TDS3561 Visual Information Processing

Trimester 1, 2022/2023

Assignment (20%)

Lecturer: Dr. Loh Yuen Peng

Prepared by:

Name	Student ID	Phone Number
Lim Wei Jie	1181103501	012-5681547

Abstract

Image segmentation is a method that is very popular in medical image analysis. Medical professionals use the techniques of image segmentation to identify their Region of Interest for further diagnosis. By using image segmentation techniques for diagnosis, it achieves fast and accurate results compared with using manual way to diagnose. So, an image segmentation algorithm is built in this assignment by using some methods like denoising the image, detecting the color range of the white blood cells and red blood cells, segmenting out the parts by using morphology transformation, etc. An evaluation matrix is provided by the lecturer to evaluate the accuracy rate of segmentation on the proposed method. The evaluation table shows that there is only an average error rate of 0.14, both average precision and average recall are above 0.85 and the average Intersection of Union(IoU) achieves 0.77. This result is considered well but I guess still can improve in the future. We have provided some suggestions for improvement in the future by having better segmentation and also classification between the cell parts using machine learning methods.

Introduction

Image contains rich information for human vision. A human can interpret the information that appears in the image but human vision is difficult to get accurate information if there is too much tiny information and need more time to interpret. So image segmentation would help us to increase the efficiency, and workload for detection. Image segmentation is the process of partitioning a digital image into multiple image segments. Image segmentation is typically used to locate objects and boundaries such as lines, curves, etc. In these years, image segmentation is widely used in various fields. Some of the applications examples are medical image analysis, computer vision for autonomous vehicles, face recognition and detection, etc. As for medial imaging analysis, the use of image segmentation can help pathologists to save more time in identifying the region of interest(ROI) and efficiency to detect and diagnosing the affected cells. There are a lot of ways to do segmentation in the images. An assignment has been done by us to test our basic skills in image processing by trying to build an image segmentation algorithm.

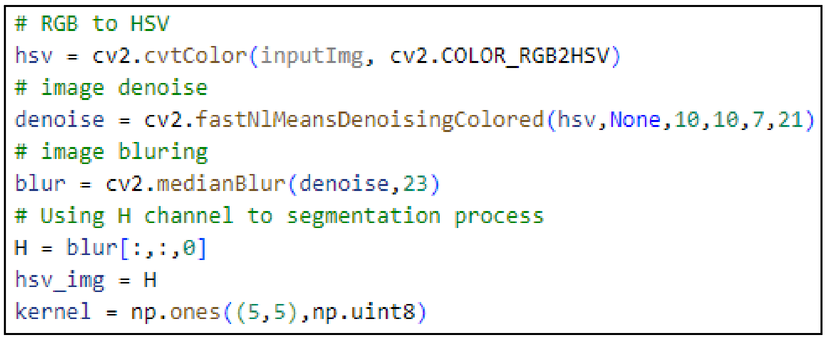
We have tried to create an image segmentation algorithm in this assignment. In this assignment, there will some trying attempts have been done by denoising and blurring the original image for a smoother image, using color range to find all the red blood cells and the white blood cells, and merging the images to compare the similarity with ground truth provided by the lecturer. Next, the image segmentation algorithm will be tested by using 50 input images along with the corresponding ground truth from the lecturer. 50 input images yet tested on the evaluateSegment.py provided by the lecturer to display the average result on error, precision, recall, and Intersection of Union(IoU). The proposed algorithm and result will be shown in the next section.

Methods

In this project, we decided to do the segmentation region part by part. The segmentation process is divided into two parts, segmentation of the White Blood Cell and segmentation of the Red Blood Cell. After the segmentation of White Blood Cell and Red Blood Cell, both segmentation will be combined and the output is formed. All the detailed steps to achieve segmentation will be explained.

Tuning Input Images

In this section, we change the input image color space from RGB to HSV. The reason we changed the input image from RGB to HSV is that the output result for the input image only displays the range of blue and purple. It provides better vision for human eyes to detect segmentation parts. After the color conversion is done, we denoise the input image using ***cv.fastNlMeansDenoisingColored*** technique from OpenCV. This method is Non-Local Means Denoising. It takes more time to process compared to blurring techniques. We also use the median blur technique to further blur the image for a better view. Next, we separate the 3 channels of HSV to observe which channel we can use for better segmentation. We choose to use H (hue) channel for further segmentation process because the figure below can show that the H channel has clear images compared with other channels.



```
# RGB to HSV
hsv = cv2.cvtColor(inputImg, cv2.COLOR_RGB2HSV)
# image denoise
denoise = cv2.fastNlMeansDenoisingColored(hsv,None,10,10,7,21)
# image bluring
blur = cv2.medianBlur(denoise,23)
# Using H channel to segmentation process
H = blur[:,:,0]
hsv_img = H
kernel = np.ones((5,5),np.uint8)
```

