ED6001
MEDICAL IMAGE ANALYSIS

ASSIGNMENT 3 - IMAGE SEGMENTATION TECHNIQUES

Swathi V (ED18B034)

26 October 2021

INTRODUCTION

Image Segmentation techniques can be used to separate out the region of interest in an image. In this report, we first look at various segmentation techniques, including Thresholding and Graph based techniques. We then define metrics that can be used to understand the performance of these techniques.

The given dataset contains 5 images, all of which are monochromatic. Thus, we read them as greyscale images for ease of computations. An example is shown in Fig. 1.

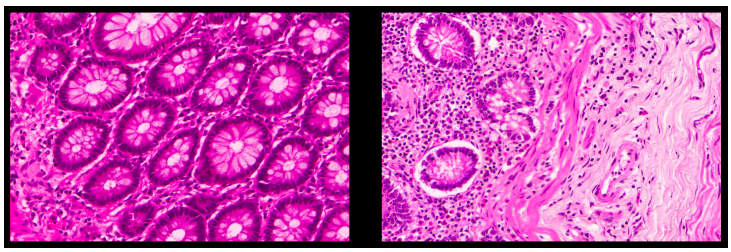


Figure 1: Input images 1 and 2

Note: The .ipynb notebook has been added to the submissions, and can be run using google colaboratory. The function cv2_imshow() has been used instead of cv2.imshow() to display images on google colaboratory. This can be changed if any other software is being used to run the file.

QUESTION 1: HISTOGRAM BASED SEGMENTATION TECHNIQUES

Thresholding is a popular technique for image segmentation because of it's relative simplicity of implementation and computational advantages. The thresholding techniques that will be analysed in this report are:

- Basic Global Thresholding
- Optimal Global Thresholding using Otsu's Method
- Multiple Thresholding
- Variable Thresholding

THRESHOLDING

If the histogram of an image $f(x,y)$ resembles Fig. 2(a), where the image is known to contain bright objects on a dark background, an obvious way to perform the segmentation between **background** and **foreground** is to set a threshold (T) to divide the intensity values. The segmented image g is then given by

$$g(x,y) = \begin{cases} 1 & \text{if } f(x,y) > T \text{ (foreground)} \\ 0 & \text{if } f(x,y) \leq T \text{ (background)} \end{cases} \tag{1}$$

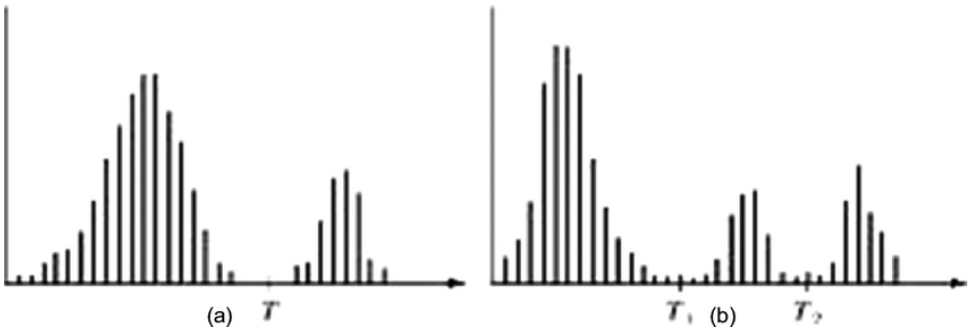


Figure 2: Intensity histograms that can be partitioned using (a) single and (b) multiple thresholding

However, if the histogram of an image resembles Fig. 2(b), using a single threshold will mute some features of the image. In this case, multiple thresholds (T_1 and T_2) can be used, wherein the segmented image g is given by

$$g(x,y) = \begin{cases} a & \text{if } f(x,y) > T_2 \\ b & \text{if } T_1 < f(x,y) \leq T_2 \\ c & \text{if } f(x,y) \leq T_1 \end{cases} \tag{2}$$

(1) BASIC GLOBAL THRESHOLDING

As explained above, when the intensity levels of the image are distinct, we can use a single threshold to perform image segmentation effectively. The iterative algorithm used for this purpose is shown in Fig. 3.

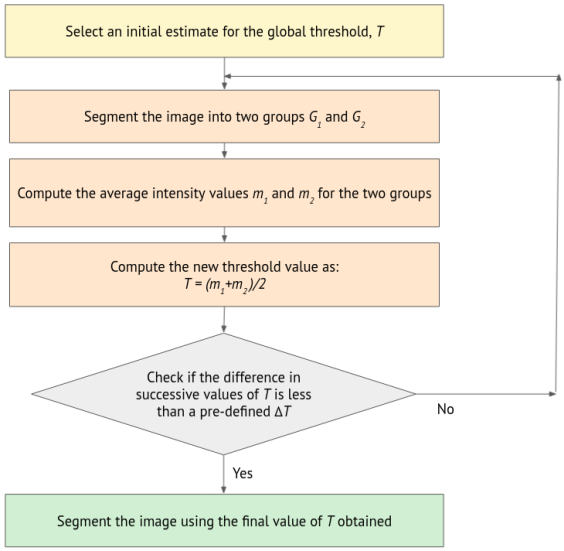


Figure 3: Flowchart illustrating the basic global thresholding algorithm

For the first input images, global thresholding was performed at varying initialisations (100, 125, 200) and difference threshold levels(1, 2). The results are shown in Fig. 4 to 9.

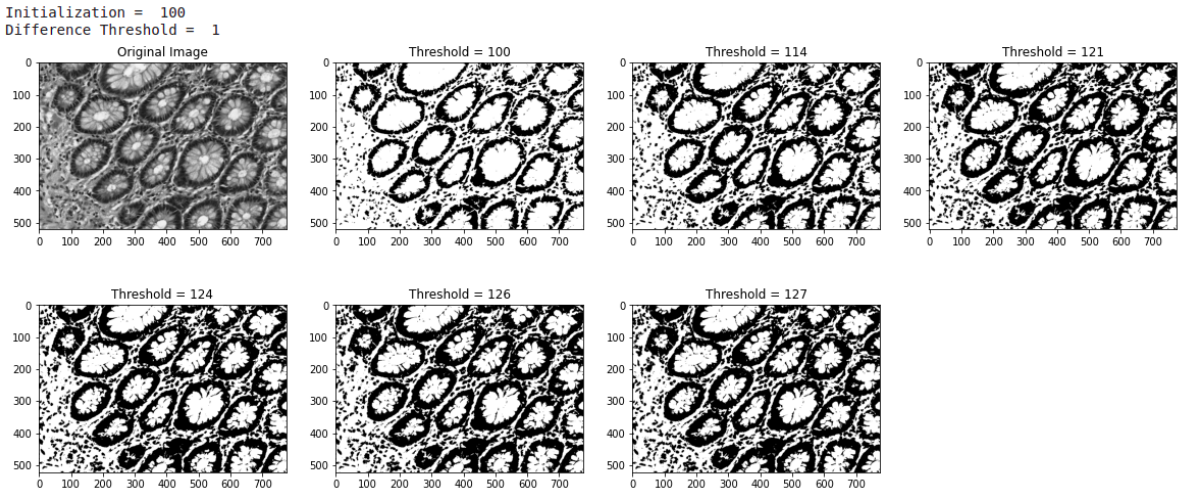


Figure 4: Iteration wise segmentation results for Image 1 (Initialization = 100, Difference Threshold = 1)

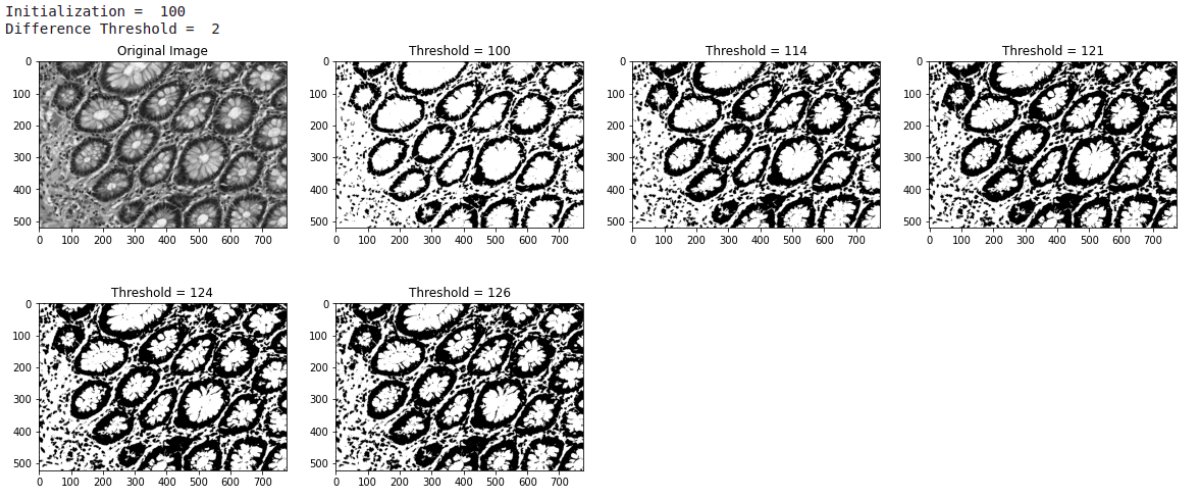


Figure 5: Iteration wise segmentation results for Image 1 (Initialization = 100, Difference Threshold = 2)

