BLOB ANALYSIS

Objectives

1. Use blob analysis to measure the size of each cell (in number of pixels). Give the best estimate of the average cell size. Best estimate is the mean of the size +/- standard deviation
2. Mark out the centroid of the cell and enclose each with a bounding box.
3. Measure other shape features of the cell such as eccentricity, major axis length, or perimeter. (not finished)

Step 0: Blob Detection

- Reference: https://www.learnopencv.com/blob-detection-using-opencv-python-c/
- The image is segmented based on its grayscale values.
- Morphological Operations were used to close unwanted holes in regions of interest or ROI (closing) and to clean the image from noise via (opening).

Step 0: Blob Detection

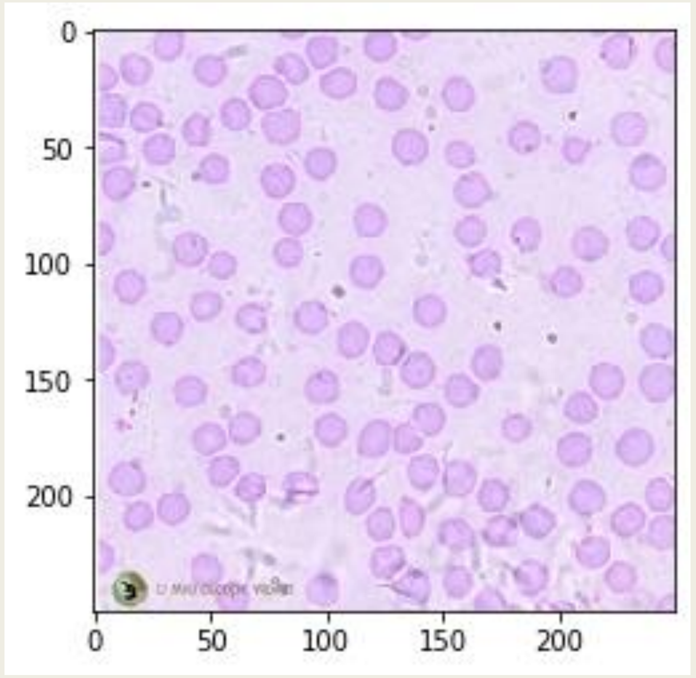


Figure 1: Original Image of Cells

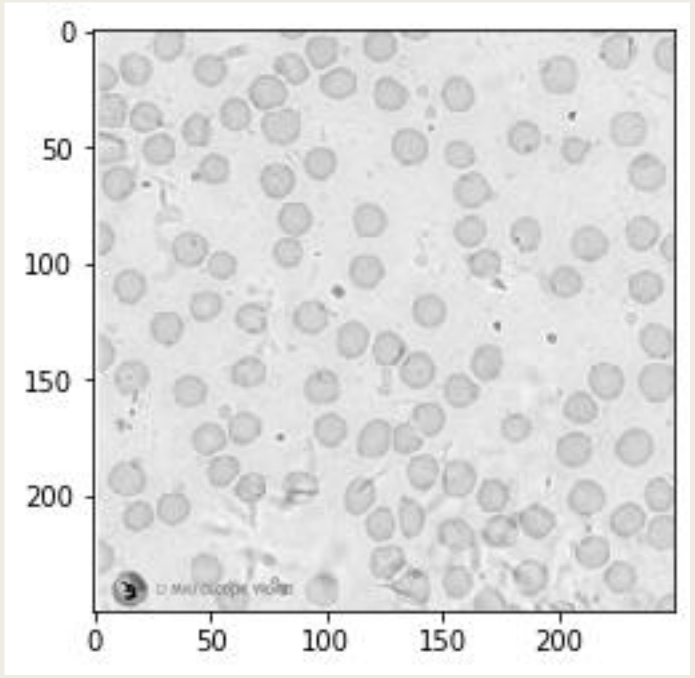


Figure 2: RGB to Grayscale

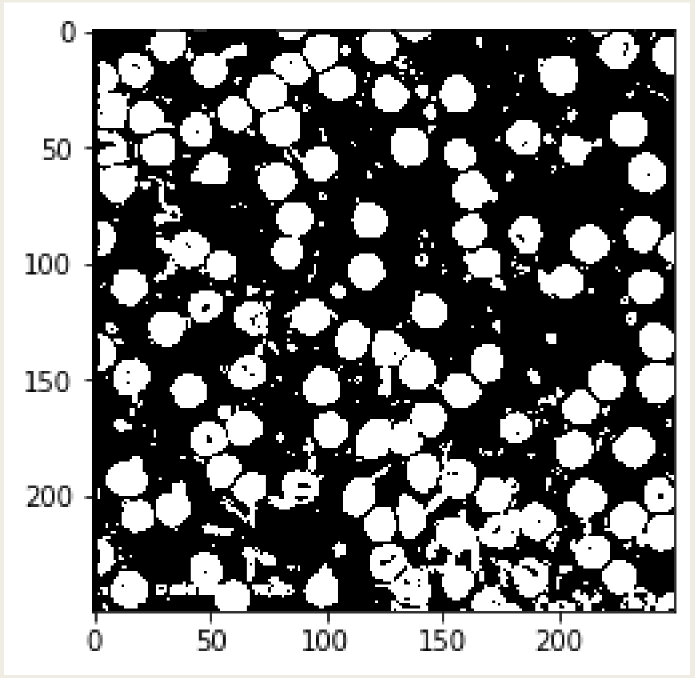


Figure 3: Grayscale to Binary (Segmentation)

Step 0: Blob Detection

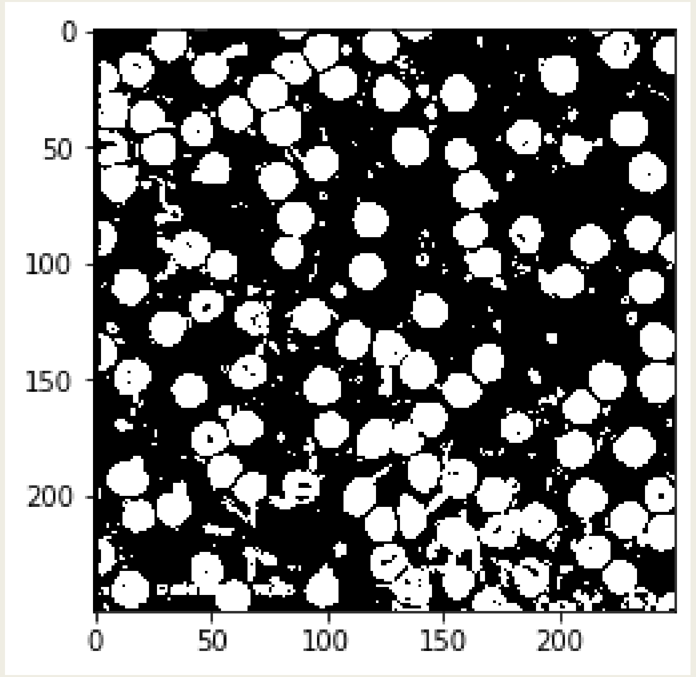


Figure 3: Grayscale to Binary (Segmentation)