Figure 1.1: Code for tuning input images

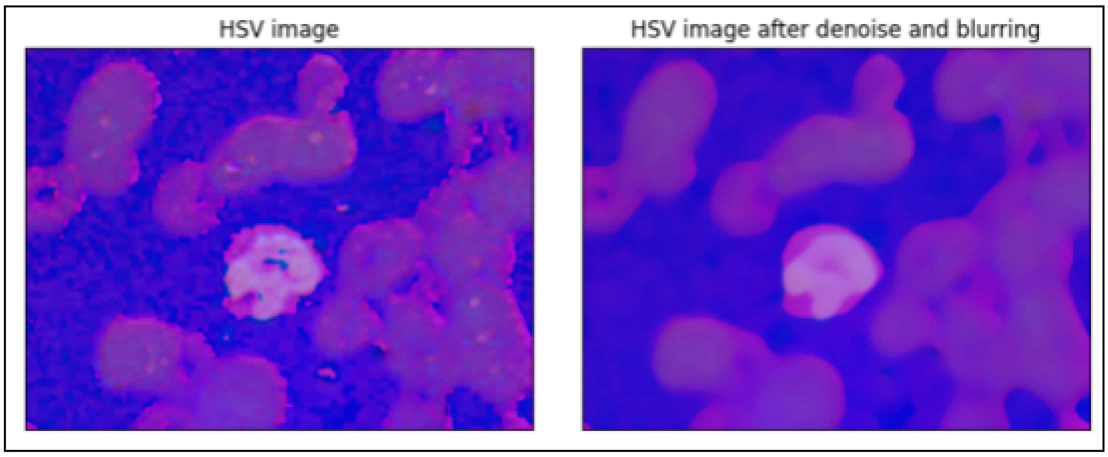


Figure 1.2: Original HSV Image and HSV image after denoise and blurring

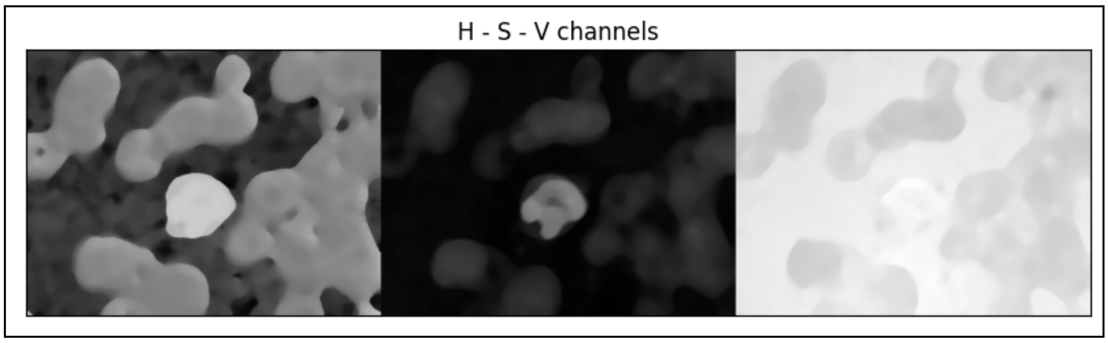
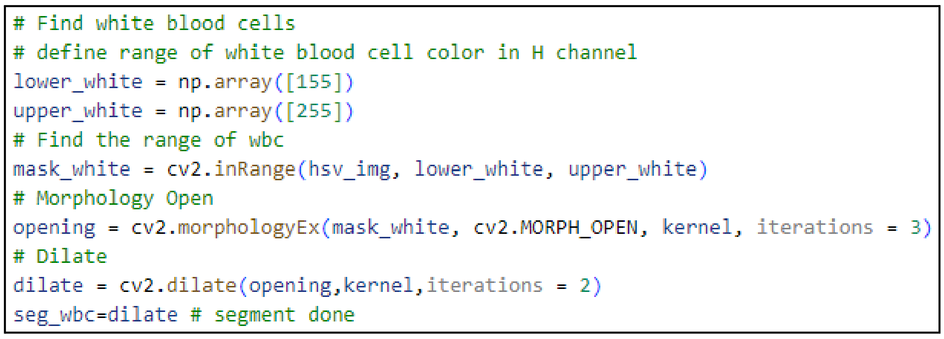


Figure 1.3: H-S-V channel images

Segmentation White Blood Cells

To segment out White Blood Cells, first we need to define a range for the white blood cell color that we intend to detect in HSV. Next, a threshold for the HSV image using the white color range that we specified earlier. We will be using the cv2.inRange technique in OpenCV that can define a lower and upper threshold. After the threshold was done, the morphological open process was used to remove the noise in the image. Morphological open involves erosion followed by dilation in the outer surface of the image. After the morphological open process, dilation is used again to generate a better segmentation white blood cells that almost matches the groundtruth of the input image. Figure 2.1 shows the result of the segmentation of white blood cells.



```
# Find white blood cells
# define range of white blood cell color in H channel
lower_white = np.array([155])
upper_white = np.array([255])
# Find the range of wbc
mask_white = cv2.inRange(hsv_img, lower_white, upper_white)
# Morphology Open
opening = cv2.morphologyEx(mask_white, cv2.MORPH_OPEN, kernel, iterations = 3)
# Dilate
dilate = cv2.dilate(opening,kernel,iterations = 2)
seg_wbc=dilate # segment done
```