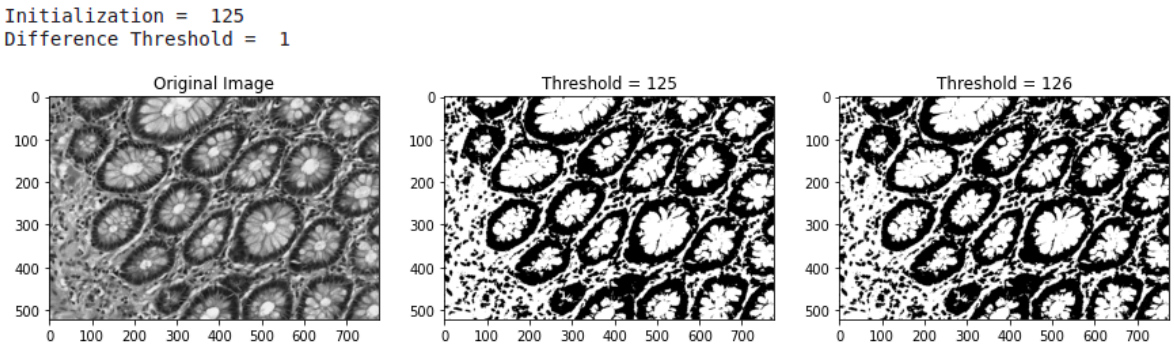


Figure 6: Iteration wise segmentation results for Image 1 (Initialization = 125, Difference Threshold = 1)

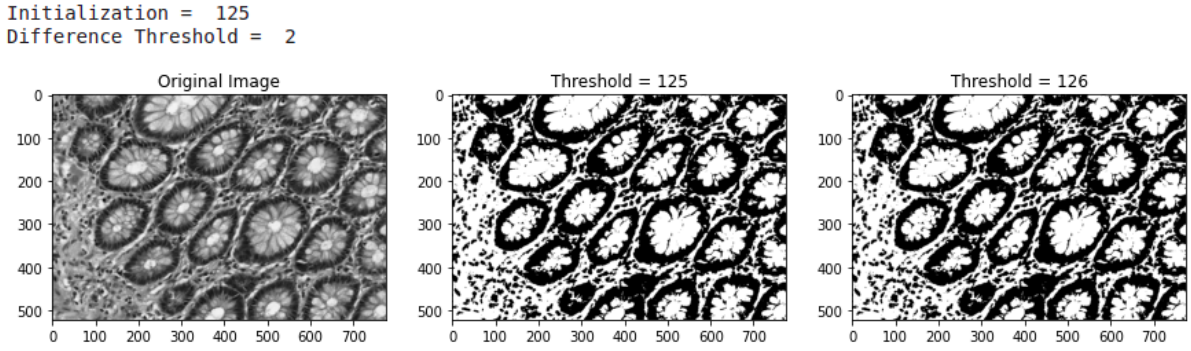


Figure 7: Iteration wise segmentation results for Image 1 (Initialization = 125, Difference Threshold = 2)

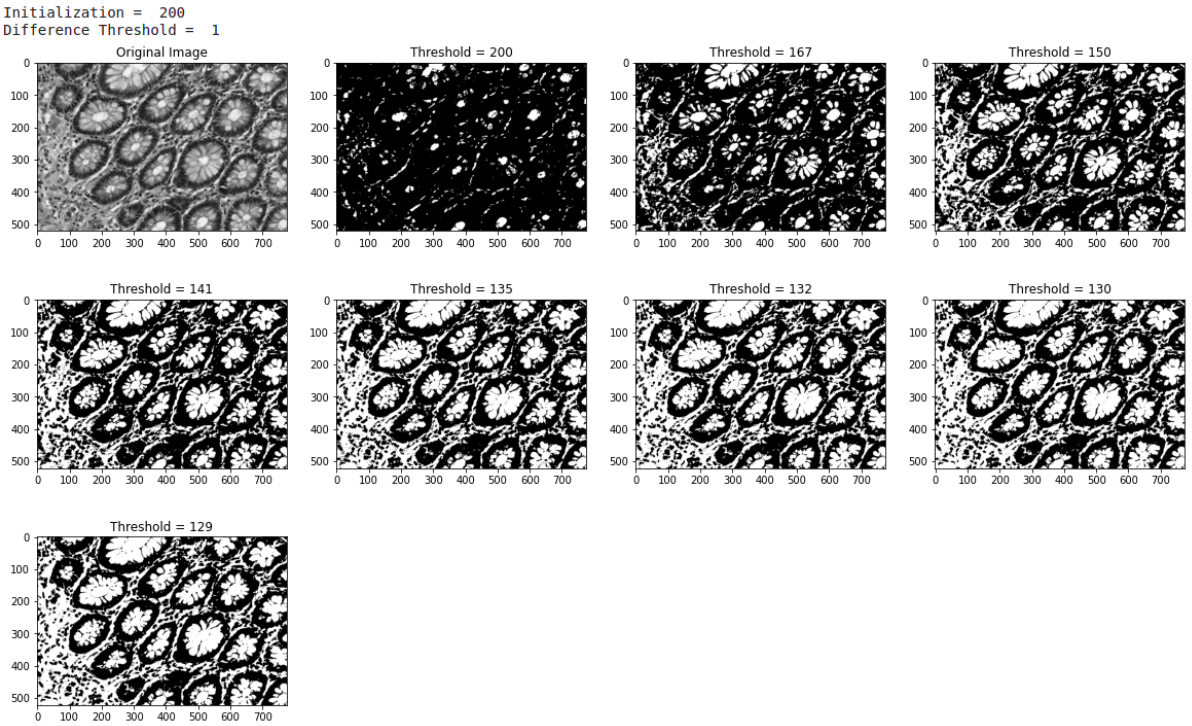


Figure 8: Iteration wise segmentation results for Image 1 (Initialization = 200, Difference Threshold = 1)

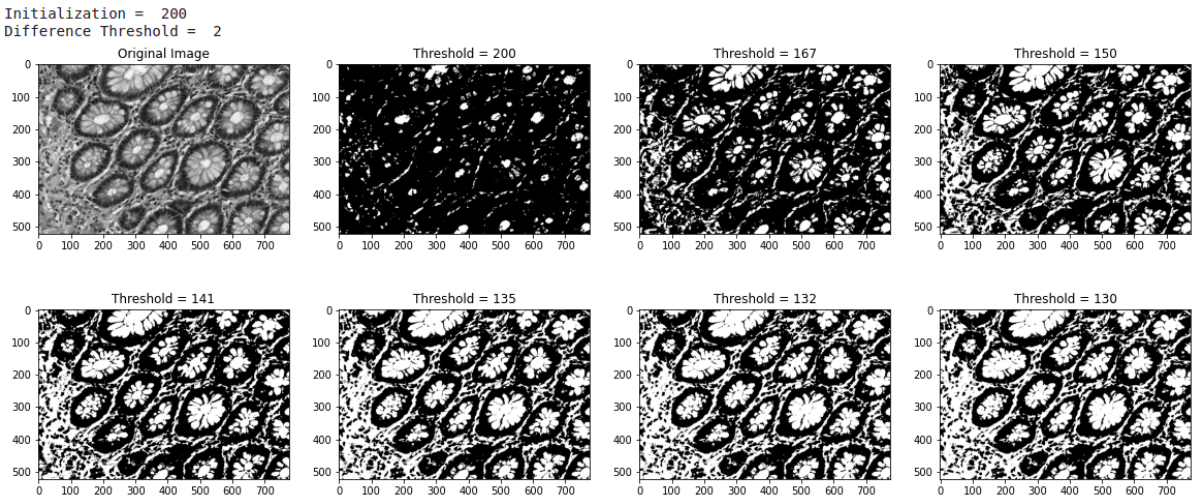


Figure 9: Iteration wise segmentation results for Image 1 (Initialization = 200, Difference Threshold = 2)

Observations

- For a small enough difference threshold (1 or 2), the global thresholding algorithm always converges near the same value, irrespective of the initialization.
- The number of iterations taken to converge varies widely depending on the initialization.

- The difference between thresholds for consecutive iterations reduces successively, making the algorithm converge.
- Setting a smaller difference threshold takes more number of iterations to converge, but gives better results in terms of reaching close to the best threshold value.
- Setting a very low initialization (in this case 100) makes the algorithm start with very skewed segmentation, where most points are classified to the foreground rather than the background.
- Similarly, setting a very high initialization (in this case 200) leads to most points being classified to the background rather than the foreground.

The final result of the global thresholding segmentation algorithm for each of the five images is shown in Fig. 10 to Fig. 14.

