Introduction

This report has two sections. The first section is going to introduce the implementation of Task1, which includes the implementation of Otsu's method, HSV thresholding, and the contrast between these two algorithms. The second section is going to show how to implement the median filter, also how to use the two-pass connected component labelling algorithm to count the number of cells. Furthermore, based on the cell numbers, the second section will continue to contrast the number of cells between Otsu's algorithm and HSV thresholding.

Section I Otsu's Method and HSV Thresholding

1.1 Otsu's Algorithm

Based on the code of assignment 1, the first step is reading the image. After that, using the histogram to calculate the frequency of each pixel. The foreground is those pixels that are darker than the white background. Then, Using the total pixel to do the subtraction we can get the number of the pixels of foreground and background. Based on the histogram, we can finally get the total value of the background pixels. This is necessary to get the ***foreground and background total pixel value***. $\boldsymbol{\omega_0(t)}$ represents the proportion of foreground pixels. It is easy to get $\boldsymbol{\omega_0(t)}$, by using the number of foreground pixels dividing the total number of pixels. The same way we can get the $\boldsymbol{\omega_1(t)}$. Here are the two equations **(1)** and **(2).**

$$\boldsymbol{\omega_0(t) = \frac{forground\ pixels}{total\ pixels}} \qquad \textbf{(1)}$$

$$\boldsymbol{\omega_1(t) = \frac{background\ pixels}{total\ pixels}} \qquad \textbf{(2)}$$

The next step is to find the value of $\boldsymbol{\mu_0}$and $\boldsymbol{\mu_1}$. These two values are the mean of the pixel value. $\boldsymbol{\mu_0}$ is the mean of the foreground pixel value. $\boldsymbol{\mu_1}$ is the mean of the background pixel value. Using the following equation **(4)** and **(5)**, it is convenient to get the result.

$$\boldsymbol{\mu_0} = \frac{foreground\ total\ pixel\ value}{foreground\ pixels} \qquad \textbf{(4)}$$

$$\boldsymbol{\mu_1} = \frac{background\ total\ pixel\ value}{background\ pixels} \qquad \textbf{(5)}$$

Then, we can get the intra-class variance, as equation **(6)** showing below.

$$\boldsymbol{\sigma_b^2(t) = \omega_0(t)\omega_1(t)[\mu_0(t) - \mu_1(t)]^2} \qquad \textbf{(6)}$$

When getting all these variables, moving to the update part. Consider the image value is from [0,255], setting the threshold from 0 to 255, then go through each threshold. Updating the number of the **foreground pixel** by **adding the number with the current threshold** in the histogram. Similarly, the total pixel in the graph is fixed, so when increasing the foreground pixel number, also **reducing the background pixel number** to make it balanced. In the same way, calculating **the total value of foreground pixel** by **adding the threshold multiple the value in the histogram**. Same way, reducing the total value of the background pixel. Then, updating the previous variables which are $\boldsymbol{\omega_0(t), \omega_1(t), \mu_0, \mu_1}$. Then updating the intra-class variance. Continue these steps when going through all these thresholds. We can get the **maximum of the intra-class variance** and the **threshold**. By doing the thresholding, we can get the image result.

1.2 HSV thresholding

In the HSV method, we should first read the image by using the OpenCV function. This picture is currently in BGR format. Then, using the convert function change it to HSV space. HSV is a 3-dimensional space. To do the threshold, we should cut the first dimension and doing the reshape operation. After that, we can get the height and width of the space. Also, create a zeros-like matrix to store the result of the threshold is necessary. Then we should find the HSV colour boundary as **Chart 1** shows below.

Chart 1. HSV colour boundary

	black	gray	white	red	orange	green	blue	purple
Hmin	0	0	0	0	11	35	100	255
Hmax	180	180	180	10	25	77	124	155
Smin	0	0	0	43	43	43	43	43
Smax	255	43	30	255	255	255	255	255
Vmin	0	46	221	46	46	46	46	46
Vmax	46	220	255	255	255	255	255	255

The picture of this assignment is cells, which are close to purple. Therefore, starting from purple to do the threshold is much better. Wrapping up these H, S, and V as the range. Cutting each dimension and input the HSV boundary value to do the threshold. If all H, S, V is in the range of threshold, we should change the pixel value to 255 which is white. Otherwise, set the value as 0 which is black.

1.3 Contrast these two methods

Here is the picture result of **c1** which is provided. **Figure 1** is shown below.