Figure 2.1: Code for segmentation of white blood cells

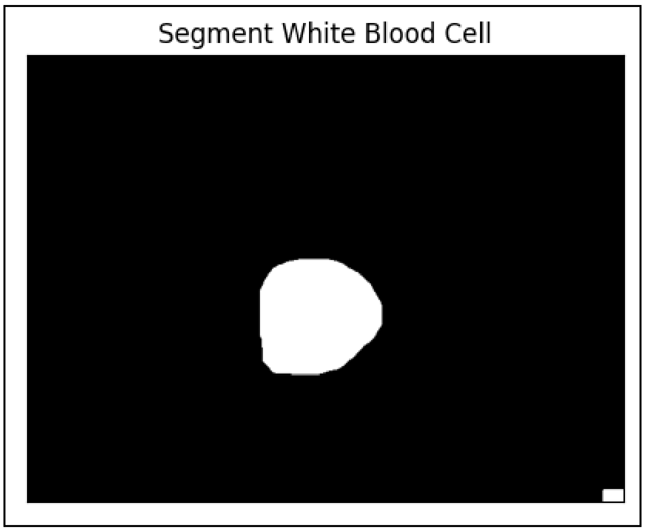
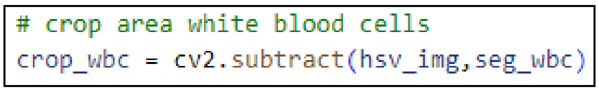


Figure 2.2: Result of the segmentation of white blood cells.

Segmentation Red Blood Cells

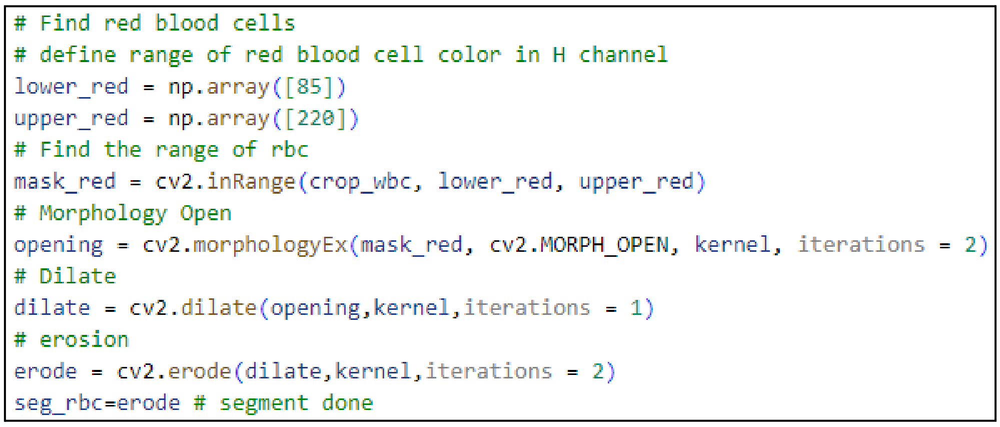
We need to subtract the segmentation object of white blood cells before we start the process of segmentation of red blood cells.



```
# crop area white blood cells
crop_wbc = cv2.subtract(hsv_img,seg_wbc)
```

Figure 3.1: Code for subtracting the segmentation object of white blood cells

There is a similar process to the segmentation of white blood cells, first, we segment out Red Blood Cells and define a range for the red blood cell color that we intend to detect in HSV. Next, a threshold for the HSV image using the gray color range that we specified earlier. We will be using the ***cv2.inRange*** technique in OpenCV that can define a lower and upper threshold. After the threshold was done, the morphological open process was used to remove the noise in the image. Morphological open involves erosion followed by dilation in the outer surface of the image. After the morphological open process, dilation following the erosion process is used again to generate a better segmentation white blood cells that almost matches the groundtruth of the input image. Figure 3.3 shows the result of the segmentation of red blood cells.



```
# Find red blood cells
# define range of red blood cell color in H channel
lower_red = np.array([85])
upper_red = np.array([220])
# Find the range of rbc
mask_red = cv2.inRange(crop_wbc, lower_red, upper_red)
# Morphology Open
opening = cv2.morphologyEx(mask_red, cv2.MORPH_OPEN, kernel, iterations = 2)
# Dilate
dilate = cv2.dilate(opening,kernel,iterations = 1)
# erosion
erode = cv2.erode(dilate,kernel,iterations = 2)
seg_rbc=erode # segment done
```