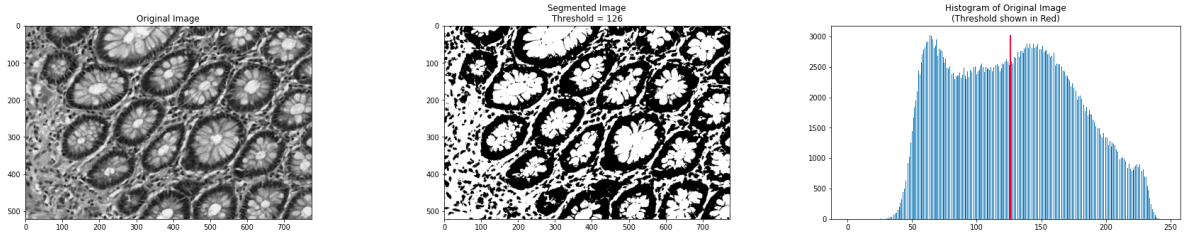


Figure 10: Result of global thresholding for Image 1

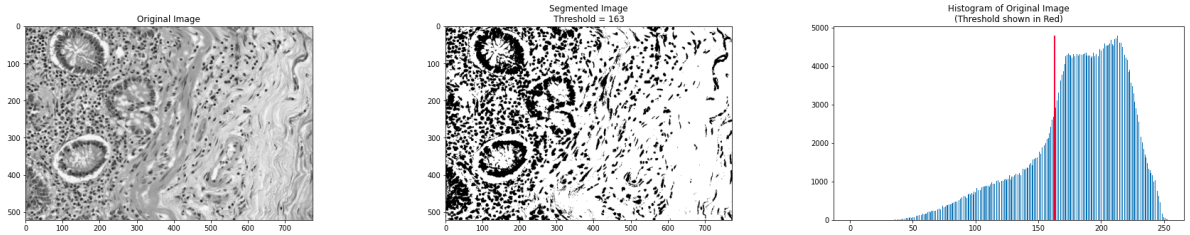


Figure 11: Result of global thresholding for Image 2

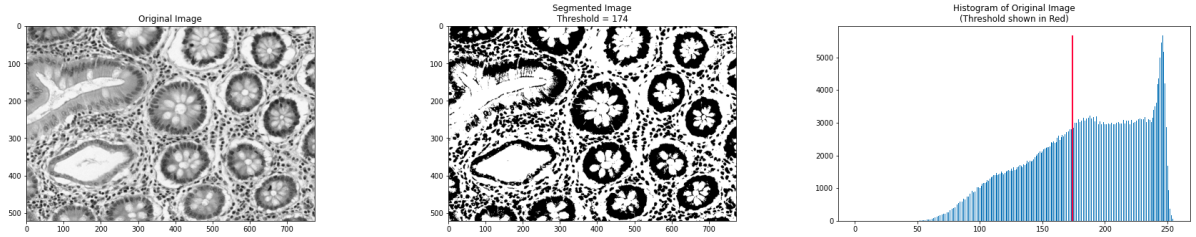


Figure 12: Result of global thresholding for Image 3

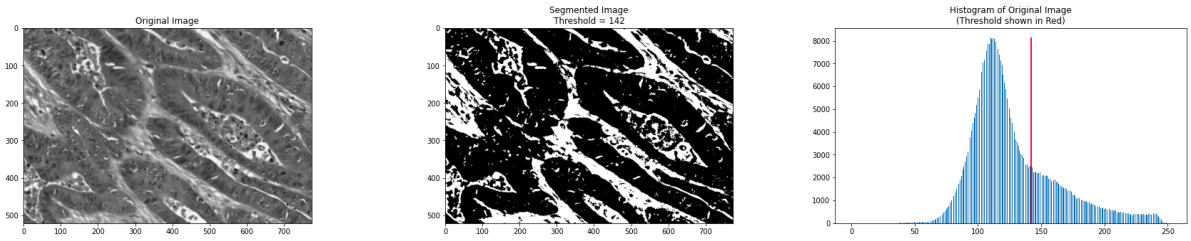


Figure 13: Result of global thresholding for Image 4

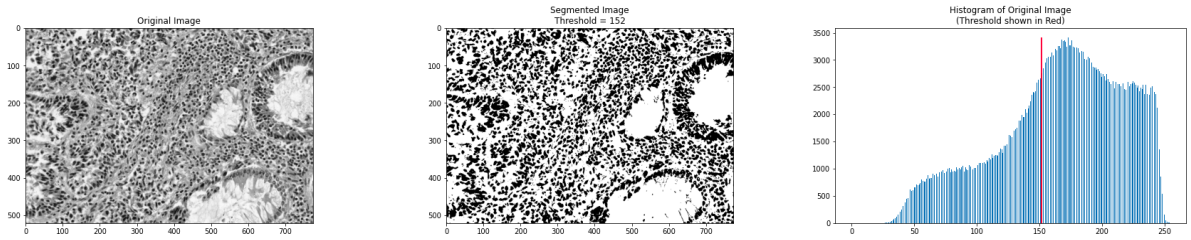


Figure 14: Result of global thresholding for Image 5

It can be seen that for Images 2 and 5 (which are noisier than the others), the global thresholding algorithm does not give very good results. A smoothing mask (Mean filter) was applied to the original image to analyse the effect of the same in thresholding. The results are shown in Fig. 15 and Fig. 17.

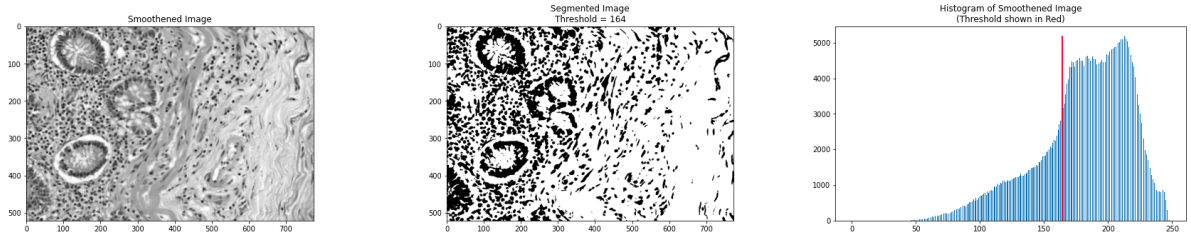


Figure 15: Result of global thresholding on smoothened image for Image 2 (filter size = 3)

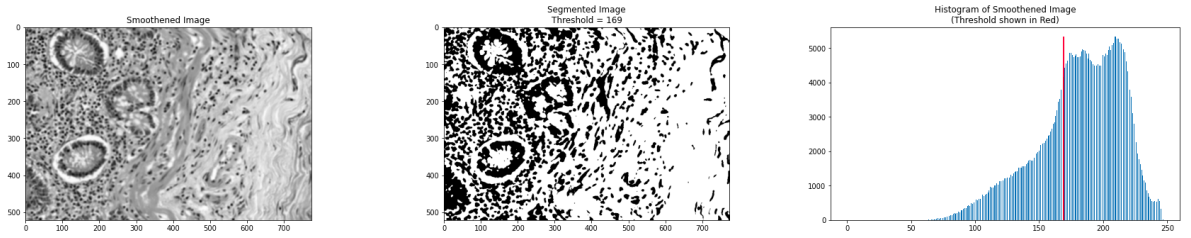


Figure 16: Result of global thresholding on smoothened image for Image 2 (filter size = 5)

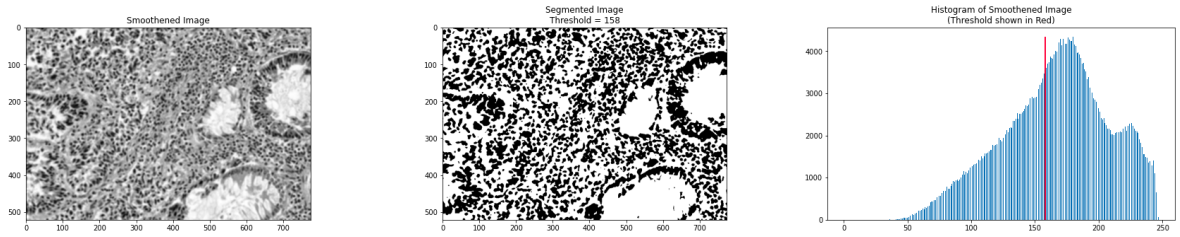


Figure 17: Result of global thresholding on smoothened image for Image 5 (filter size = 5)

Observations

- It is clear that when the histograms of the foreground and background are clearly distinct (example, Image 1), the global thresholding algorithm gives good results.
- When the histograms are not distinct, a smoothening mask can help reduce the noise and make the foreground and background more distinct, which gives better segmentation results.
- It is also observed that as a smoothening mask leads to loss of features from the original image, the same can be expected for the segmented image as well.
- Using a smoothening mask of higher window size leads to more loss of features but reduces the noisy components to a higher extent.

OTSU'S METHOD

Otsu's method is used to find a global threshold for segmentation using an optimization problem. The goal is to maximize the *between-class variance*, which also leads to the minimization of *within-class variance*. A major advantage of the Otsu's method is that all computations are performed based on the histogram of the image, which is an easy to compute 1D array. Otsu's algorithm is summarized in Fig. 18.

The results of Otsu's Thresholding algorithm are shown in Fig. 19 to Fig. 23.

