Q1.

For the pseudocode of this part, first I tried different filters like Gaussian Blur, Laplacian of Gaussian, Median Filter and Sobel. When I checked them with my eyes, Sobel seemed more reasonable to continue with. I used grayscale image for Sobel. After I get the result from Sobel, I apply a thresholding operation. In the end, to refine the masks more, I apply dilate and erode operations. Hyperparameters in this code can be listed as following:

- Kernel size for Sobel : 3
- Threshold : 21
- Kernel size for erode & dilate : 5
- # of iterations for erode : 1
- # of iterations for dilate : 3

To select these hyperparameters, I tried lots of different parameters and try to understand which values I need to use. I experimented with larger kernel sizes, smaller kernel sizes, larger and smaller threshold values and different # of iterations.

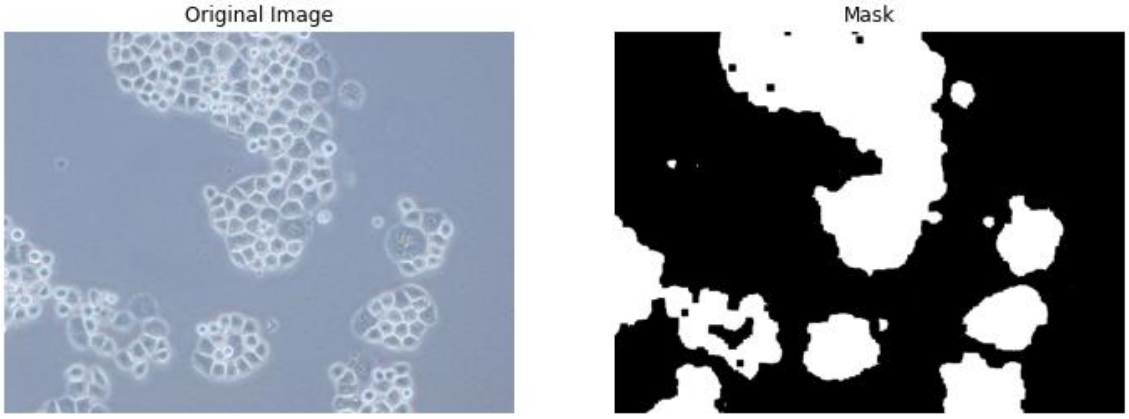

Figure 1: Image 1 and Mask 1

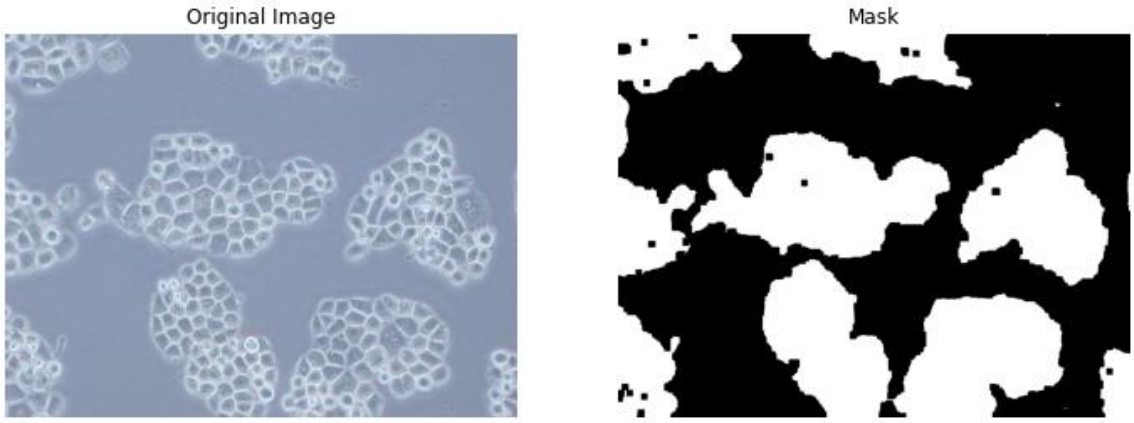

Figure 2: Image 2 and Mask 2

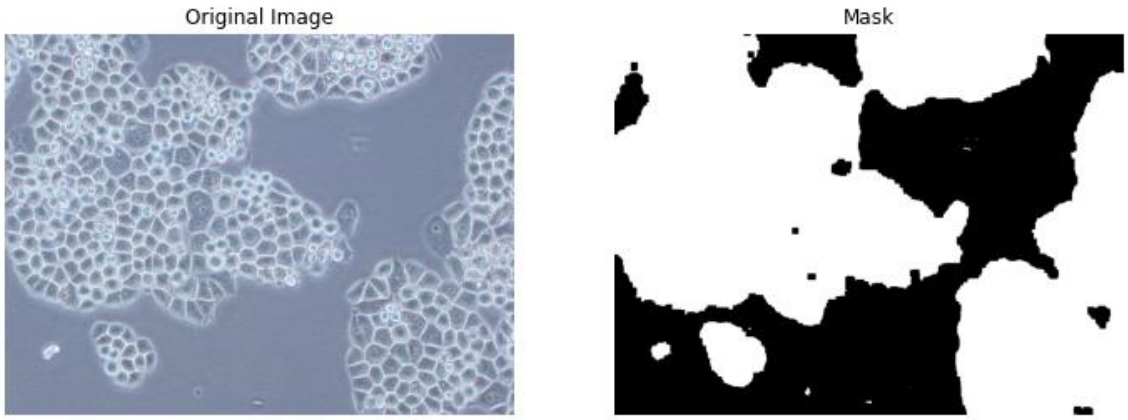

Figure 3: Image 3 and Mask 3

```
Image1 results
True Positives (TP): 226006
True Negatives (TN): 543846
False Positives (FP): 10565
False Negatives (FN): 6015
precision:  0.9553411026710797
recall:  0.9740756224652078
f1 score:  0.9646174070406665

Image2 results
True Positives (TP): 323427
True Negatives (TN): 442248
False Positives (FP): 8554
False Negatives (FN): 12203
precision:  0.9742334651681873
recall:  0.9636415099961266
f1 score:  0.9689085410516003

Image3 results
True Positives (TP): 468447
True Negatives (TN): 304152
False Positives (FP): 7813