Figure 3.2: Code for segmentation of red blood cells

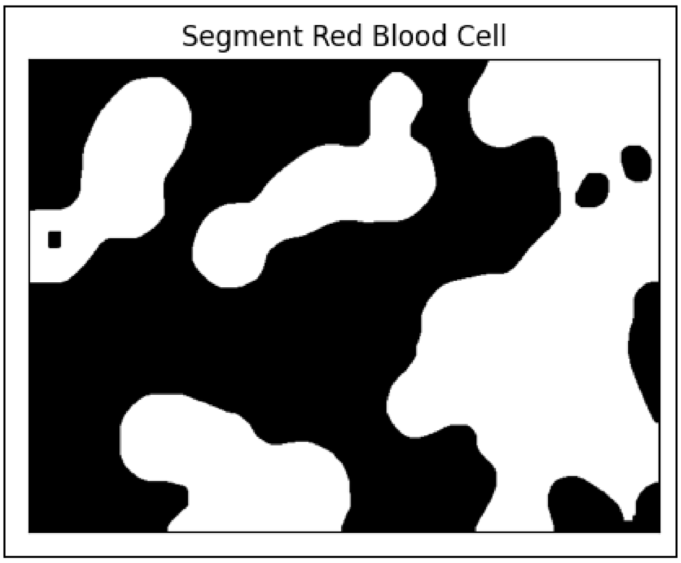
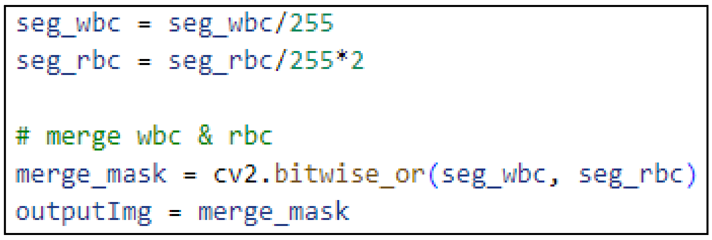


Figure 3.3: Result of the segmentation of red blood cells.

Merging the segmentation images

Both of the segmentation images will merge together using the ***cv2.bitwise_or*** function. The segmentation of white blood cells and segmentation of red blood cells will be merged in one image and also as our output image for evaluation. Figure 4.2 shows the result of the segmentation process after merging.



```
seg_wbc = seg_wbc/255
seg_rbc = seg_rbc/255*2

# merge wbc & rbc
merge_mask = cv2.bitwise_or(seg_wbc, seg_rbc)
outputImg = merge_mask
```

Figure 4.1: Merging both segmentation images into one image

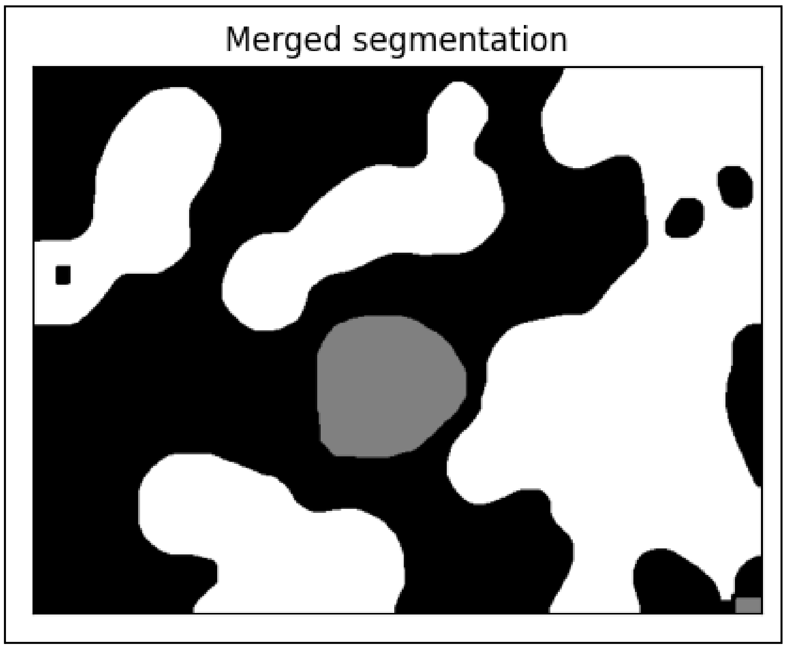


Figure 4.2: Result of the segmentation process after merge.

Result & Analysis

The result of this assignment is evaluated by using evaluateSegment.py which is provided by the lecturer. During the evaluation, 50 images and 50 groundtruth images were taken as the test images to evaluate the proposed method in this assignment. The result of the evaluation will be displayed in a table that consists of 4 columns: Error, Precision, recall, and Intersection of Union(IoU). The second results of the evaluation will be displayed in a table that measures the accuracy of the segmentation parts of the red blood cells, white blood cells, and background.