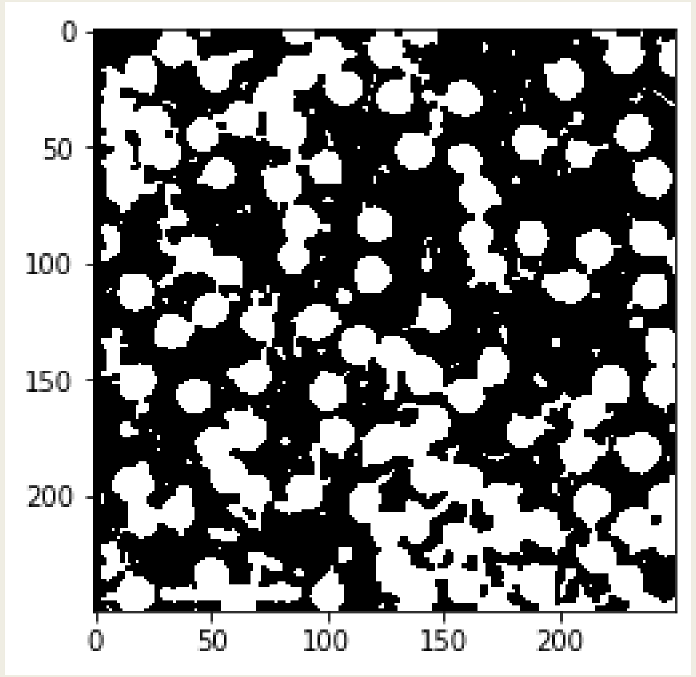


Figure 4: Holes were close via closing

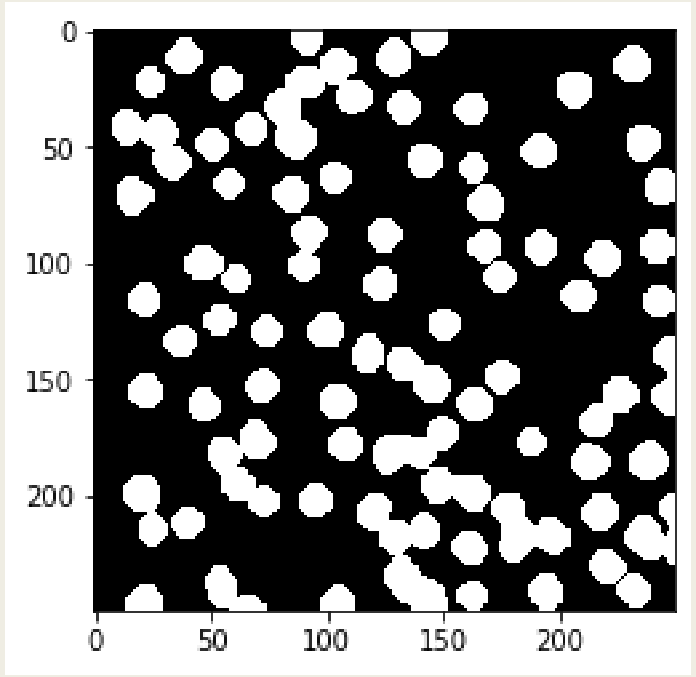
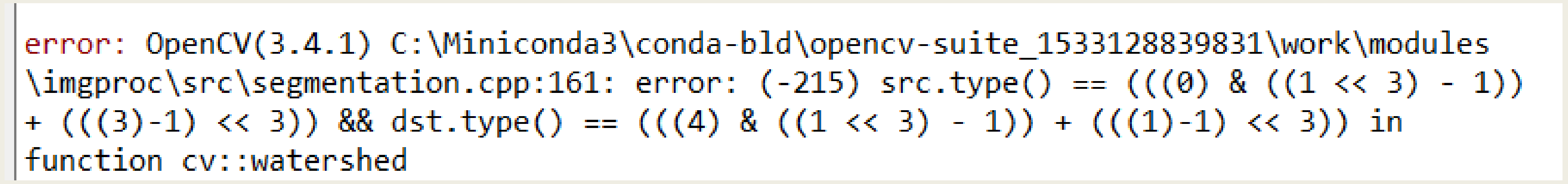


Figure 5: Noise removed via opening

Main Problem

- Some blobs merged together due to morphological operations and noise connecting blobs.
- This can be resolved via the watershed method as mam Jing recommended. Unfortunately I get an error so I have to come back at it later if I have time.
- For now, I will focus first on the separated blobs.



```
error: OpenCV(3.4.1) C:\Miniconda3\conda-bld\opencv-suite_1533128839831\work\modules
\imgproc\src\segmentation.cpp:161: error: (-215) src.type() == (((0) & ((1 << 3) - 1))
+ (((3)-1) << 3)) && dst.type() == (((4) & ((1 << 3) - 1)) + (((1)-1) << 3)) in
function cv::watershed
```

Step 0b: Labeling Regions

- Reference: https://docs.opencv.org/master/d3/db4/tutorial_py_watershed.html
- Using, cv2.connectedComponents(), it labels background of the image with 0, then other objects are labelled with integers starting from 1.
- Some blobs connected each other and/or are ‘incomplete’ since they are at the sides of the image. There are for now discarded.

