# Cell CNN Labbook

## 2017-08-08

- The data are all in .tif files. I can use PIL to import these as np arrays.
- I will need some sort of edge detection to cut the cells out of the large image.

## 2017-08-09

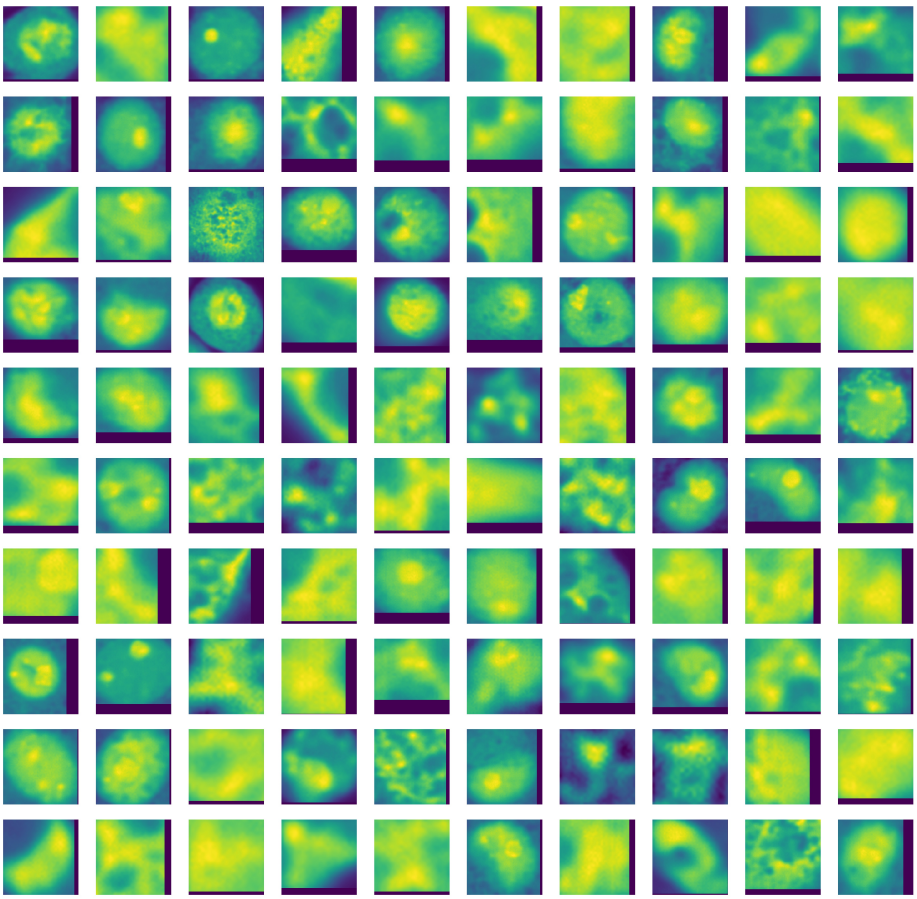
- The edge detection doesn't work too well... I think the issue is the background noise drowning out the cells.
- It works with certain fields. I am using otsu atm.
- Cy5 is the channel that is useful to me (it shows the vacuoles).

## 2017-08-11

- A naive approach of two rounds of object detection, once on the entire image, and again on each found cell, doesn't seem to be working
  too well. The object detection algorithm (otsu) doesn't pick up all the holes, doesn't discriminate between joined cells, and doesn't
  discriminate between background and holes.
- Maybe using entropy and compression would be a good measure to find holes (if entropy is larger there are more holes?).
- I will try entropy on many cells to see what happens.
- Entropy seems to work...
- I can't save the cells into h5py without putting them in a numpy array. I will have to resize them.

## 2017-08-14

- Have a HDF5 file of a load of cells now.
- The entropy route doesn't seem to be working.
- I could try finding what % of the cell is dark.
- It looks like k-means clustering has worked. I don't know how small the holes have to be, but it picks them up just fine. I also need to find a way to see how well the clustering is performing...
- Looks like the k-means clustering is filtering the more "yellow" cells:



- Choosing the larger of the two clusters results in the "blue" cells.
- The current segmentation of individual cells is a bit crude... The background is included in the vacuole number, and holes are not detected if they are directly adjacent to the background. There is probably a nice way to mask the cell so that only the interior is used in the vacuole finder.

## 2017-08-18

- Objectness algorithms are exactly what I'm looking for:
    - [(http://docs.opencv.org/3.0-beta/modules/saliency/doc/objectness_algorithms.html)](http://docs.opencv.org/3.0-beta/modules/saliency/doc/objectness_algorithms.html)
    - Installing opencv (dependency for objectness-bing) is a real pain. I need to compile from source. Also I forgot to flag opencv-contrib for compilation, so I need to compile all over again.