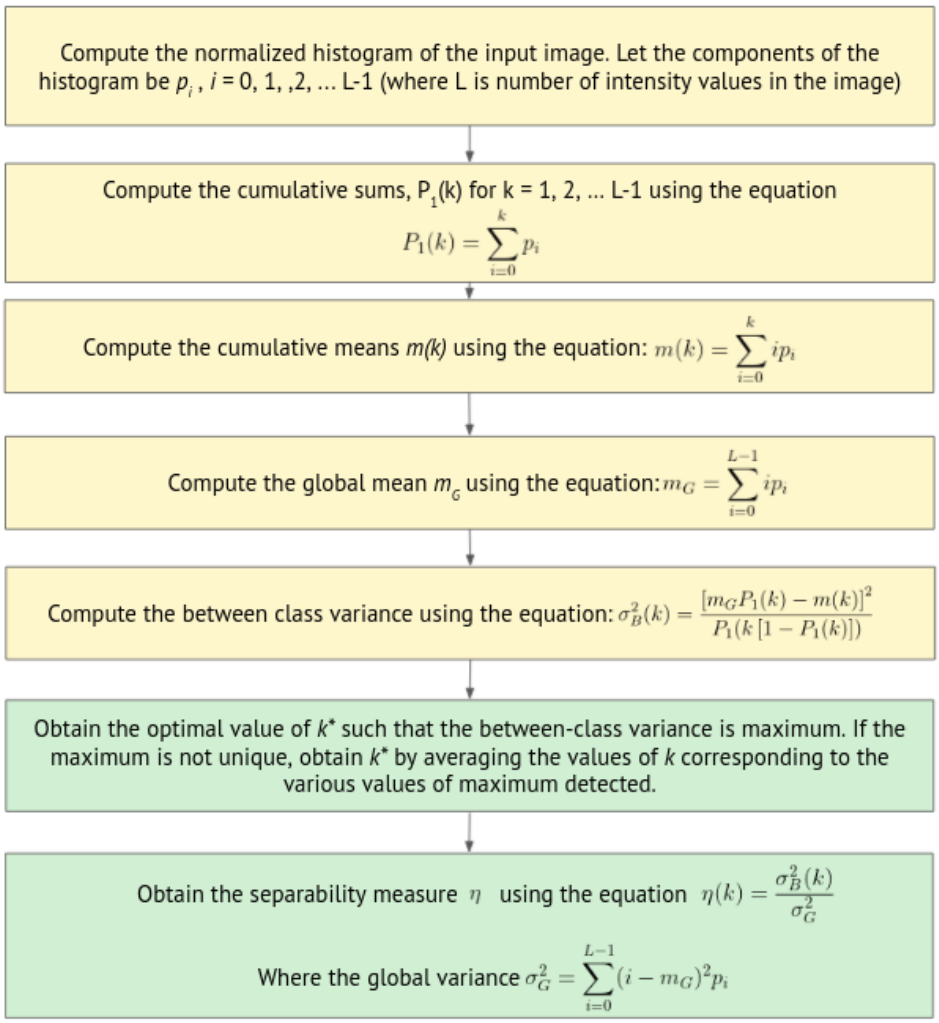


Figure 18: Flowchart illustrating the Otsu's Thresholding Algorithm

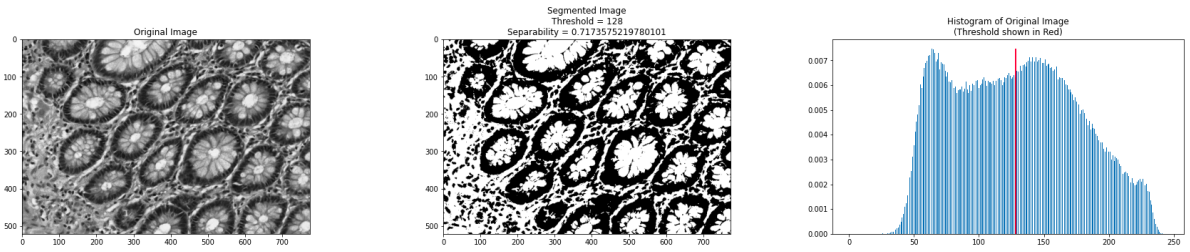


Figure 19: Result of Otsu's Thresholding for Image 1

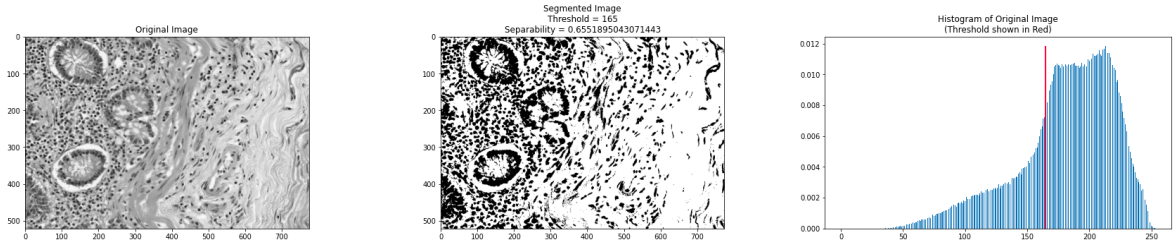


Figure 20: Result of Otsu's Thresholding for Image 2

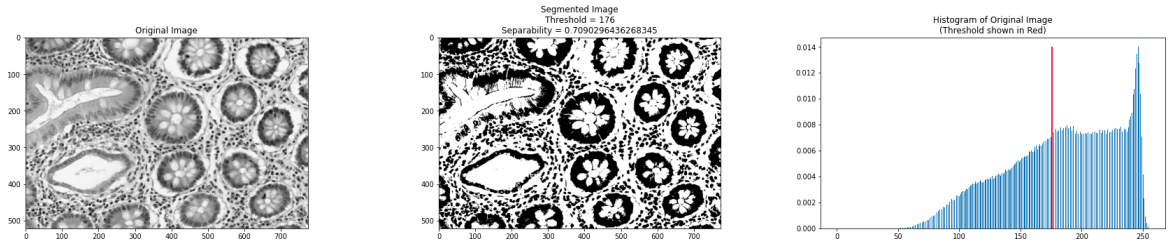


Figure 21: Result of Otsu's Thresholding for Image 3

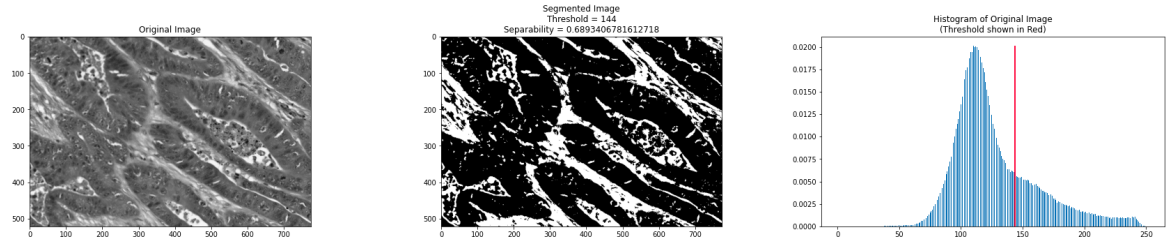


Figure 22: Result of Otsu's Thresholding for Image 4

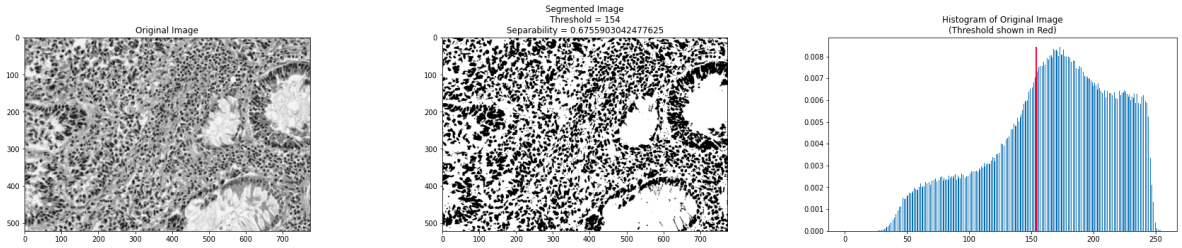


Figure 23: Result of Otsu's Thresholding for Image 5

Observations

- In most cases, the result of Otsu's algorithm was very close to that of the basic global thresholding algorithm.
- Otsu's algorithm converged much faster in most cases, since it is not an iterative algorithm.

- The results provided by Otsu's algorithm are in some cases better than that provided by the Global thresholding algorithm, since instead of trying to find the mean, we try to maximize the separability or between class variance.
- For all five images given, Otsu's algorithm gave good results (separability at least 65%).
- As expected, the separability was low for Images 2 and 5 (since they did not converge well even with the global thresholding algorithm). These images were then analysed with a smoothening mask as done previously. The results are shown in Fig. 24 and Fig. 25.

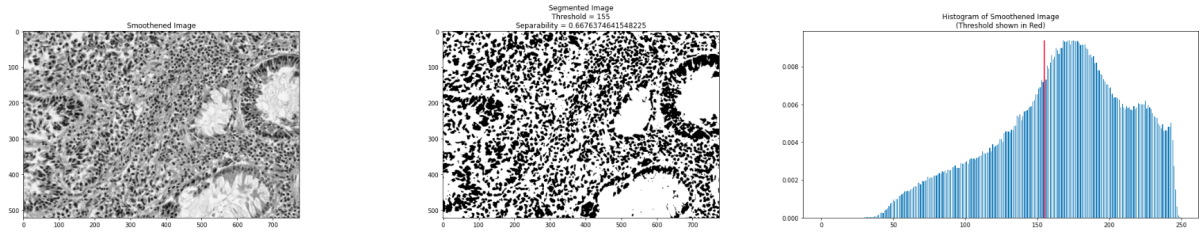


Figure 24: Result of Otsu's Thresholding for Image 5, mean filtered with a window size of 3

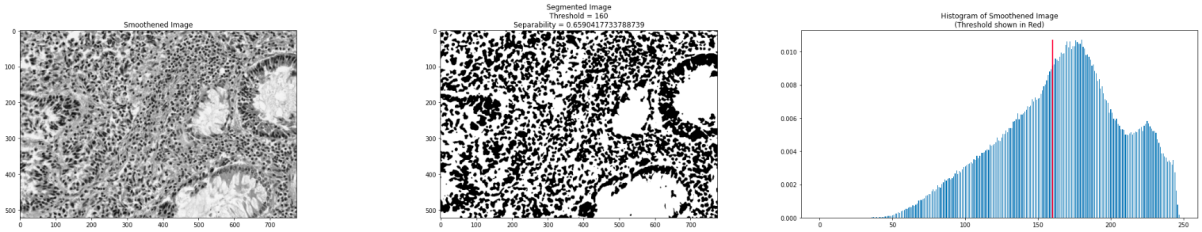


Figure 25: Result of Otsu's Thresholding for Image 5, mean filtered with a window size 5

- We observe that applying a smoothening mask reduces the separability of the image.
- Increasing the window size also reduces the separability of the image.

