April 25, 2020 [Saturday 2 AM]

This is the first entry in my progress report (I should have started this since the beginning).

In the first meeting, I screenshot a colorized version of an image of cells (demo_img/png.png) and segmented it / picked out the cells by using only the blue channel, because I noticed the cells were tainted heavily blue. I observed that the foci were tainted red, which is how I decide to late pick them out. I used random walker segmentation for this.

In the next meeting, I don't remember what I did… But I do remember I need to use tifffile in Python instead of Pillow because these are tiff files, and other libraries cannot handle them properly. Professor Hariharan also told me that tiff files, when read into NumPy arrays using tifffile, are of pages × width × height, the pages for these cell images are the three different color channels, and it doesn't matter too much what we think the corresponding color is for each page. At this point, I am still using only image processing techniques; no machine learning. Professor Hariharan told me to automate finding a threshold number instead of manually choosing one.

Fast forward to this past week. I'm using scipy version 1.1.0 because of the imsave function. I tried to use the new alternative imageio.imwrite but it's sooooo slow and just outright wrong, or maybe I'm using it wrong I don't know. I explored several options with skimage (scikit-image). Because imsave is deprecated, it will give a lot of warnings when my code is run, but it's fine! For supervised learning, I could do thresholding with prior human input (which is what I did in the very beginning), random walker method (I did that too, along with manual thresholding), and active contour method. Active contour method looks legit but is overkill for my purposes right now. For unsupervised learning, I could do thresholding without prior user input, SLIC, and Felzenszwalb. SLIC has little computational overhead but isn't helpful to this project at this point and is overkill. I decided that thresholding on the blue channel without prior user input or knowledge is best. I tried them out, and they're pretty legit:

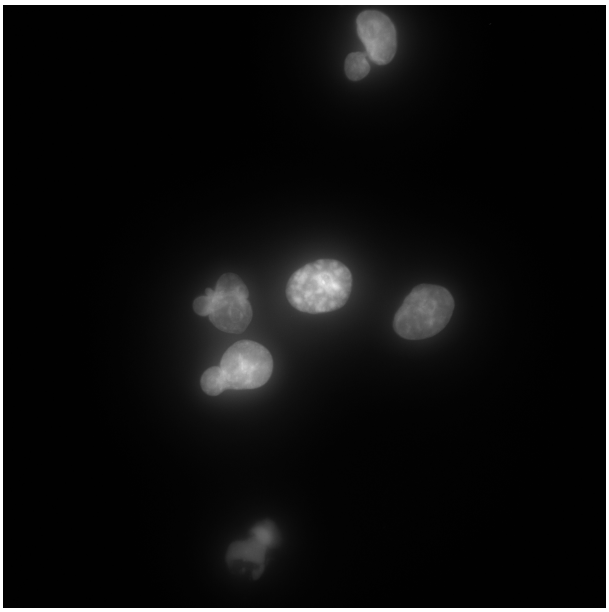


Figure 1 cells, blue channel

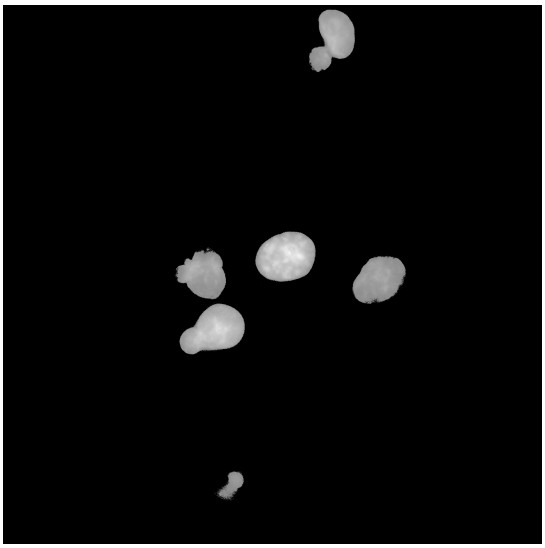


Figure 2 filters.threshold_minimum

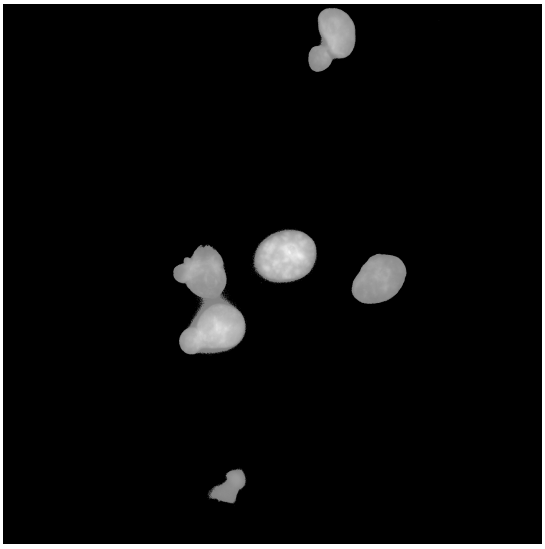


Figure 3 filters.threshold_isodata

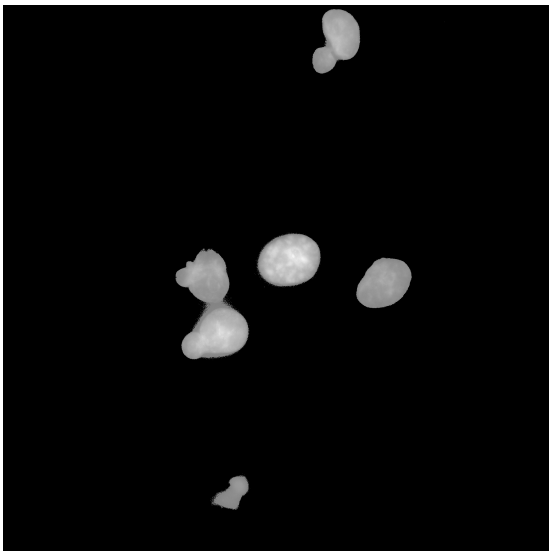


Figure 4 filters.threshold_otsu

And some not so helpful thresholding algorithms:

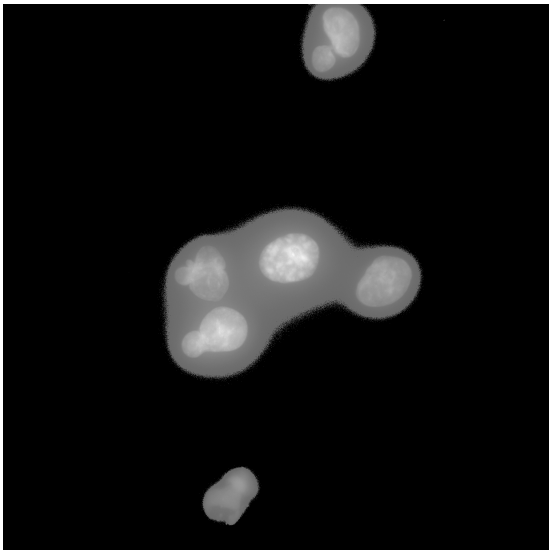


Figure 5 filters.threshold_li

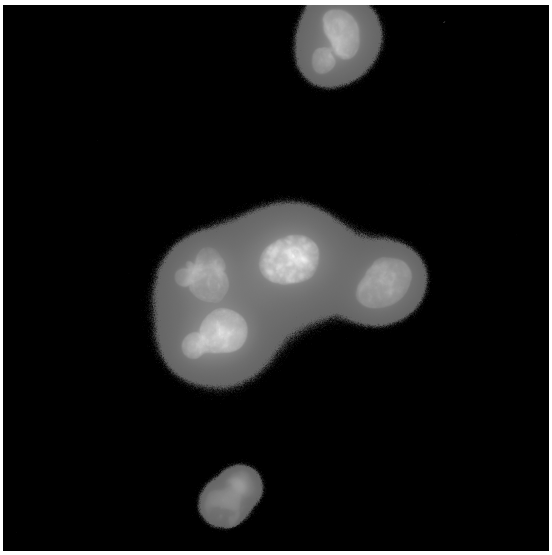


Figure 6 threshold_triangle

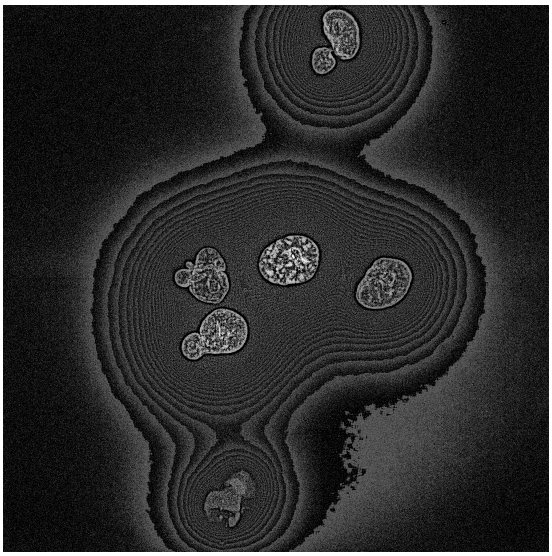


Figure 7 threshold_local; window size 15

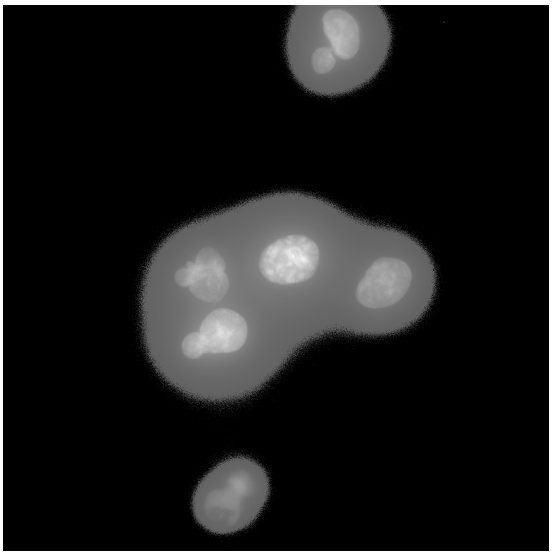
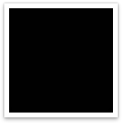


Figure 8 filters.threshold_mean

I did this for all the tiff files in `'/MCF10A/'`, `'/MDA_231/'`, `'/MDA_231_2/'`, `'/rep1_PCNT_MCF10A/'`, `'/rep1_PCNT_MDA231/'`, `'/rep2_MCF10A/'`, `'/rep2_MDA231/'`. Some of these tiff files are corrupted, apparently, Their icons look like this:

2.tif

whereas an uncorrupted one looks like this:



1.tif

(I'm on MacOS). Corrupted files:

- MCF10A/2.tif
- MCF10A/8.tif
- MDA_231_2/4.tif
- rep1_PCNT_MCF10A/4.tif
- rep1_PCNT_MDA231/ 2.tif
- rep1_PCNT_MDA231/ 3.tif
- rep2_MCF10A/2.tif
- rep2_MCF10A/5.tif
- rep2_MCF10A/8.tif
- rep2_MDA231/5.tif

I didn't try multi_otsu or yen because I knew they weren't gonna be useful for this.

The minimum threshold filter (filters.threshold_minimum) is pretty good. I ran all these filters on all tiff files in all the directories, and minimum holds up pretty well. This is unsupervised, and thus fully automated.

For next week, I will identify the (locations of) foci, and the locations of the cells, like using the center coordinates or something.