Mete UZ

COMP 448

HW1

General Psuedocode:

- Noise Elimination: I used non-local means denoising from cv2 library with the recommended parameters which gave me a satisfactory result.
- Increase contrast to make edges more distinct
- Blur image to make edges smoother, I used gaussian blur with a small kernel

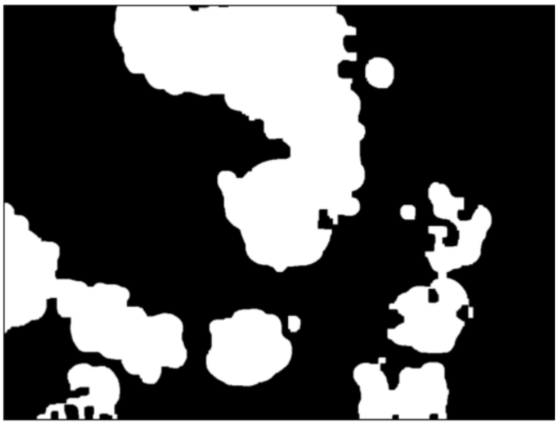
Part 1:

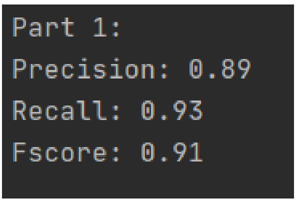
Psuedocode:

```
- Thresholding, I used Otsu's method
- Closing , I used a large kernel to connect regions
- Dilation, I dilated the image with a small kernel to close some of the unconnected regions
- Fill large holes with flood fill, I used this to avoid losing some of the detail
- Opening, I used a 9 by 9 kernel to get rid of the smaller regions and make larger regions more distinct
- Closing, A final Closing operation with a 11 by 11 kernel gave me a more desirable result
```

Results:

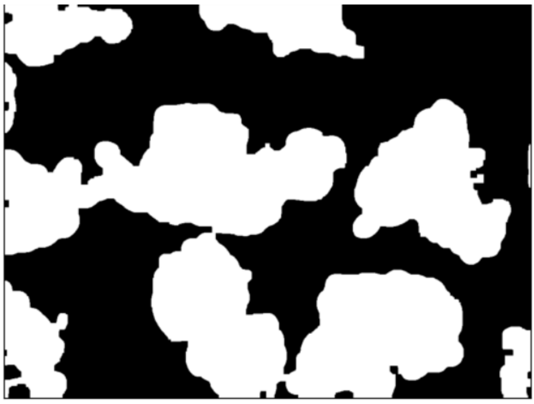
Image 1:

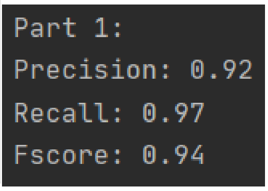




```
Part 1:
Precision: 0.89
Recall: 0.93
Fscore: 0.91
```

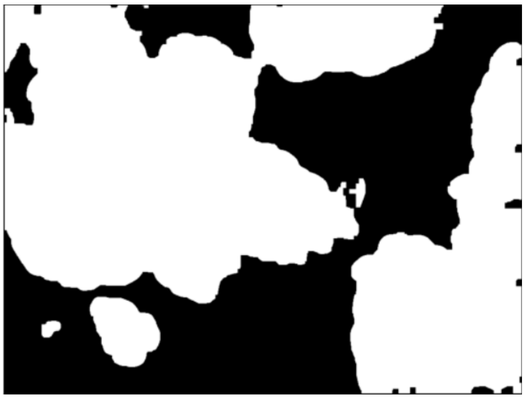
Image 2:

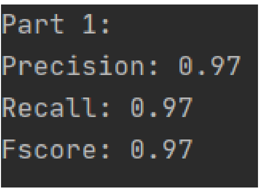




```
Part 1:
Precision: 0.92
Recall: 0.97
Fscore: 0.94
```