False Negatives (FN): 6020
precision:  0.9835950951161131
recall:  0.987312078606099
f1 score:  0.9854500818847052
```

Figure 4: Results for both images.

Q2.

For the pseudocode of this part, I started with grayscale images. First applied Gaussian Blur to the image to smooth the image, then I applied thresholding. Then I used dilate and erode operations. With the help of bitwise_not and bitwise_and operations, I tried to obtain the locations of the cells. Than I applied distance transform to get the centers. In the end, with connectedComponentsWithStats, I returned centroids. Hyperparameters of this code can be listed as follows:

- Kernel size & standard deviation for Gaussian Blur : 5, 0.03
- Threshold: 203
- Kernel size for erode & dilate : 1
- # iterations for erode & dilate: 1 & 3
- Neighborhood size in distance transform: 5

Both hyperparameters are selected with trying. I tried lots of different kernel sizes, threshold and iteration numbers to get better results.

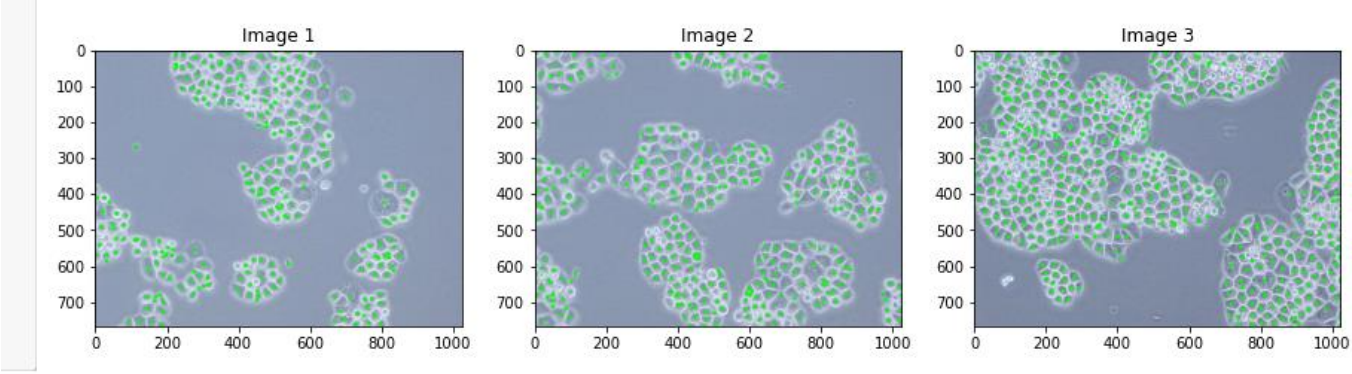

Figure 5: Regional Maximas (centroids)

```
Image 1
True Positives:  236
False Positives:  17
False Negatives:  6.0
Precision:  0.932806324110672
Recall:  0.9752066115702479
F1 Score:  0.9535353535353536

Image 2
True Positives:  302
False Positives:  32
False Negatives:  9.0
Precision:  0.9041916167664671
Recall:  0.9710610932475884
F1 Score:  0.9364341085271317

Image 3
True Positives:  439
False Positives:  43
False Negatives:  70.0
Precision:  0.9107883817427386
Recall:  0.862475442043222
F1 Score:  0.8859737638748738
```

Figure 6: Results of algorithm

Q3.