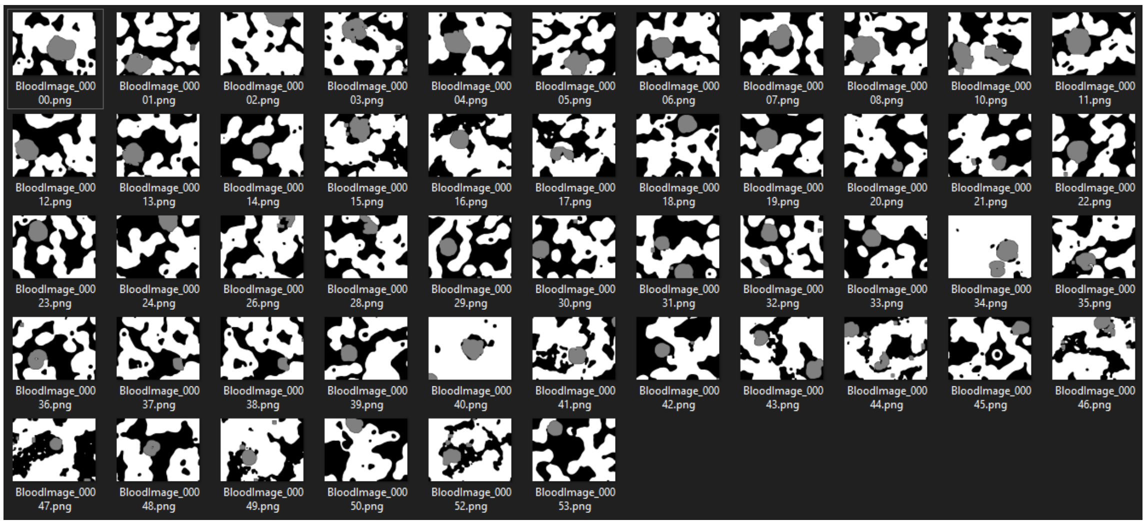
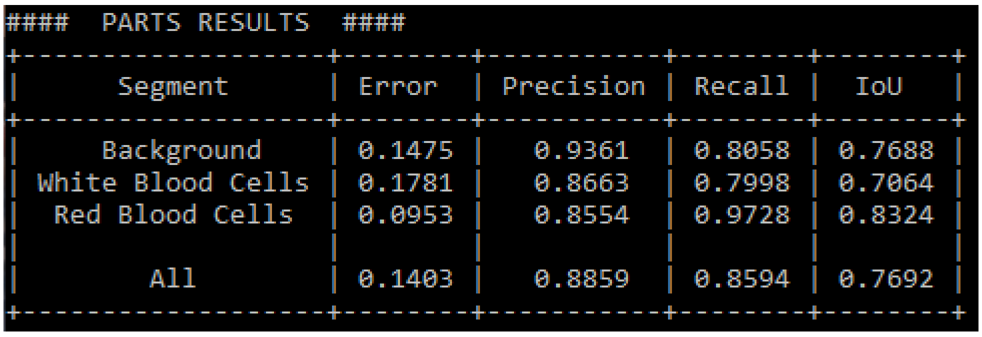


Figure 5.1: The output images of the segmentation

Evaluation Table based on the segmentation parts results

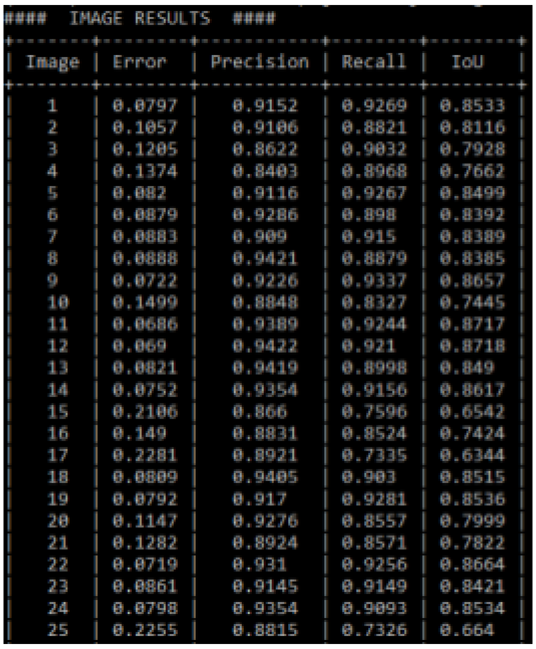


```
####  PARTS RESULTS  ####
+-------------------+--------+-----------+--------+--------+
|      Segment      | Error  | Precision | Recall |  IoU   |
+-------------------+--------+-----------+--------+--------+
|     Background    | 0.1475 |   0.9361  | 0.8058 | 0.7688 |
| White Blood Cells | 0.1781 |   0.8663  | 0.7998 | 0.7064 |
|  Red Blood Cells  | 0.0953 |   0.8554  | 0.9728 | 0.8324 |
|                   |        |           |        |        |
|        All        | 0.1403 |   0.8859  | 0.8594 | 0.7692 |
+-------------------+--------+-----------+--------+--------+
```

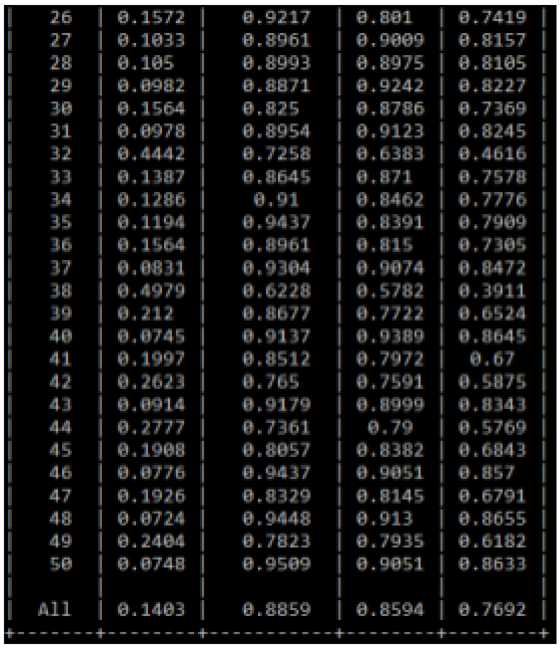
Figure 5.2: Table of results based on the segmentation parts