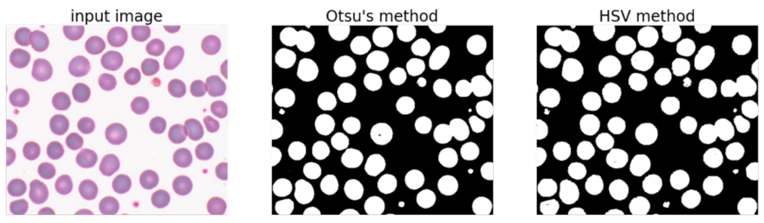

Figure 1. c1 output

Here is the picture result of **c2**. **Figure 2** is shown below.

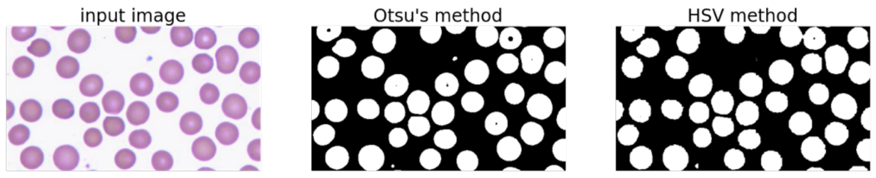

Figure 2. c2 output

Here is the picture result of **c3**. **Figure 3** is shown below.

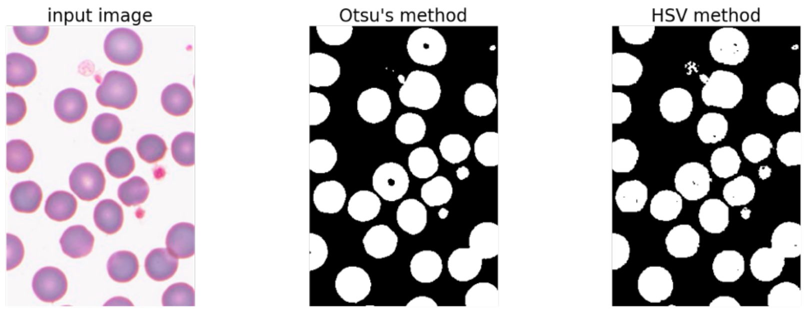

Figure 3. c3 output

Here is the picture result of **c4** which is provided. **Figure 4** is shown below.

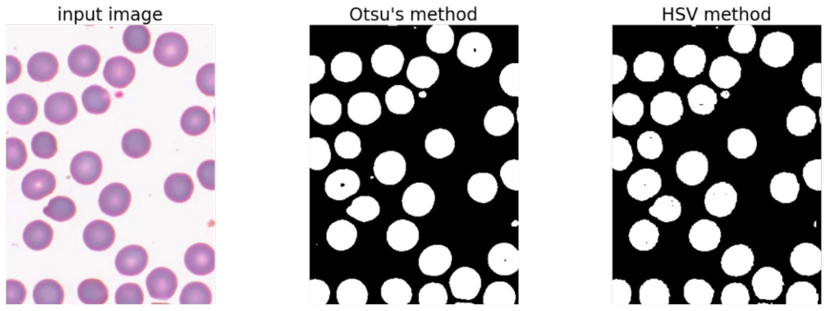

Figure 4. c4 output

From these pictures, HSV Methods' output can still find some noise in the cell. However, Otsu's Methods' output is cleaner. During the iteration, Otsu's method can finally get the best threshold and variance. Also, setting an accurate boundary can improve the performance of the HSV Method. The Otsu's Method also has disadvantages. When the threshold and variance approach the best result infinitely, the centre of the cell will be regarded as the background. Hence, we can find the black dot in the centre.

Section II Median filter and Connected Component Labelling

2.1 Median filter

In this assignment, using a 3*3 median filter to select the data from the picture. Using for-loop selecting the centre of the matrix. If the neighbour of the centre exists, put these pixel values together. When looping all these pixels in the 3*3 matrix and getting all these values, using the median function provided by NumPy can get the median value. After that changing a new centre, until all the pixels in the picture were selected. Here is the **Charter 2**. This is an example of using [-1,0,1] to implement the filter window.

Charter 2. The position matrix

(-1,-1)	(-1,0)	(-1,1)
(0,-1)	(0,0)	(0,1)
(+1,-1)	(+1,0)	(+1,+1)

2.2 Connected Component Labelling algorithm

This algorithm has two steps. The first step is called the first pass, which is labelling the pixel in the picture. The second step is to do the statics of the label. This algorithm is based on the foreground pixels. Here are the rules of how to label pixels in the first pass. Here is the first situation, 0 means the background, X means the current position is not detected. Consider the top of the centre and the left of the centre is 0, therefore, the centre should label a new value which is 2.