For the pseudocode of this question, I took the mask from Q1, and make all 1s 255. Then I apply Gaussian Blur to the original image. Then the blurred image is bitwise ANDed with itself using the adjusted mask. This operation retains only the regions of the image where cells are present, effectively masking out the background. Then I did a channel selection. The masked image is split into its color channels, and the red channel is extracted. Then I did a edge detection with a 3x3 kernel, and a normalization. Then a thresholding, another Gaussian Blur and image enhancement applied to enhance the boundaries. Then region grow algorithm applied. The hyperparameters of this algorithm can be listed as follows:

- Kernel sizes of Gaussian Blur
- 3x3 manual kernel size
- Thresholds
- Depth used in region grow algorithm

Both hyperparameters selected with a trying process. I tried a lot of different values and decided to stick with these.

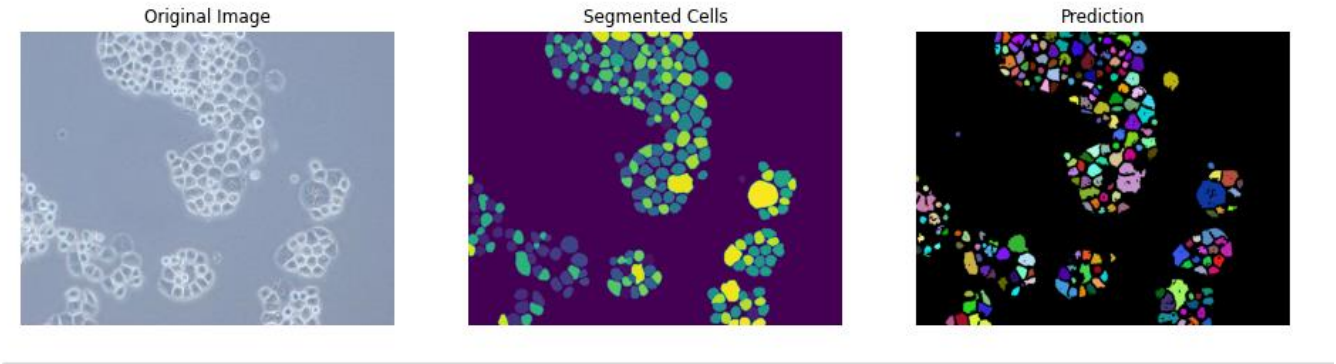

Figure 7: Segmentation map of image 1

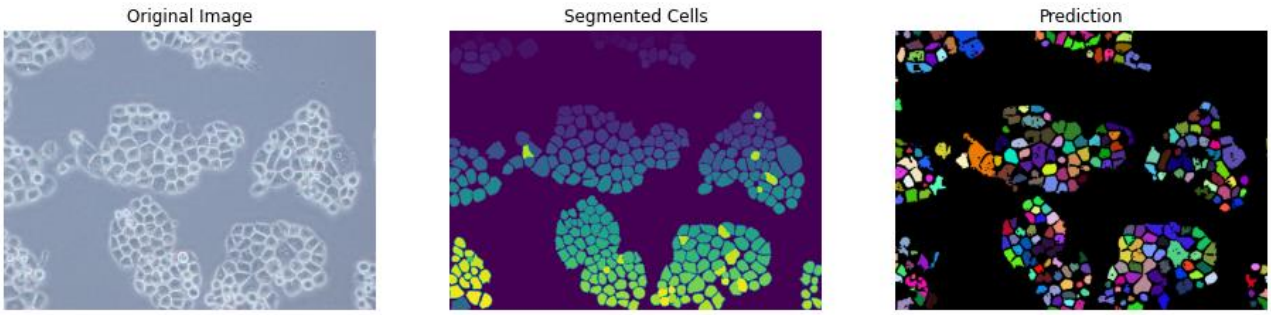

Figure 8: Segmentation map of image 2

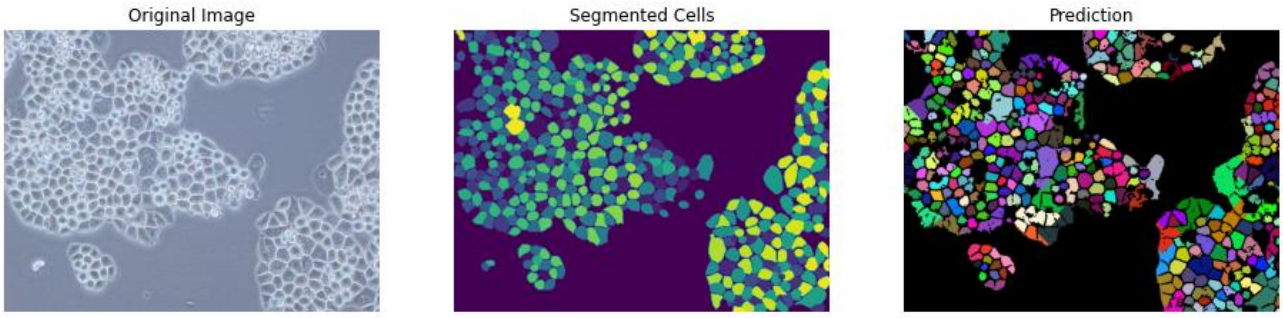

Figure 9: Segmentation map of image 3

```
Image 1:
Ratio :  0.5
precision : 0.5354330708661418
recall : 0.5619834710743802
fscore :  0.5483870967741936
Image 1:
Ratio :  0.75
precision : 0.14960629921259844
recall : 0.15702479338842976
fscore :  0.15322580645161293
Image 1:
Ratio :  0.9
precision : 0.007874015748031496
recall : 0.008264462809917356
fscore :  0.008064516129032258