MULTIPLE THRESHOLDING

Otsu's algorithm can be extended to a number of thresholds, in this case, we will look at two thresholds. The algorithm is the same as the one explained in Fig. 18, with simple modifications to the expressions used to calculated between class variance, to accomodate multiple classes. The results of multiple thresholding are shown in Fig. 26 to Fig. 30.

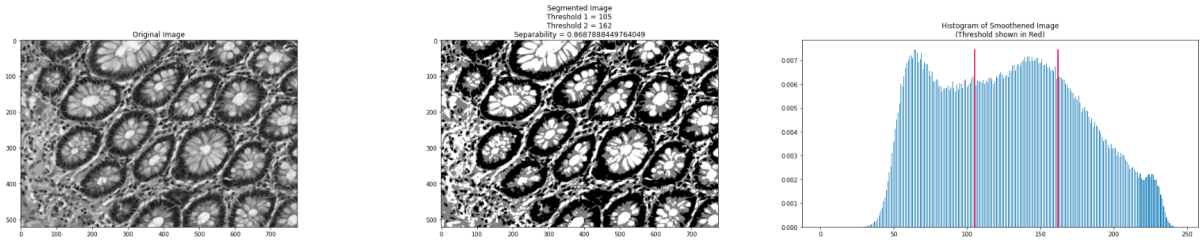


Figure 26: Result of Multiple Thresholding for Image 1

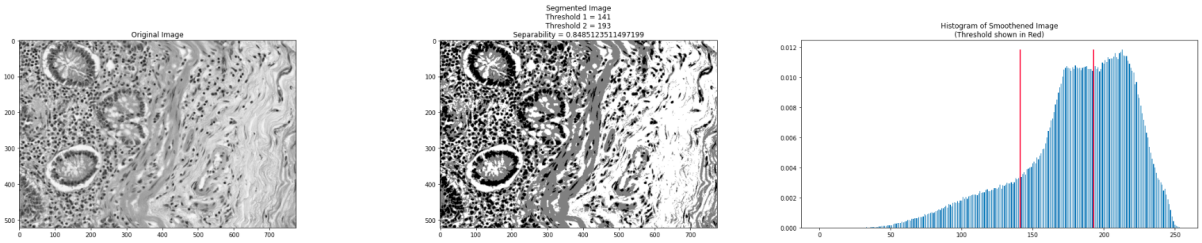


Figure 27: Result of Multiple Thresholding for Image 2

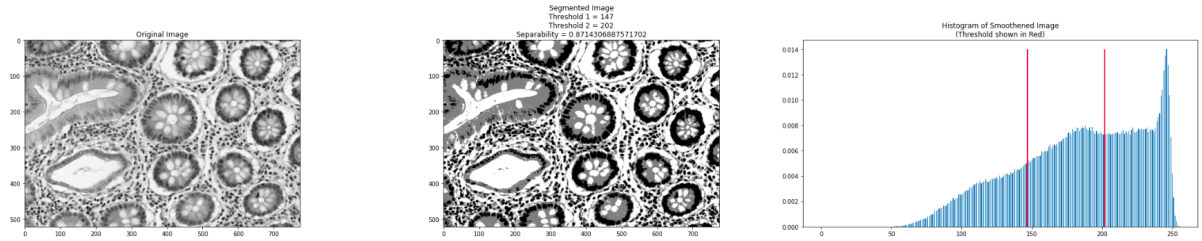


Figure 28: Result of Multiple Thresholding for Image 3

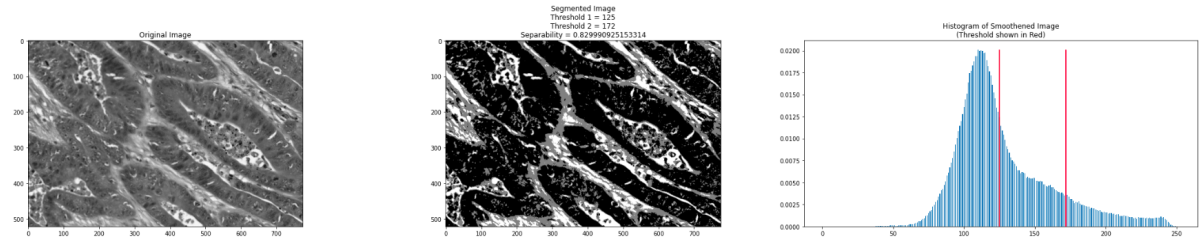


Figure 29: Result of Multiple Thresholding for Image 4

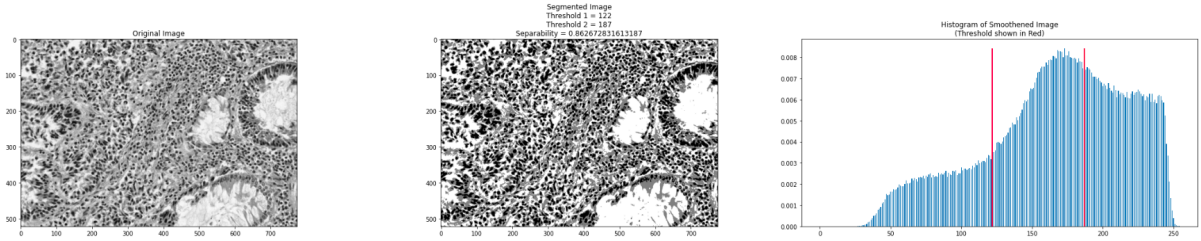


Figure 30: Result of Multiple Thresholding for Image 5

Observations

- The results of multiple thresholding are best in Images 1, 2, 3 and 5, where the histogram clearly shows 3 distinct peaks.
- The algorithm does not work well for Image 4, since the histogram shows a single peak.
- In general, the multiple thresholding algorithm performed better than otsu's single thresholding because of the higher flexibility it offers.
- In images with three peaks on the histogram, even barely visible, the multiple threshold algorithm adapts well to distinguish between the multiple modes on the histogram.

QUESTION 2: SEGMENTATION USING GRAPH CUT ALGORITHMS

Image segmentation problems can be modelled as an energy minimization problem using a graph. In this case, each pixel in the image is represented as a node, along with two additional nodes, a source and sink (that represent the foreground and background respectively). We then define an energy function which we aim to minimize, as follows:

$$E(f) = \sum_{p \in G} D(f(p)) + \sum_{(p,q) \in N} V(f(p), f(q)) \tag{3}$$

where D is the **data cost**, which represented the likelihood of a point to belong in a particular label; V is the **interaction cost**, which represents the smoothing parameter; G is every node in the graph, i.e., every pixel in the image, and N is a list of all the labels. In this implementation, we have used two labels only.

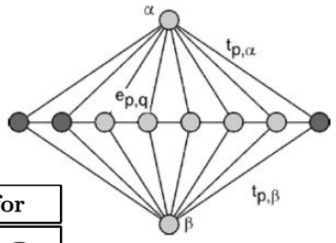
Data cost: The data cost is defined as the likelihood of a point to belong in a particular label. Common choices include the squared error and absolute difference between the pixel intensity and label value. In this implementation, we use the squared error difference.

Interaction Cost: In this case, we use the difference between the two label intensities as the smoothing parameter, i.e., the larger the difference between the intensities, the less the interaction.

$\alpha - \beta$ SWAP

$\alpha - \beta$ swap finds an optimal label swap between all the sites labelled α and those labelled β. The graph and weights for the same are shown in Fig. 31.

Fig. 8.11 Graph of all nodes currently labeled α or β for an α–β-swap move



edge	weight	for
t_p^α	$D_p(\alpha) + \sum_{\substack{q \in \mathcal{N}_p \\ q \notin \mathcal{P}_{\alpha\beta}}} V(\alpha, f_q)$	$p \in \mathcal{P}_{\alpha\beta}$
t_p^β	$D_p(\beta) + \sum_{\substack{q \in \mathcal{N}_p \\ q \notin \mathcal{P}_{\alpha\beta}}} V(\beta, f_q)$	$p \in \mathcal{P}_{\alpha\beta}$
$e_{\{p,q\}}$	$V(\alpha, \beta)$	$\{p,q\} \in \mathcal{N}$ $p,q \in \mathcal{P}_{\alpha\beta}$

Figure 31: Graph and edge weights for $\alpha - \beta$ swap

The algorithm for the same is given below:

```
Start with a labelling f
For each pair of labels {α,β}:
```

Find $\hat{f}$ = argmin $E(f)$ using one $\alpha-\beta$ swap
If $E(\hat{f}) < E(f)$, $f = \hat{f}$
Return f

α EXPANSION

α expansion finds an optimal expansion of the region labelled α into all other regions. The graph and weights for the same are shown in Fig. 32.