Image 3:





```
Part 1:
Precision: 0.97
Recall: 0.97
Fscore: 0.97
```

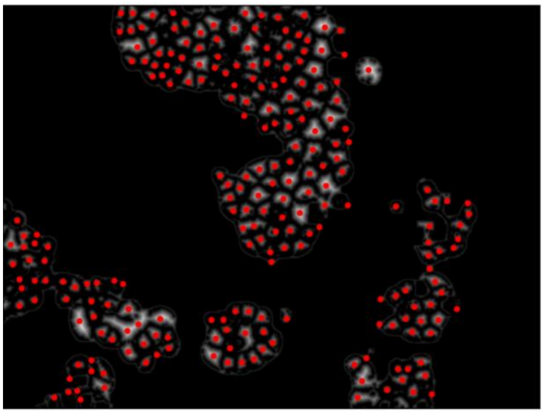
Part 2:

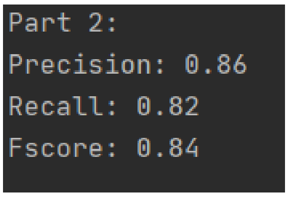
Psuedocode:

- Get the white boundaries from the original image and overlay them to the mask created in part 1 to find boundaries. I used the grayscale image and used the pixels with value above 195 to find the boundaries. I also used erosion with 3 by 3 kernel to get rid of small details.
- Distance transform to find points further from boundaries.
- Find the local maxima and save that spot as the cell coordinates.

Results:

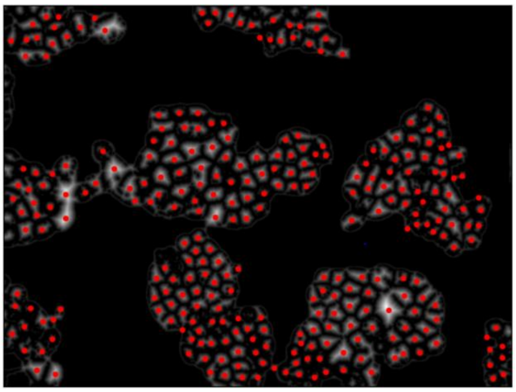
Image 1:

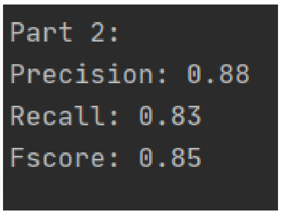




```
Part 2:
Precision: 0.86
Recall: 0.82
Fscore: 0.84
```

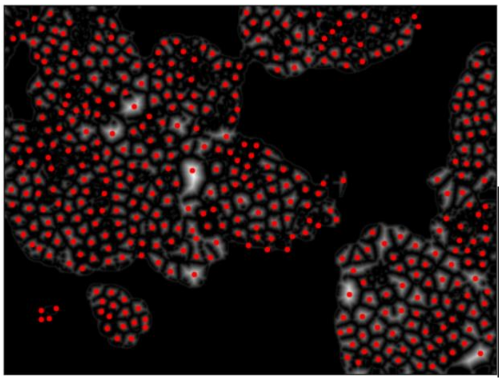
Image 2:

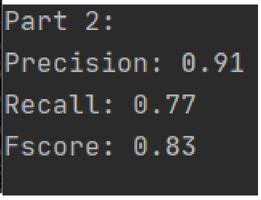




```
Part 2:
Precision: 0.88
Recall: 0.83
Fscore: 0.85
```

Image 3:





```
Part 2:
Precision: 0.91
Recall: 0.77
Fscore: 0.83
```

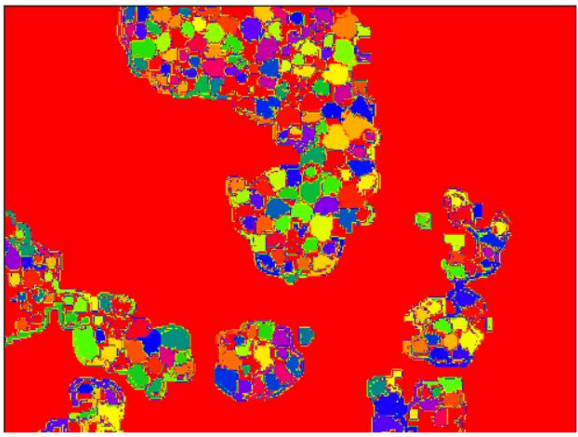
Part 3:

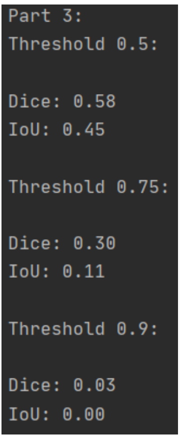
Psudeocode:

```
- Use the boundaries found on the mask in part 2 and the coordinates also found in part 2.
- Use region growing to fit those regions into the boundaries:
    - I grew each region in small increments so that if there is any overlap the algorithm will not paint over the same pixel twice.
    - I used a bigger square to paint over the image mask/boundary overlay to improve performance, padding so that the index is not out of bounds.
```

Results:

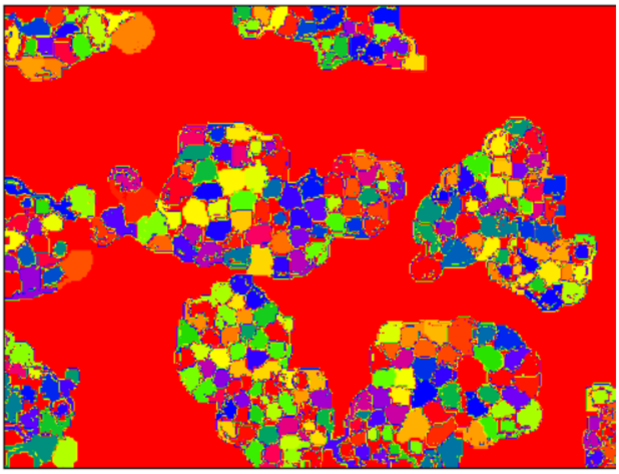
Image 1:

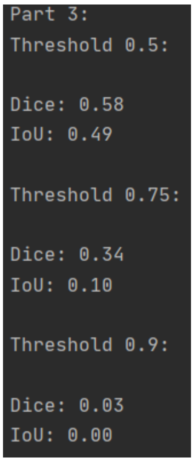




```
Part 3:
Threshold 0.5:

Dice: 0.58
IoU: 0.45

Threshold 0.75:

Dice: 0.30
IoU: 0.11

Threshold 0.9:

Dice: 0.03
IoU: 0.00
```

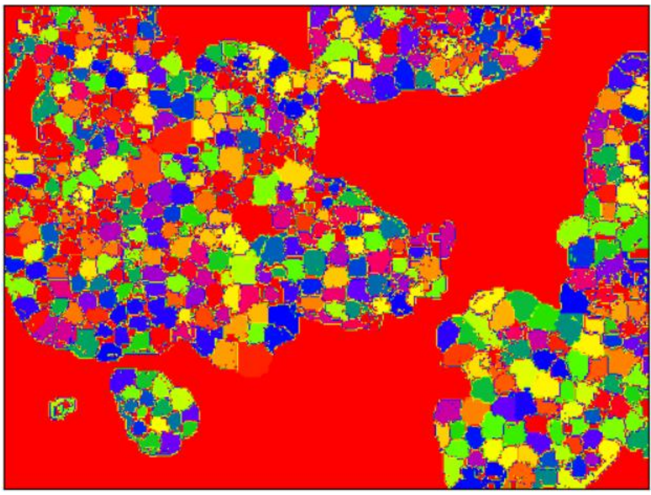
Image 2:

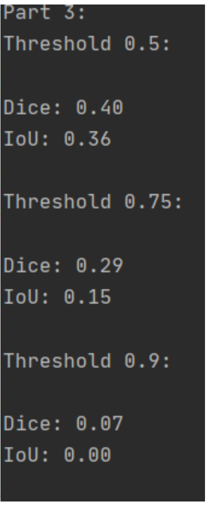




```
Part 3:
Threshold 0.5:

Dice: 0.58
IoU: 0.49

Threshold 0.75:

Dice: 0.34
IoU: 0.10

Threshold 0.9:

Dice: 0.03
IoU: 0.00
```

Image 3:





```
Part 3:
Threshold 0.5:

Dice: 0.40
IoU: 0.36

Threshold 0.75:

Dice: 0.29
IoU: 0.15

Threshold 0.9:

Dice: 0.07
IoU: 0.00
```