## 2017-08-21

- Tensorflow has a object detection api. [(https://github.com/tensorflow/models/tree/master/object_detection)](https://github.com/tensorflow/models/tree/master/object_detection)

- Doesn't look like it will work without training on cells.

## 2017-08-22

- I don't think an objectness algorithm will work for me without training. Our data is simple, and so there must be an easy way to separate each cell efficiently.
- skimage.segmentation() may be useful.
- There is a function on there to remove edge objects (probably good for my cell finder).
- I should definitely clean up the mess of files that is forming....

## 2017-08-23

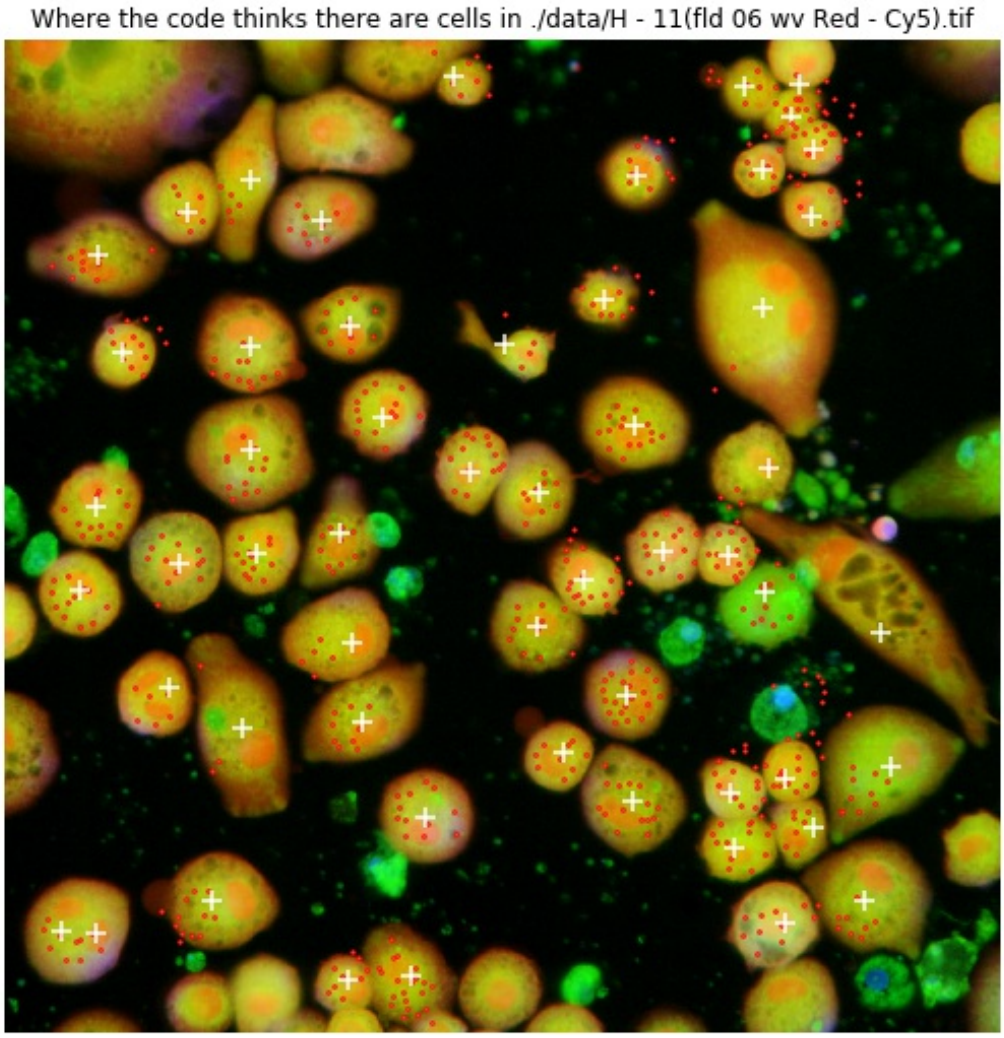
- New algorithm (in pseudocode), will write up today:

```
slide = readim(slide_red.tif)
rsSlide = rescale_intensity(slide, in_range=(10%,70%))
labelled = watershed(rsSlide)
labelled = removeBorderCells(labelled)
cells = segmentLabels(labelled)
```

- Since this uses watershed segmentation, we should be able to separate each cell...
- Once cells are separated we can try watershed segmentation on a negative of each cell and see how well/poorly it works.
    - The cell nuclei may be good places to have watershed seeds. Looks like UV has the nuclei nicely picked out for me.
- Watershed works better but still not great, probably "good enough" though.
  <img src="./figures/cells/cellsWatershed.png" alt="Example of cells found via watershed" style="width: 900px;"/
- Also I think that watershed would be a good choice for finding the vacuoles in the image.