Table 8.3 Weights for an α-expansion move

Edge type	Weight	Edge
$t(\mathbf{p}, \bar{\alpha})$	∞	All $\mathbf{p} \in P_\alpha$
$t(\mathbf{p}, \bar{\alpha})$	$\gamma D(f(\mathbf{p}))$	All $\mathbf{p} \notin P_\alpha$
$t(\mathbf{p}, \alpha)$	$\gamma D(\alpha)$	All $\mathbf{p} \in P$
$e(\mathbf{p}, \mathbf{a})$	$V(f(\mathbf{p}), \alpha)$	All $(\mathbf{p}, \mathbf{q}) \in N$, $f(\mathbf{p}) \neq f(\mathbf{q})$
$e(\mathbf{a}, \mathbf{q})$	$V(\alpha, f(\mathbf{q}))$	
$t(\mathbf{a}, \bar{\alpha})$	$V(f(\mathbf{p}), f(\mathbf{q}))$	
$e(\mathbf{p}, \mathbf{q})$	$V(f(\mathbf{p}), \alpha)$	All $(\mathbf{p}, \mathbf{q}) \in N$, $f(\mathbf{p}) \neq f(\mathbf{q})$

Fig. 8.12 The graph for an α-expansion consists of all nodes whether being labeled α or not, plus auxiliary nodes between neighboring nodes with label α and some other label

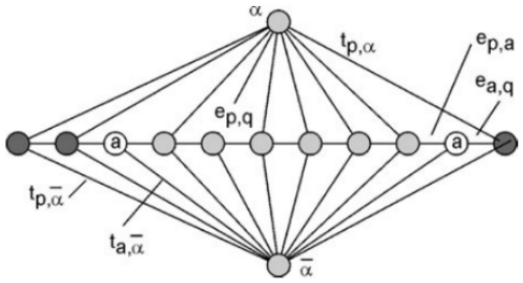


Figure 32: Graph and edge weights for α expansion

The algorithm for the same is given below:

Start with a labelling f
For each labels α:
Find $\hat{f}$ = argmin $E(f)$ using one α expansion
If $E(\hat{f}) < E(f)$, $f = \hat{f}$
Return f

The *pymaxflow* library was used for the implementation of both these functions. The hyper-parameters which need to be tuned are the class labels. The results are shown in Fig. 33 to 37. A three class segmentation was also tried, the results of which are shown in Fig. 38.

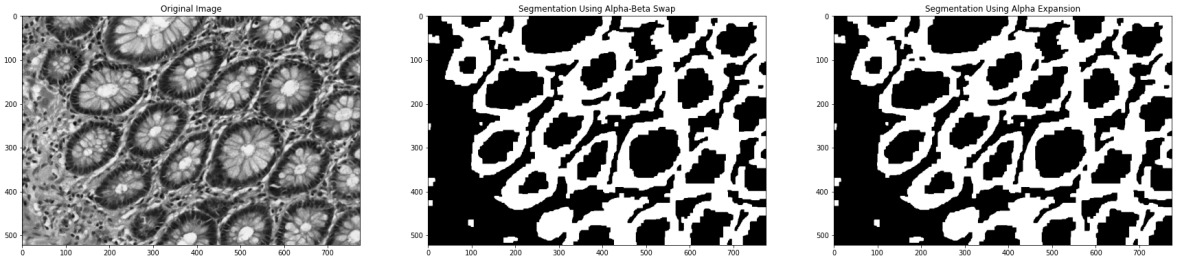


Figure 33: Graph cut segmentation on Image 1 with classes labelled as [0, 0.99]

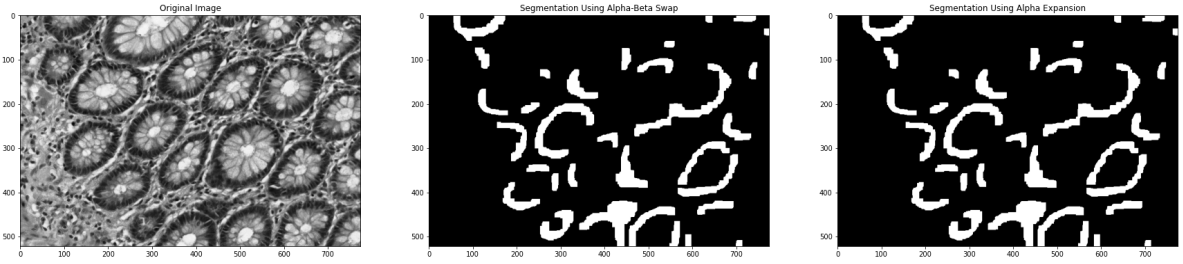


Figure 34: Graph cut segmentation on Image 1 with classes labelled as [0, 0.75]

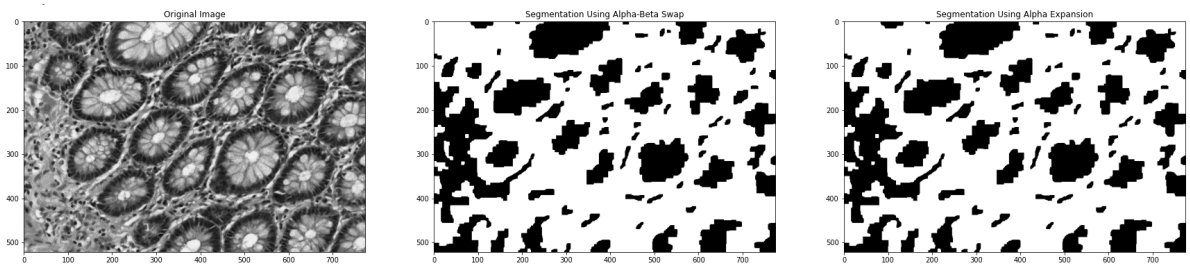


Figure 35: Graph cut segmentation on Image 1 with classes labelled as [0.2, 0.99]

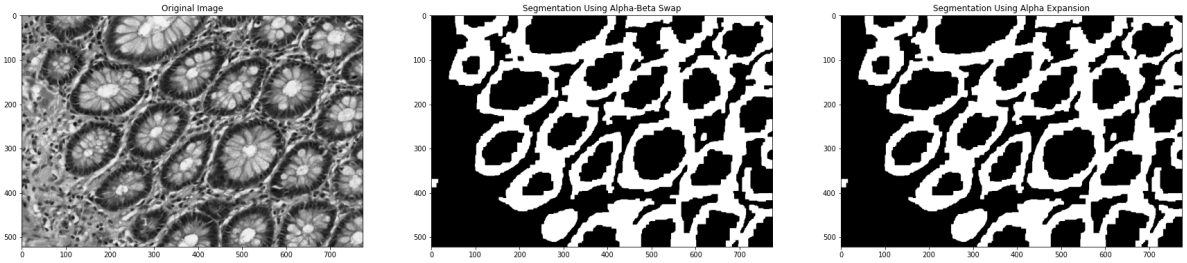


Figure 36: Graph cut segmentation on Image 1 with classes labelled as [0.2, 0.75]

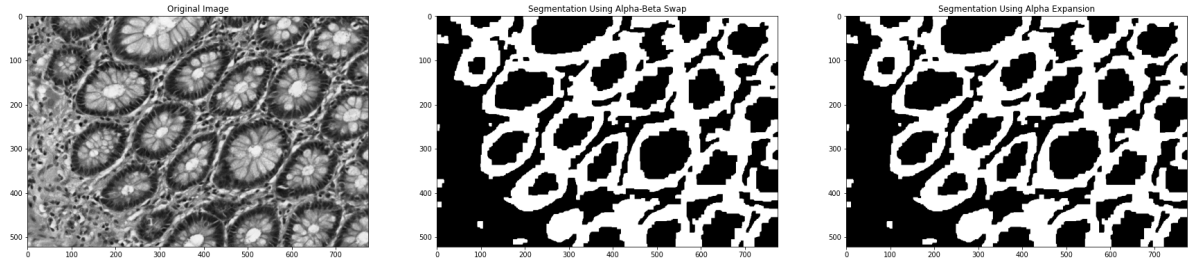


Figure 37: Graph cut segmentation on Image 1 with classes labelled as [0.25, 0.75]

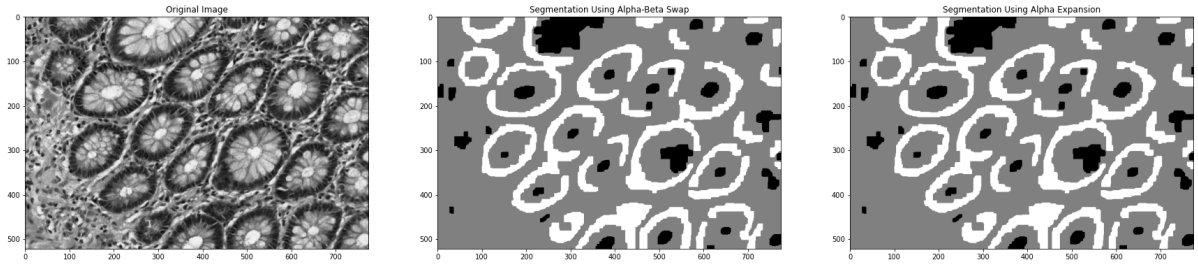


Figure 38: Graph cut segmentation on Image 1 with three classes labelled as [0.2, 0.6, 0.8]

Observations

- Graph cut methods in general are better for image segmentation than thresholding, as they optimize the separation between classes.
- For a two class segmentation, the two approaches ($\alpha - \beta$ swap and α expansion) give similar results, because for a two class problem the algorithm used by expansion is the same as $\alpha - \beta$ swap.
- Increasing the label values lead to more points being classified as background (foreground after the complement is taken), and decreasing the label values lead to more points being classified as foreground (background after complement).
- For 2 classes, the label values 0.25, 0.75 seem to perform well visually. This might be because the values are evenly distributed between 0 and 1.
- Even using a three class classification, the results are highly explainable, having strong correlations with the original image.