In this table, the results of the segmentation of background, white blood cells, and also red blood cells are shown. Based on the error column, the segment of red blood cells have the lowest error rate which is 0.0953 and followed by background and white blood cells which are 0.1475 and 0.1781. Next, the background achieves the highest precision which is 0.9361. The precision of white blood cells and red blood cells achieved 0.8663 and 0.8554. Both precisions of the cells are average with only a different 0.01. On the other hand, red blood cells achieve the highest recall rate which is 0.9728, and white blood cells have a recall rate of 0.7998. The recall rate for the background is 0.8058. IoU columns provide the overall performance of the segmentation. Red blood cells have the highest IoU rate which is 0.8324 and followed by the background and white blood cells with 0.7688 and 0.7064. Based on the results, we can see that the overall results of the red blood cells have done segment at the most accurate rate, and the average result on the background and white blood cells is also not that bad.

Evaluation Table based on the segmentation parts results



```
####  IMAGE RESULTS  ####
+-------+--------+-----------+--------+--------+
| Image | Error  | Precision | Recall |  IoU   |
+-------+--------+-----------+--------+--------+
|   1   | 0.0797 |   0.9152  | 0.9269 | 0.8533 |
|   2   | 0.1057 |   0.9106  | 0.8821 | 0.8116 |
|   3   | 0.1205 |   0.8622  | 0.9032 | 0.7928 |
|   4   | 0.1374 |   0.8403  | 0.8968 | 0.7662 |
|   5   | 0.082  |   0.9116  | 0.9267 | 0.8499 |
|   6   | 0.0879 |   0.9286  | 0.898  | 0.8392 |
|   7   | 0.0883 |   0.909   | 0.915  | 0.8389 |
|   8   | 0.0888 |   0.9421  | 0.8879 | 0.8385 |
|   9   | 0.0722 |   0.9226  | 0.9337 | 0.8657 |
|  10   | 0.1499 |   0.8848  | 0.8327 | 0.7445 |
|  11   | 0.0686 |   0.9389  | 0.9244 | 0.8717 |
|  12   | 0.069  |   0.9422  | 0.921  | 0.8718 |
|  13   | 0.0821 |   0.9419  | 0.8998 | 0.849  |
|  14   | 0.0752 |   0.9354  | 0.9156 | 0.8617 |
|  15   | 0.2106 |   0.866   | 0.7596 | 0.6542 |
|  16   | 0.149  |   0.8831  | 0.8524 | 0.7424 |
|  17   | 0.2281 |   0.8921  | 0.7335 | 0.6344 |
|  18   | 0.0809 |   0.9405  | 0.903  | 0.8515 |
|  19   | 0.0792 |   0.917   | 0.9281 | 0.8536 |
|  20   | 0.1147 |   0.9276  | 0.8557 | 0.7999 |
|  21   | 0.1282 |   0.8924  | 0.8571 | 0.7822 |
|  22   | 0.0719 |   0.931   | 0.9256 | 0.8664 |
|  23   | 0.0861 |   0.9145  | 0.9149 | 0.8421 |
|  24   | 0.0798 |   0.9354  | 0.9093 | 0.8534 |
|  25   | 0.2255 |   0.8815  | 0.7326 | 0.664  |
```



```
|  26   | 0.1572 |   0.9217  | 0.881  | 0.7419 |
|  27   | 0.1033 |   0.8961  | 0.9009 | 0.8157 |
|  28   | 0.105  |   0.8993  | 0.8975 | 0.8105 |
|  29   | 0.0982 |   0.8871  | 0.9242 | 0.8227 |
|  30   | 0.1564 |   0.825   | 0.8786 | 0.7369 |
|  31   | 0.0978 |   0.8954  | 0.9123 | 0.8245 |
|  32   | 0.4442 |   0.7258  | 0.6383 | 0.4616 |
|  33   | 0.1387 |   0.8645  | 0.871  | 0.7578 |
|  34   | 0.1286 |    0.91   | 0.8462 | 0.7776 |
|  35   | 0.1194 |   0.9437  | 0.8391 | 0.7909 |
|  36   | 0.1564 |   0.8961  | 0.815  | 0.7305 |
|  37   | 0.0831 |   0.9304  | 0.9074 | 0.8472 |
|  38   | 0.4979 |   0.6228  | 0.5782 | 0.3911 |
|  39   | 0.212  |   0.8677  | 0.7722 | 0.6524 |
|  40   | 0.0745 |   0.9137  | 0.9389 | 0.8645 |
|  41   | 0.1997 |   0.8512  | 0.7972 |  0.67  |
|  42   | 0.2623 |   0.765   | 0.7591 | 0.5875 |
|  43   | 0.0914 |   0.9179  | 0.8999 | 0.8343 |
|  44   | 0.2777 |   0.7361  |  0.79  | 0.5769 |
|  45   | 0.1908 |   0.8057  | 0.8382 | 0.6843 |
|  46   | 0.0776 |   0.9437  | 0.9051 | 0.857  |
|  47   | 0.1926 |   0.8329  | 0.8145 | 0.6791 |
|  48   | 0.0724 |   0.9448  | 0.913  | 0.8655 |
|  49   | 0.2404 |   0.7823  | 0.7935 | 0.6182 |
|  50   | 0.0748 |   0.9509  | 0.9051 | 0.8633 |
|       |        |           |        |        |
|  All  | 0.1403 |   0.8859  | 0.8594 | 0.7692 |
+-------+--------+-----------+--------+--------+
```