dice index : 0.7229994852304977
```

Figure 10: Results of image 1

```
Image 2:
Ratio :  0.5
precision : 0.4626865671641791
recall : 0.4983922829581994
fscore :  0.47987616099071206
Image 2:
Ratio :  0.75
precision : 0.21492537313432836
recall : 0.2315112540192926
fscore :  0.22291021671826622
Image 2:
Ratio :  0.9
precision : 0.005970149253731343
recall : 0.006430868167202572
fscore :  0.006191950464396284
dice index : 0.6790253694051392
```

Figure 11: Results of image 2

```
Image 3:
Ratio :  0.5
precision : 0.4472049689440994
recall : 0.4243614931237721
fscore :  0.43548387096774194
Image 3:
Ratio :  0.75
precision : 0.14699792960662525
recall : 0.13948919449901767
fscore :  0.14314516129032256
Image 3:
Ratio :  0.9
precision : 0.002070393374741201
recall : 0.0019646365422396855
fscore :  0.002016129032258065
dice index : 0.6290080471774077
```

Figure 12: Results of image 3

Q4.

For the pseudocode of this question, I used the grayscale image. Then I applied Contrast Limited Adaptive Histogram Equalization (CLAHE) to enhance the contrast of grayscale image. Continued with Gaussian Blur and Canny edge detector. Then continued with thresholdings and erode dilate operations to refine the output. The hyperparameters of this code are listed as following:

- Kernel sizes
- # of iterations on erode and dilate
- Thresholds
- Clip limit & grid size on CLAHE

All these hyperparameters are selected with a try and see method. I tried a lot of different filters and parameters to find better ones.