The results of these graph cut segmentation techniques for all five input images are given in Fig. 39 to 42. For all of these images, various label values were tried, and the best results are shown here.

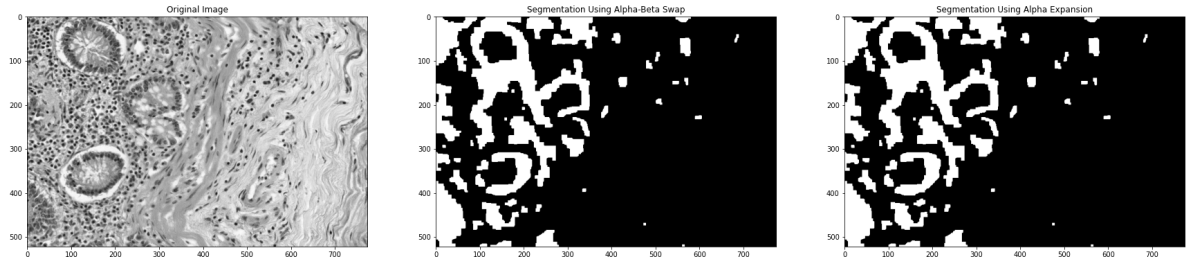


Figure 39: Graph cut segmentation on Image 2 with classes labelled as [0.5, 0.8]

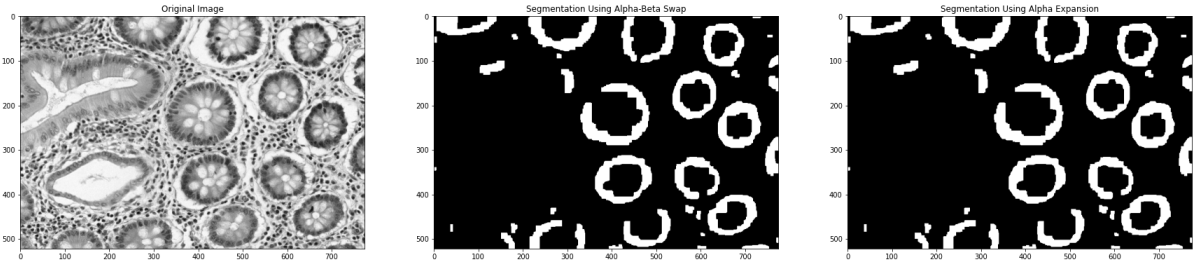


Figure 40: Graph cut segmentation on Image 3 with classes labelled as [0.25, 0.99]

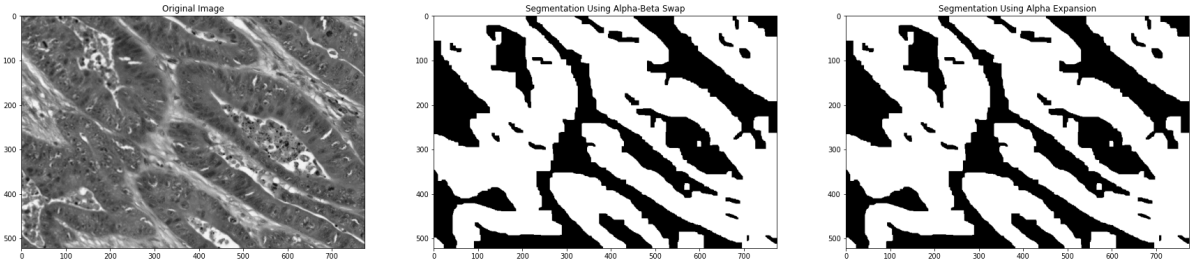


Figure 41: Graph cut segmentation on Image 4 with classes labelled as [0.25, 0.75]

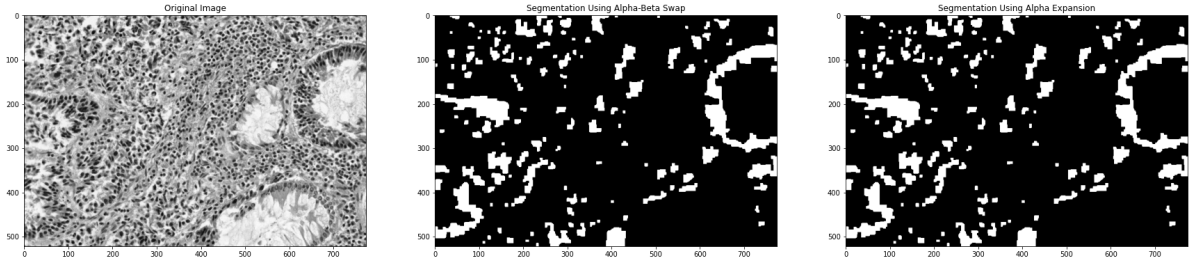


Figure 42: Graph cut segmentation on Image 5 with classes labelled as [0.35, 0.8]

Observations

- The label values have to be adjusted for each image, depending on the composition of the image.
- As observed above, both algorithms give similar results for 2 - class classification.

QUESTION 3 - SEGMENTATION METRICS

There are various segmentation metrics that can be used to quantify the performance of a technique. They are defined as follows:

$$\text{Accuracy} = \frac{\text{Number of correctly classified points}}{\text{Total number of points}} \tag{4}$$

$$= \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

where TP =true positives, TN = true negatives, FP = false positives, FN = false negatives.

$$\text{Dice Coefficient} = \frac{2 \times TP}{(2 \times TP) + FP + FN} \tag{6}$$

$$\text{Jaccard Index} = \frac{TP}{TP + FP + FN} \tag{7}$$

$$\text{Sensitivity} = \frac{TP}{TP + FN} \tag{8}$$

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{9}$$

The results along with the metrics are given for each image.

	Method	Accuracy	Dice Coefficient	Jaccard Index	Sensitivity	Specificity
0	Otsu Thresholding	0.405952	0.454292	0.293906	0.415723	0.391611
1	Alpha Beta Swap	0.620885	0.634398	0.464556	0.553008	0.720518
2	Alpha Expansion	0.620895	0.634436	0.464596	0.553083	0.720432

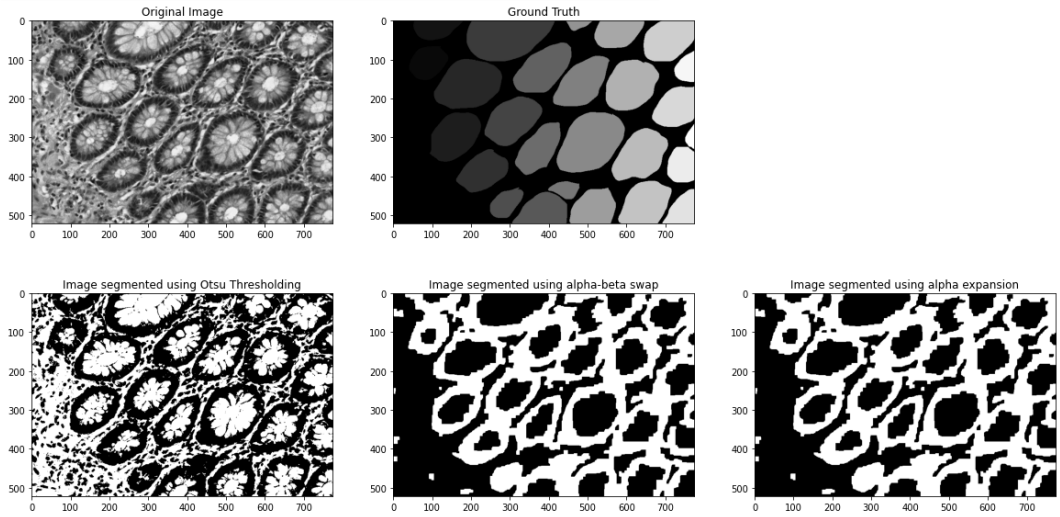


Figure 43: Results for Image 1

	Method	Accuracy	Dice Coefficient	Jaccard Index	Sensitivity	Specificity
0	Otsu Thresholding	0.249351	0.113655	0.060252	0.408536	0.228094
1	Alpha Beta Swap	0.838522	0.466970	0.304606	0.600424	0.870317
2	Alpha Expansion	0.838522	0.466970	0.304606	0.600424	0.870317