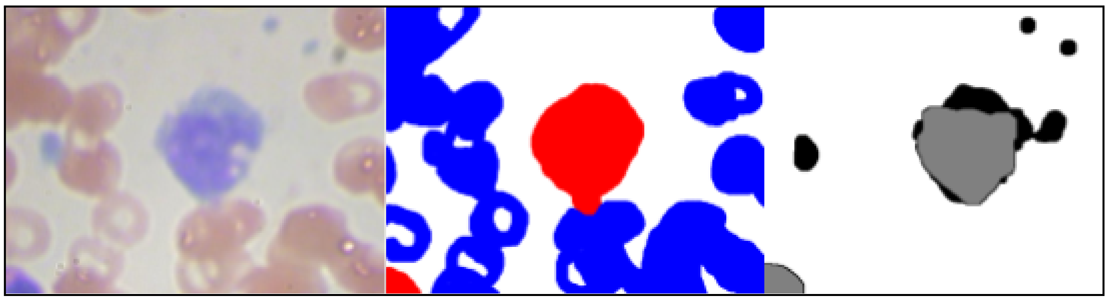
Figure 5.3: Table of results based on every picture

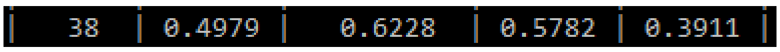
Based on the table results, we can clearly see that the average error of the algorithm achieves 0.1403. The average precision achieves 0.8859 and the average recall rate also achieves 0.8594. The overall IoU rate achieves 0.7692 which shows the results seem quite good. I believe that if there are still other methods that can enhance the images, the IoU rate can be higher and have better results.

Comparing the best and worst results of images

In this section, I will show the best result image and the worst result image based on the evaluated results. Based on my evaluation results, the image on 38th has the worst IoU rate and the images on the 12th have the best IoU rate among the others. Below will discuss the factors that caused the result.

Worst Result Image





```
|    38    | 0.4979 |     0.6228    | 0.5782 | 0.3911 |
```

Figure 6.1: Original image, Groundtruth image, Result image, and evaluation result of the 38th image

From the observation of the resulting image compared with the original image and ground truth image, we can see that only the white blood cells are clearly segmented out. The background and red blood cells are classified into one group. In order to trace what had happened to the resulting image, I check the dataset provided by the lecturer and find out if there have any differences between the images. From the dataset, we can identify some of the background images were slightly pinkish. I guess that this is one of the factors that affect the whole segmentation process. As we can see in the original image, the red blood cells color and the background color of this image are slightly matched together. The other factor that affects the resulting image is the preprocessing process before we take the images for the segmentation process.

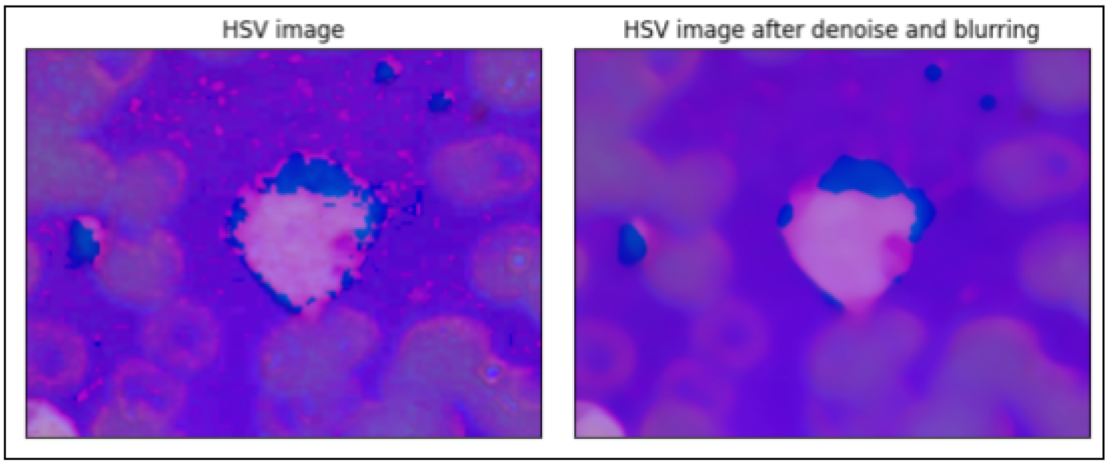
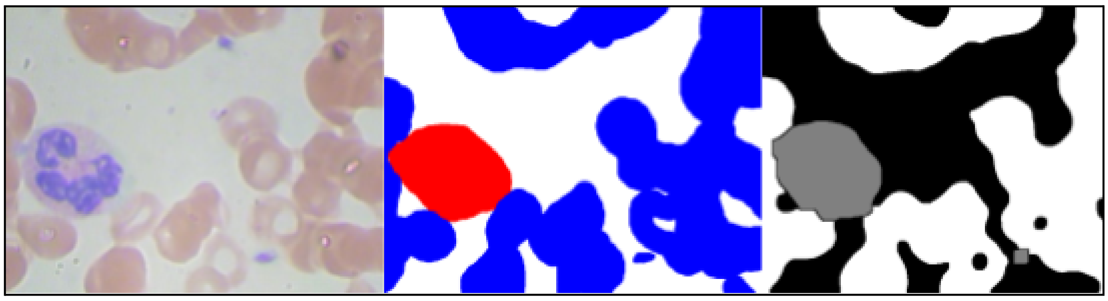
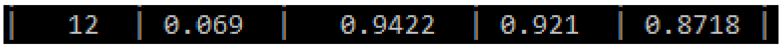