Step 0b: Labeling Regions

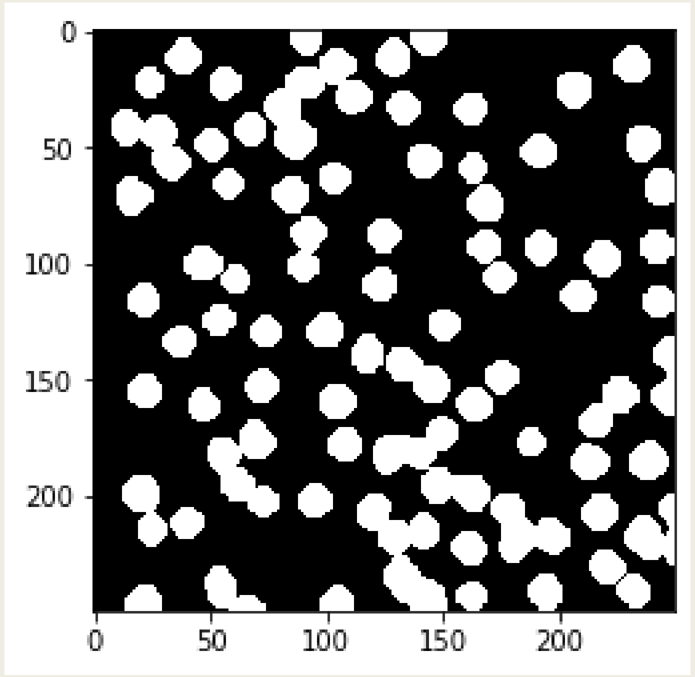


Figure 5: Noise removed via opening

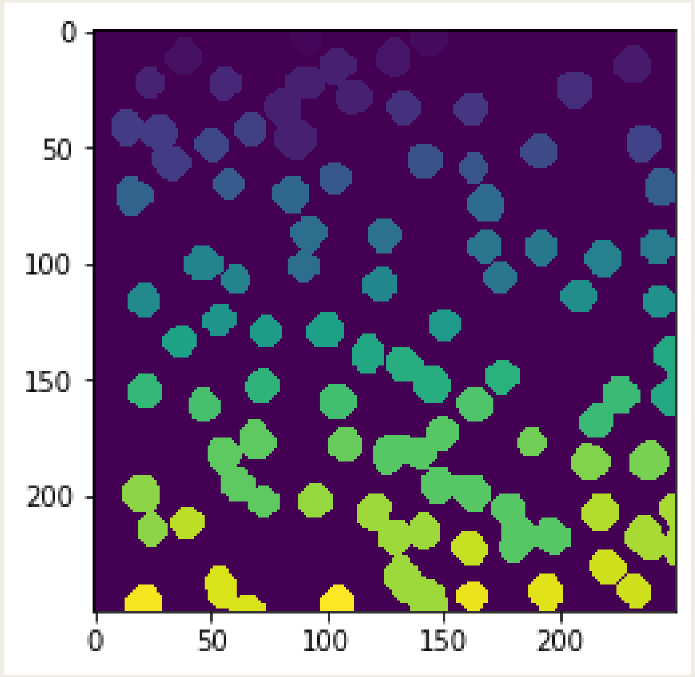


Figure 6: Each ROI is labelled a distinct integer. Background = 0

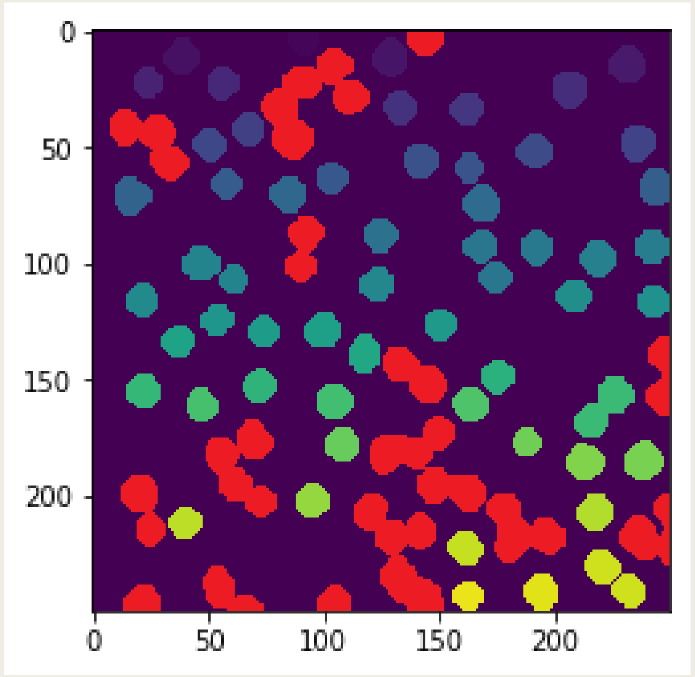


Figure 7: In Red, ROIs discarded due to blobs connecting together (desired)

Step 1: Getting the Area

- Reference: https://stackoverflow.com/questions/10741346/numpy-most-efficient-frequency-counts-for-unique-values-in-an-array
- The area of a blob is the number of pixels of a ROI
- Since the regions are index with a certain value, numpy.unique() collect the values used and the frequencies of those values.
- The average count of ROIs is the area while the standard deviation can be calculated with numpy.std()
- I got a mean of **173.5 pixels** with a standard deviation of **23.6 pixels.**

Step 2: Find the centroid and bounding circles

- Reference: https://www.learnopencv.com/blob-detection-using-opencv-python-c/
- The website teaches how to use SimpleBlobDetector() on OpenCV
- Figure 5 was inverted so that the regions of interest have 0 value to satisfy any arbitrary minimum threshold (say 100).
- Minimum convexity was set to .95 to only take account the circle and remove the clumped blobs and blobs at the sides.
- keypoints[k].pt[n] is a property of ‘keypoints’ that can extract x and y-coordinates of the centroids. k is the index bounding circle while n is the axis number.

