2020

Assignment 1

COMP9517 | COMPUTER VISION | 20 T3

ANANT KRISHNA MAHALE |Z5277610

Task 1

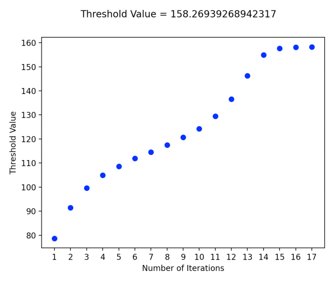
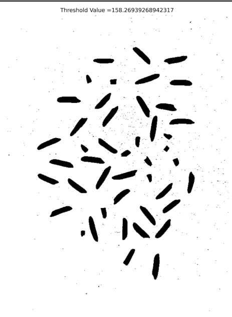
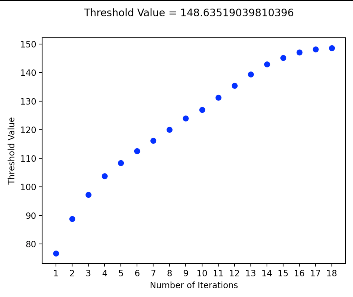
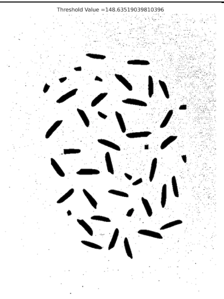
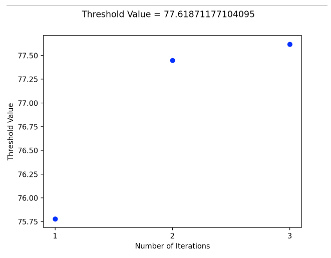
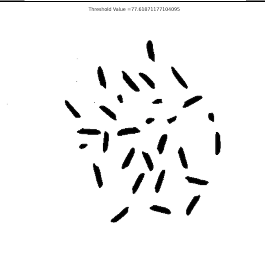
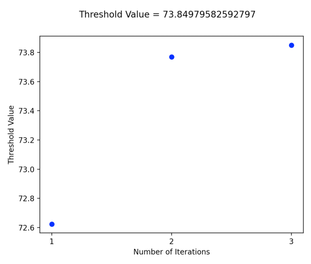
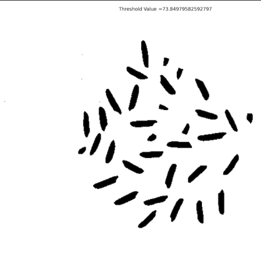
This task aims to find the Iso-data intensity thresholding using the algorithm given in the document.

The algorithm is as follows:

1. Select an arbitrary initial threshold t
2. Compute μ0 mean of intensity values < t and μ1, mean of intensity values > = t
3. Update the threshold to the mean of the means t_new = (μ0 + μ1)/2
4. If the threshold has changed non-trivially in step 3 go to step 2.

For arbitrary initial threshold value, I have taken it as 60 and epsilon as 0.001 and converted above steps to python script.

Threshold values for given input images:

Image Name	Threshold Value	Output Graph	Final_Output Image
rice_img1.png	158.269		
rice_img2.png	148.635		
rice_img6.png	77.618		
rice_img7.png	73.849		

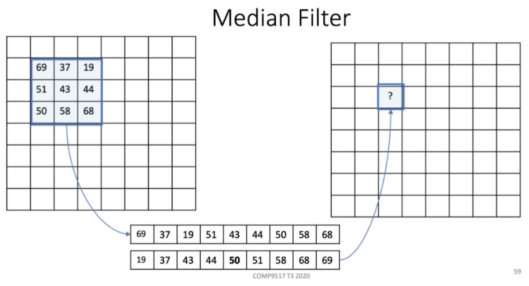
All the intermediate results are in the directory rice_img*_intermediate_results/Task_1

Once the threshold value for the particular image is found, I'm replacing values that are less than < t with 0 and >=t with 255 for further processing of Task 2. Since Task one requires image with white background, I'm replacing 255 with zero and vice versa.