## 2017-08-24

- Watershed has improved the cell finding algorithm, and we now find the vacuoles with threshold.niblack().
  There is probably a better way to find the vacuoles, this way is too sensitive in dark areas, and sometimes misses obvious ones,
  but it's getting there:




## 2017-08-25

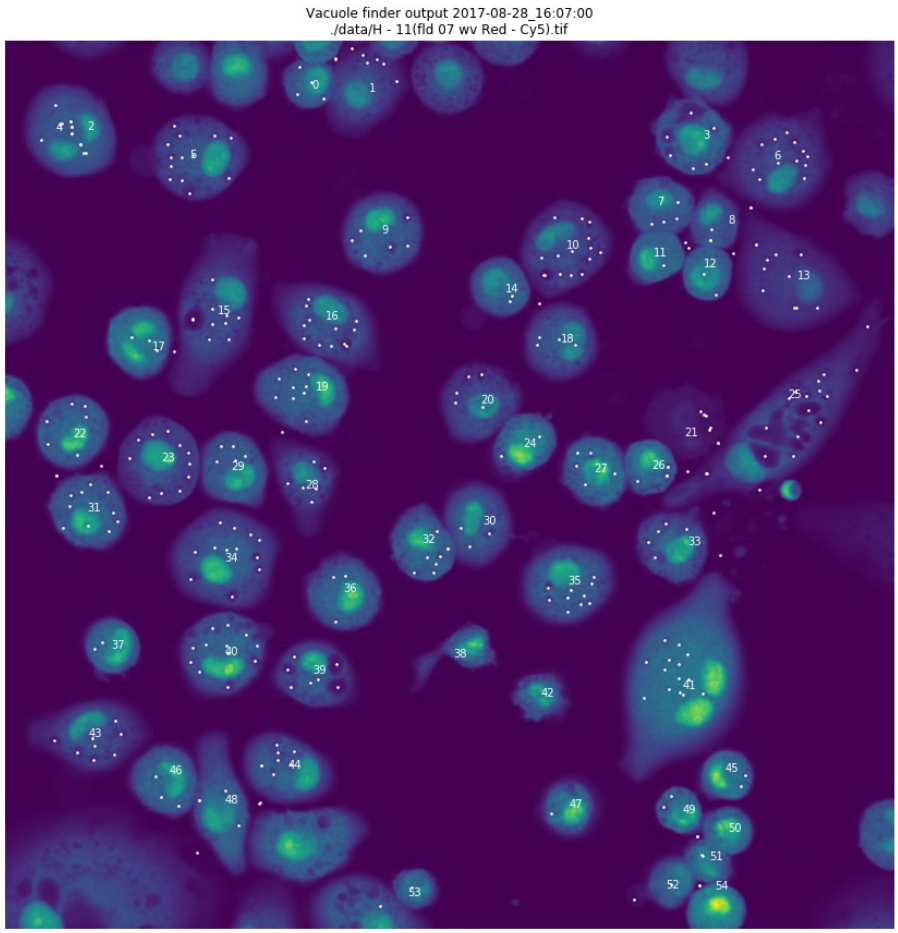
- Looks like the k-means clustering is required still, to remove noisy bits. Added it to getCells.py.
- I think the cell finding is now at an acceptable level, but if I think of something to improve it I'll give it a go.
- I should try k-means clustering on the vacuole finder... maybe it will help remove all the false positives.
- Still getting too many false negatives with my current method of finding vacuoles (threshold method). I need to think of a better way to go about this.
- It doesn't look like clustering works on the vacuoles...
- This seems to work!!!:

```
from skimage.morphology import reconstruction
im = cellImages[11]
seed = np.copy(im)
seed[1:-1, 1:-1] = im.max()
mask = im

filled = reconstruction(seed, mask, method='erosion')
fig, ax = plt.subplots(1,2)
ax[0].imshow(filled - im)
ax[1].imshow(im)
```

## 2017-08-28

- Fixed the issue with outputting found vacuoles to the big image. I forgot I scaled the cells so that they would fit into the np.array, so the
  indices were all wonky.




- The issue is that I have to use lists so that the cells can be different sizes. I will rewrite the python script to find a way around this.

## 2017-08-29