	0	1
0	?	X
	X	

Here is the second situation, X means the current position is not detected. Consider the top of the centre is 4 and the left of the centre is 1. In this situation, the centre here should be 1 which is the minimum between 1 and 4.

	4	
1	?	X
	X	

Based on the label rules, by looping the image we can label these pixels. The position selection is similar to **Charter 2**. During the looping, we should set up a list to append the labels. After the looping, if the list is empty, create the set in the dictionary, and move the label value to the next one. The dictionary key is a label, the value is a set that is used to store the labels. This kind of design is convenient to do the merger of the dictionary. If the list is not empty, we should get the min value of the list, which is the label of the current position. After that, updating the dictionary and doing the merger of the dictionary. The operation of the dictionary can also use a tree structure to implement.

In the second pass, we should set another dictionary to get the minimum value at one time. This can help us save more time to get the minimum one by one. And setting the minimum value in the matrix. Then doing the statics, create a set (no sequence, faster) and counting the number of the foreground pixels. Check the current set with the previous label, if they are the same, means there is an object. Then getting the length of the object as well as checking the length is in the range of minimum area and overlap area. If the length is less than the minimum area, the system will regard this area as noise. If the length is larger than the overlap area, the system will add another cell.
Here is the result of Task2. To emphasize the contrast between, putting these 2 pictures together. In the uploading package can find the standard output of Task2.
Figure 5 is the result of **c1** in Task2.

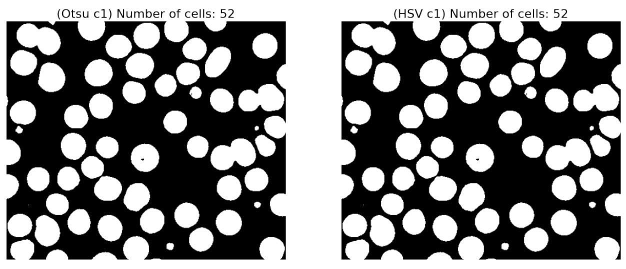

Figure 5. c1 cell number

Figure 6 is the result of **c2** in Task2.

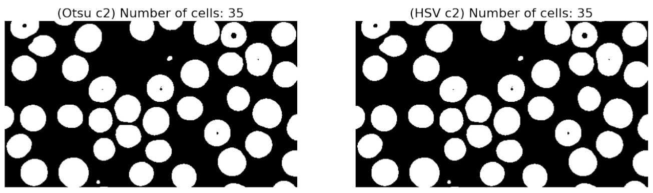

Figure 6. c2 cell number

Figure 7 is the result of **c3** in Task2.

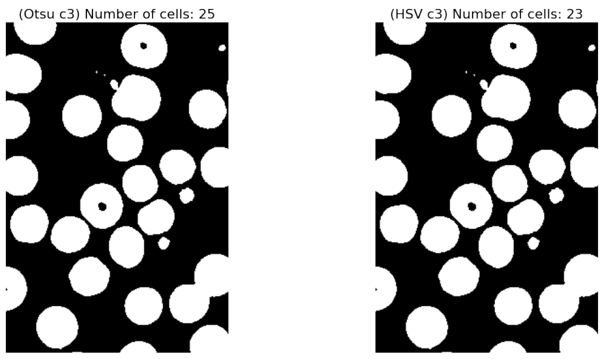

Figure 7. c3 cell number

Figure 8 is the result of **c4** in Task2.

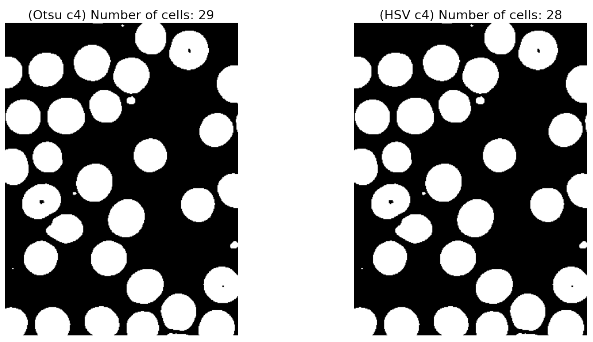

Figure 8. c4 cell number

The cell number result is based on the Task 1 output. Therefore, we can find Otsu's output is more accurate in checking the cell numbers. Furthermore, the median filter has a good performance in eliminating the noise in HSV, which makes the HSV clean.