mass of each cluster, as well as the radius of each cluster (see radius of gyration in lecture PDF). **Submit a screenshot of your output for Image1.jpg, located in the /test_dataset/thermal_motion folder. Units should be μm.**

(HINTS: numpy.unique() will return all unique labels in an image. numpy.where() will tell you where they are located, but as we explored in exercise 1 be careful with the format of the output.)

One of the problems you might notice is that, by inspection, some clusters are only partially in the frame. Also some clusters are very small (notice that to our eyes there are only really 14 clusters in Image1.jpg, while your code will say there are 19 clusters). These issues will lead to an error in estimating the size of the particles, and their number. **What criteria can one use to exclude these measurement artifacts (Hint: think about mean and standard deviation, as well as cluster size)?** You will need to use these ideas to clean up the data in your lab writeup.