Step 2: Find the centroid and bounding circles

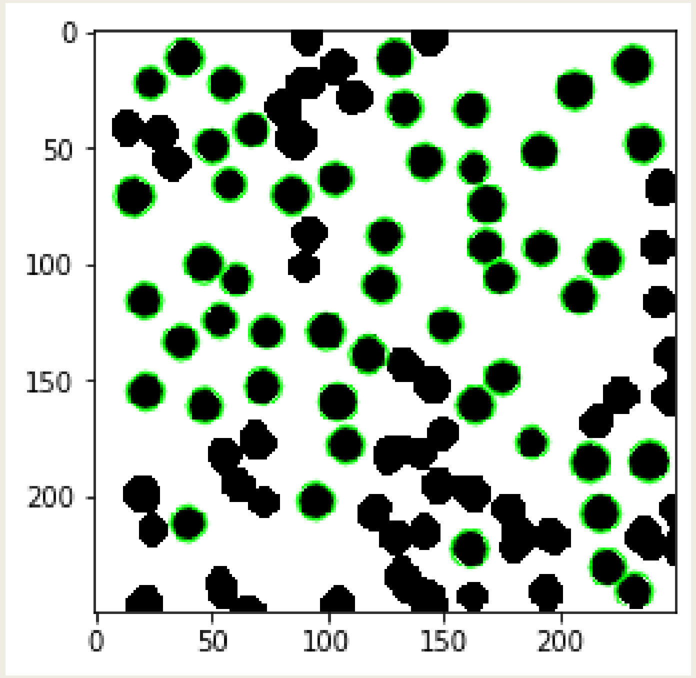


Figure 8: Minimum bounding circles determined

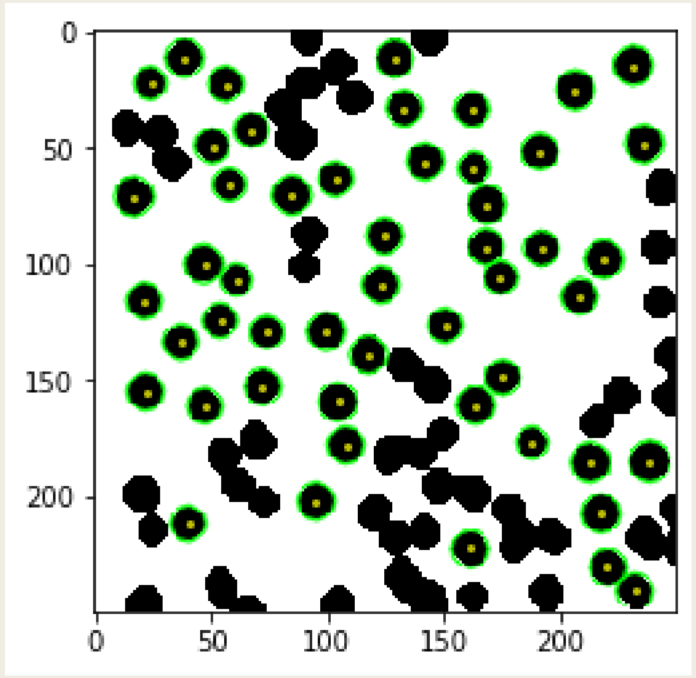


Figure 9: Centroid of each bounding circle was determined

Step 2: Bounding boxes (unfinished)

- Since SimpleBlobDetector() uses a circular bounding box.
- For rectangles cv::boundingRect() may be used.
- *For circles cv::minEnclosingCircle may be used
- Documentation: https://docs.opencv.org/3.4/da/d0c/tutorial_bounding_rects_circles.html

Step 3: eccentricity, major axis length, or perimeter. (unfinished)

Feelings

- I see a lot of codes on the internet and to be honest I have a hard time realizing how the code works especially for incomplete/large codes which took most of my time.
- Took me a week just to try to find a segment but it still not the best.
- In OpenCV, every morphological operation done, the results shifts a bit to the right and down….