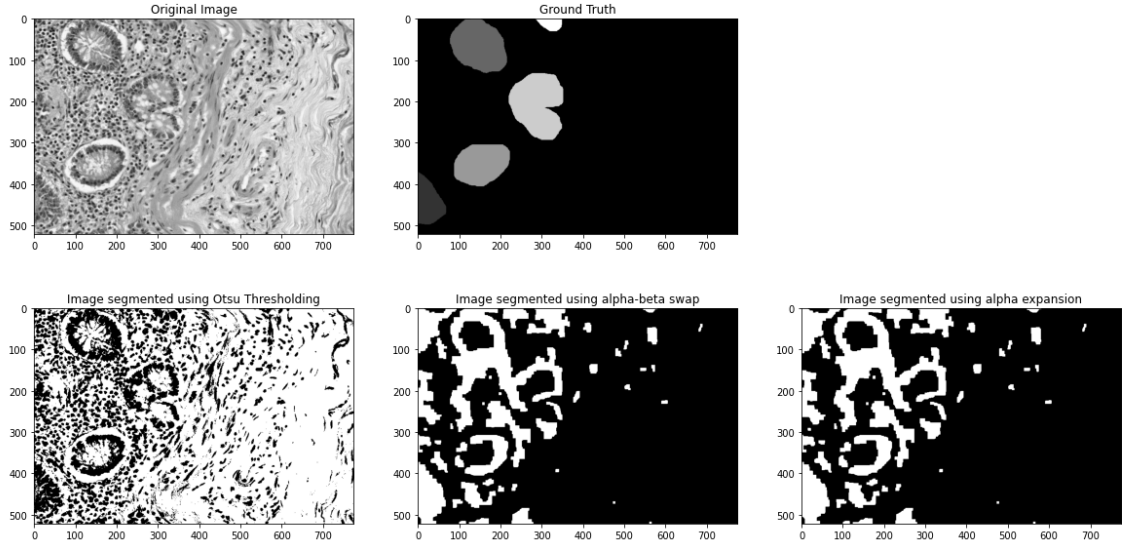


Figure 44: Results for Image 2

	Method	Accuracy	Dice Coefficient	Jaccard Index	Sensitivity	Specificity
0	Otsu Thresholding	0.382249	0.424774	0.269659	0.473720	0.297312
1	Alpha Beta Swap	0.654273	0.457378	0.296494	0.302622	0.980807
2	Alpha Expansion	0.654273	0.457378	0.296494	0.302622	0.980807

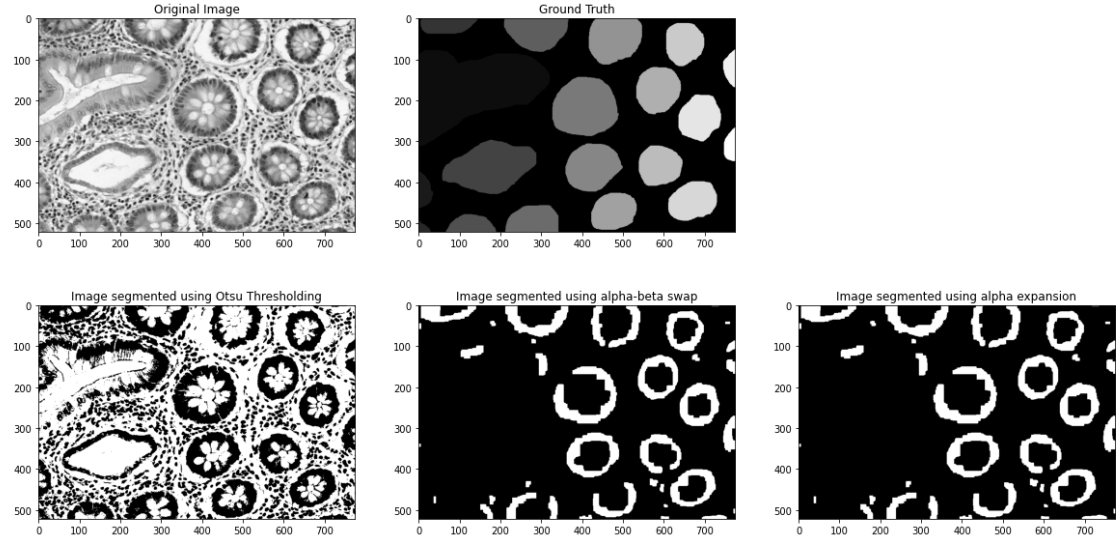


Figure 45: Results for Image 3

	Method	Accuracy	Dice Coefficient	Jaccard Index	Sensitivity	Specificity
0	Otsu Thresholding	0.173450	0.175441	0.096155	0.116123	0.352262
1	Alpha Beta Swap	0.836379	0.883814	0.791816	0.821838	0.881734
2	Alpha Expansion	0.836396	0.883859	0.791888	0.822109	0.880960

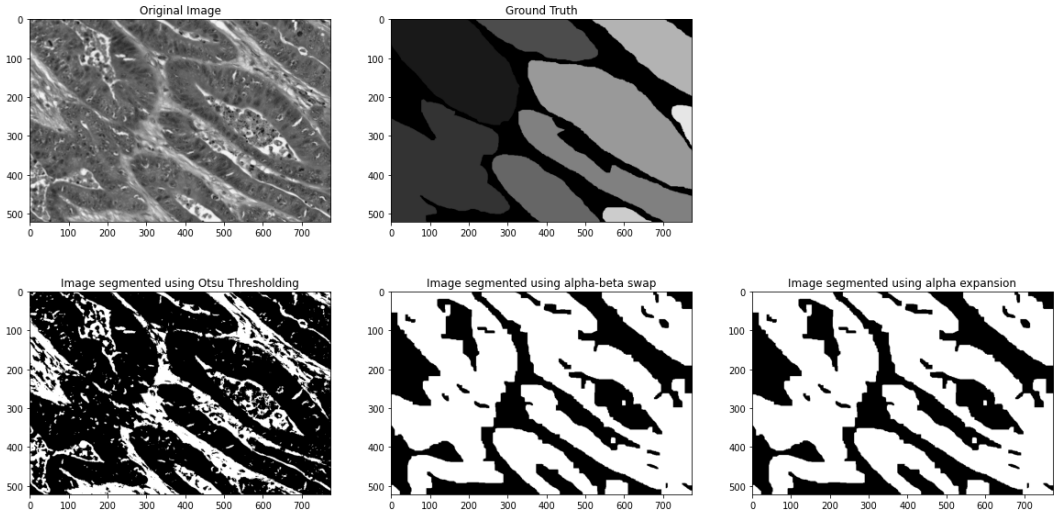


Figure 46: Results for Image 4

	Method	Accuracy	Dice Coefficient	Jaccard Index	Sensitivity	Specificity
0	Otsu Thresholding	0.460475	0.409282	0.257294	0.684200	0.376388
1	Alpha Beta Swap	0.706761	0.292066	0.171005	0.221431	0.889170
2	Alpha Expansion	0.706761	0.292066	0.171005	0.221431	0.889170

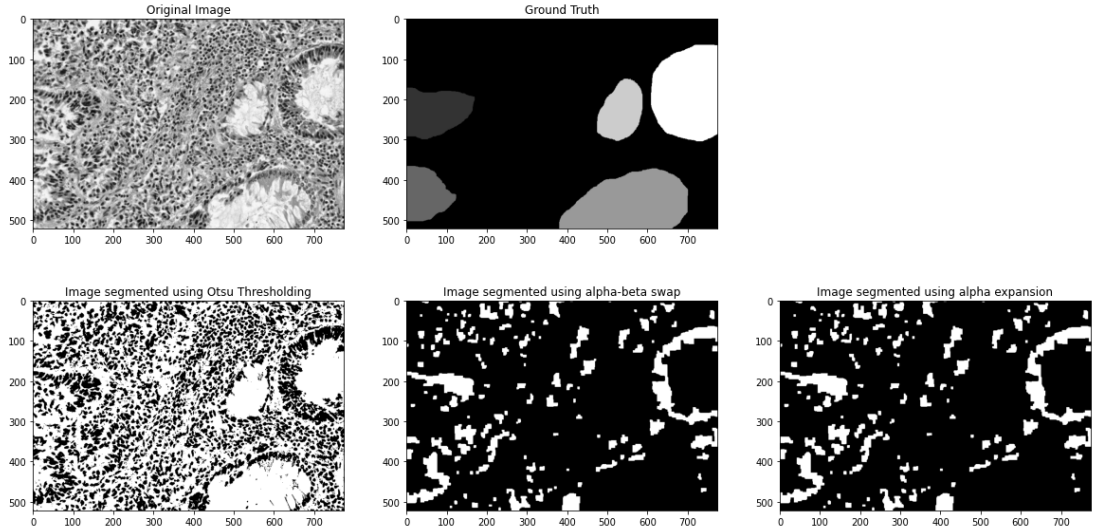


Figure 47: Results for Image 5

Observations

- In most cases, graph cut algorithms perform better than thresholding for all metrics.
- In some cases, increased accuracy can lead to a drop in the other metrics, especially the Dice

and Jaccard coefficients. This is because accuracy takes into account both True Positives and True Negatives, whereas Jaccard and Dice Coefficients are high only if True Positive values are high.

- In most images, the segmented images have a hollow dark region inside the gland, which is because of the bright intensities in the original image. This leads to a significant drop in accuracy and other metrics.
- In most images, the specificity is higher than the sensitivity. This is because of a larger concentration of dark pixels in the segmented image.