Figure 6.2: HSV image, HSV image after denoise and blurring process

Next, we have discussed the methodology for this project in the above section. First, we convert the input image from RGB to HSV, denoising, and blurring occur in this process. Figure 6.2 shows the resulting image after we convert colorspace, denoise, and blurring. The result shows that when we convert the color space from RGB to HSV, the information on the image was lost. The red blood cells and the background look alike and we can slightly see that there is a mask-like background only for the white blood cells. Thus, the evaluation result is totally bad and the resulting image only can detect white blood cells. We can see that the error rate of this image is about 0.4979 which almost achieves half the loss percentage. Then, the recall of this image is around 0.5782 and the IoU rate only has 0.3911 which is the worst result in all test images. The precision rate of this image is quite good which is 0.6228 because the segmentation of the white blood cells holds a very high percentage of the image.

Best Result Image





```
|    12    | 0.069 |     0.9422    | 0.921  | 0.8718 |
```

Figure 6.3: Original image, Groundtruth image, Result image, and evaluation result of the 12th image

Based on the observation on the image above, the images are segmented in the best result among the test images. Make a comparison between the ground truth image provided by the lecturer and the resulting image, the segmentation of the background, red blood cells, and white blood cell have almost similar to each other. We can observe that the white blood cell in the resulting image is slightly wider than the ground truth image. On the other hand, the red blood cell provides better accuracy on segmentation compared to the white blood cells.

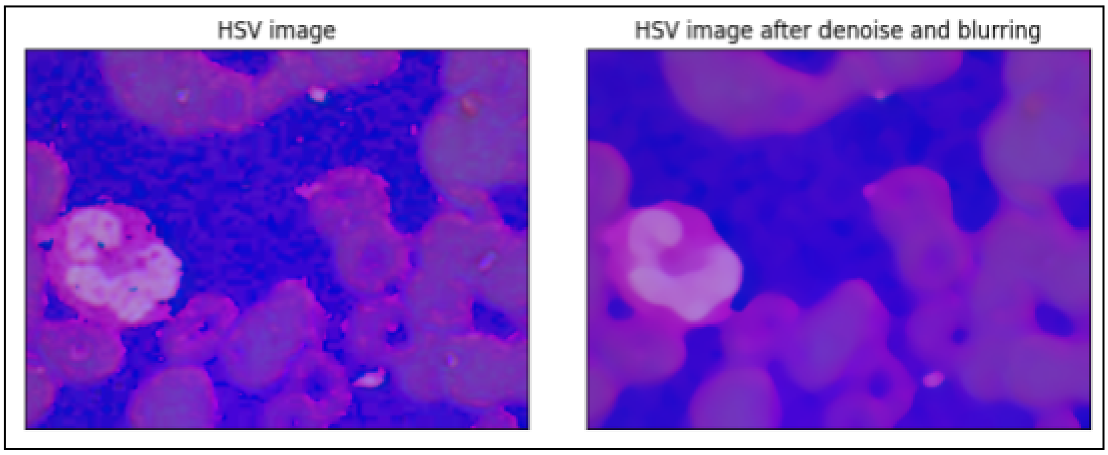


Figure 6.4: HSV image, HSV image after denoise and blurring process

Consequently, the evaluation result of this image is quite good which is the best result of all the images. The error rate of this image is just 0.069, the Precision rate achieves 0.9422 and the recall rate achieves 0.921 which is the highest result. Then, the overall result of this picture is 0.8718 which is higher than the average IoU result in this method. As we can see in figure 6.4 shows that this image provides a clean and clear image not only for the background but also a clear vision of identifying white blood cells and red blood cells. The difference can be clearly shown after the denoising and blurring process. Not only that, from the observation from the original image, we can see that the image background is more apparent in the white background compared to the 38th image. Thus, this is the reason why this image achieves the highest accuracy among the others.

Suggestions for Improvement

The limitation of this proposed method is the setting of tuning for the image may not be suitable for all images for the segmentation process. The color range value is hard to determine and tune the image one by one to the ideal segmentation results. Based on the proposed method in this assignment, some improvements can be made in the future. K-Means Clustering can be used to have an unsupervised learning algorithm to segment different parts. The machine learning algorithm is able to cluster different image colors. With the clustering of colors, we are able to choose the target color for either segment for background, red blood cells, and also white blood cells accordingly.

Collaboration

Student Name: Loo Chen Zhi, Chang See Jie, Sia Mandy

Student ID: 1181103230,1181103362,1181103091

I have some discussions with the following students for ideas to segment out the necessary parts. We exchange ideas and brief each other on the process flow of segmentation at the beginning phase.