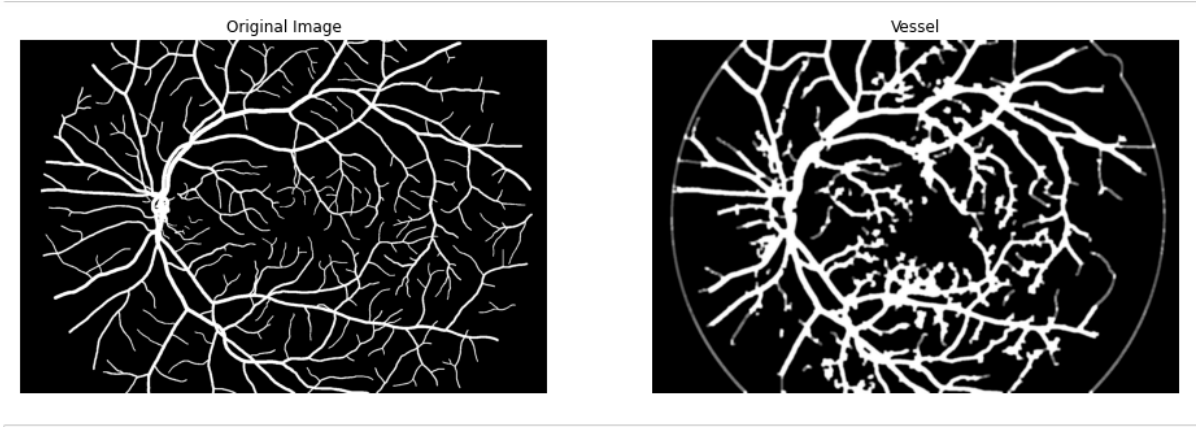

Figure 13. Vessels of image 1

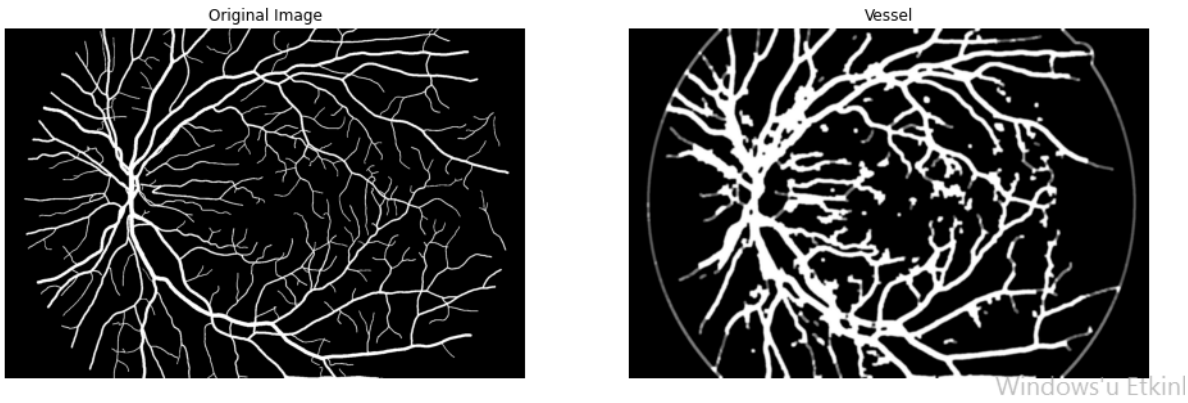

Figure 14. Vessels of image 2

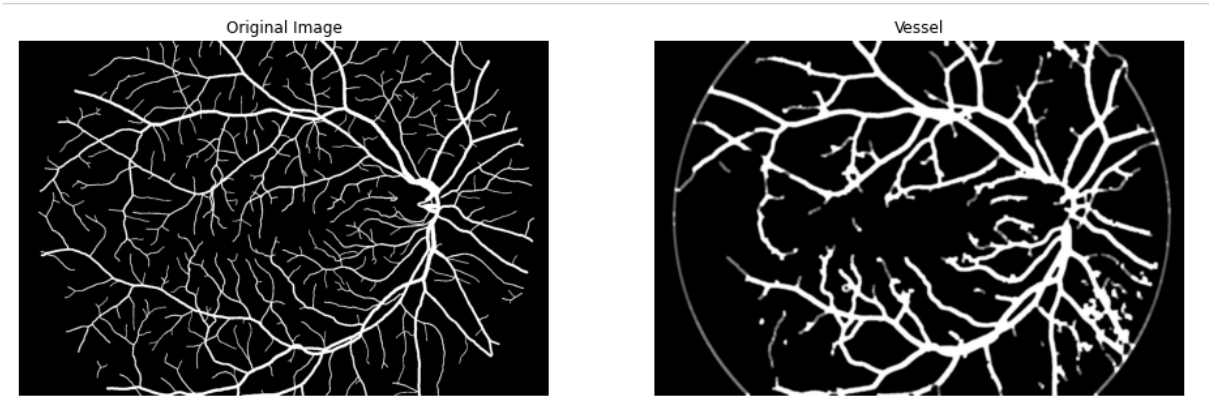

Figure 15. Vessels of image 3

```
Image1
precision: 0.7376397065681977
recall: 0.7630225773862901
F-score: 0.750116473381299

Image2
precision: 0.656563748782282
recall: 0.7715644128779029
F-score: 0.709433840667662

Image3
precision: 0.8056731107732179
recall: 0.653393835574832
F-score: 0.7215869640323485
```

Figure 16. Results of vessel segmentations.

Berk Coşar 69557