Task 2:

Task 1 returns an image which is input to Task 2. Since this image has salt and pepper grains, Median filter need to be applied.

A function apply_median_filter() is used to remove the salt and pepper grains. I have taken 6*6 filter for in this function. If it's a corner pixel, zeros are added in the filter else, pixels that fall within the filter size are considered. And finally, the pixel value is replaced with median value. Below picture is taken from the lecture slide which has 3*3 filter. However, my code has 6*6 filter.



Connected component algorithm to count the rice kernals:

1. First Pass:
   In this step, Image is iterated through each pixel, if it is background (0 in my program), it is ignored, else, smallest neighbour value is assigned and equivalences are stored.

   How?
   The function find_neighbour_value() returns all the neighbours of particular pixel in a list. The function connected_component_labelling() then assigns smallest value in the list to the output image. Rest of the values are saved as equivalences in a dictionary.

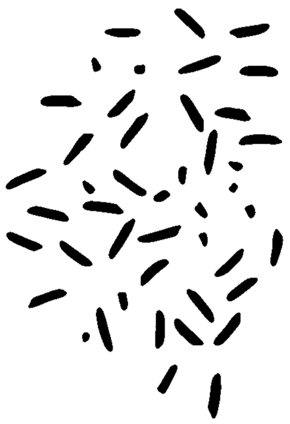
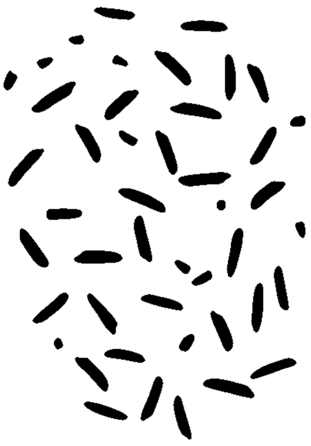
```
{1: [1], 2: [2, 3], 3: [2, 3, 4], 4: [3, 4], 5: [5, 6], 6: [5, 6, 7
], 7: [8, 6, 7], 8: [8, 9, 7], 9: [8, 9, 10], 10: [9, 10, 11], 11:
[10, 11, 13], 12: [12, 14], 13: [11, 13, 15], 14: [16, 12, 14], 15:
[17, 13, 15], 16: [16, 14], 17: [17, 15], 18: [18, 19], 19: [18, 19
, 20]…}
```

   Then the function modifyLabelDict() replaces the list with smallest equivalences and returns the dictionary to the connected_component_labelling() function.

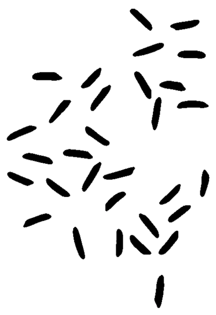
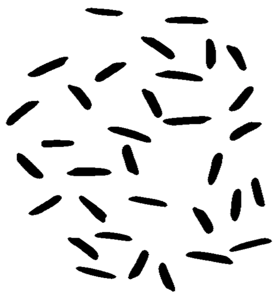
2. Second Pass:
   In this step, while iterating through each pixel, the value of the connected component is replaced with smallest equivalence value from the dictionary.

Counting the Number of Rice Kernels:
The number of connected components (unique) obtained in the above steps is nothing but number of rice kernels in the image. This can be done by getting unique values from the dictionary created before.

Output of the Image:
Since Task_1 returned image with black background, I had to convert to white background before writing it to the disk.

Input Image	Filtered Image	No. of Rice Kernals
rice_img1.png		45
rice_img2.png		45
rice_img6.png		29
rice_img7.png		34

Task 3:
This task aims to find the number of damaged rice kernels. This can be done by iterating through each pixel in the image and counting the number of each connected component. Whichever component has less area (count) than the min_area passed by the user, should be set to zero.

Input Image	min_area	Filtered Image	Damaged Rice Kernels
rice_img1.png	690		14
rice_img2.png	600		12
rice_img6.png	200		6
rice_img7.png	200		7