COSC 6380 Digital Image Processing

Region Counting and Image Compression

1. <u>Region Counting</u>
   A. In this part, I have computed histogram to find the occurrence of each pixel value in the image and plotted using matplotlib. I have written a code to find the optimal threshold using probability distribution and expectation. I have calculated the threshold until the difference between 2 thresholds do not differ. Using the optimal threshold, I have computed binary image (0 and 255).

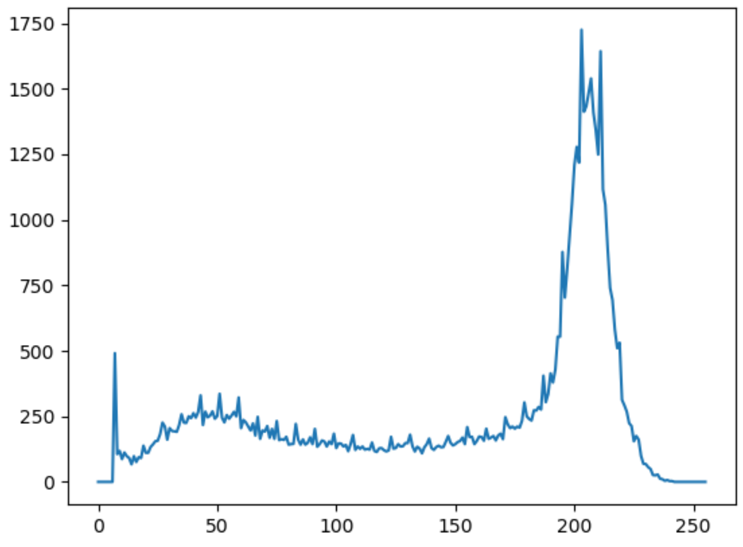


Figure 1. Histogram of given image

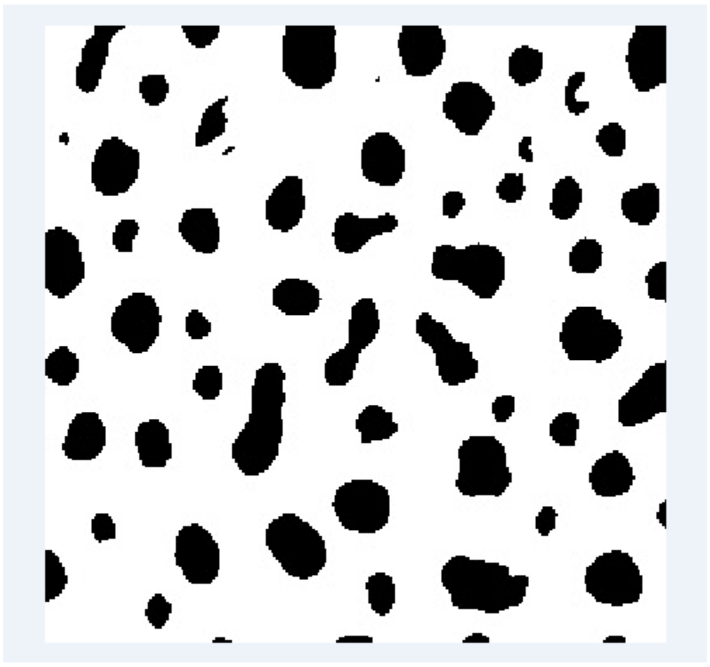


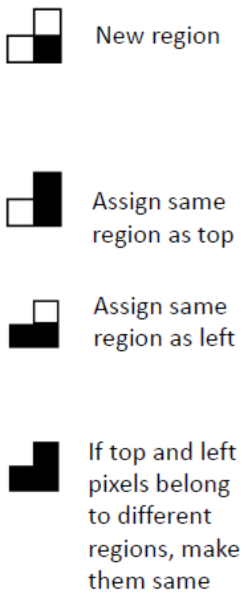
Figure 2. Binary image

Problem faced:
I did not face any problem except finding mean1 and mean2. To solve the issue, I created to empty lists, temp1 and temp2, and appended values for two different regions in corresponding temp list to find sum and therefore new threshold.

B. In second part, using 2 neighbour windows, I have assigned region numbers to the blob in the binary image. I have followed the technique mentioned in the figure. To verify the number of regions computed, I have created colour image by random colour to each region computed. This function returns the region dictionary having positions of each region computed. For example, region = {‘1’: [(1,1), (1,2)], ‘2’: [(2,1), (2,2)], ‘3’: [(3,1), (3,2)]}, where key denotes region number and values denote pixel positions for that corresponding region.

Later, using the region dictionary, first I have removed the regions which are having less than 15 pixels and then, computed area and centroid for each region. For example, stats = {‘1’: [(1,1), 100], ‘2’: [(2,1), 200], ‘3’: [(3,1), 300]}, where key denotes region number and first value belongs to centroid and second belongs to area of the region.

In last part, I have used cv2.putText() command to denote centroid and area in the image. Here are the outputs.



New region
Assign same region as top
Assign same region as left
If top and left pixels belong to different regions, make them same

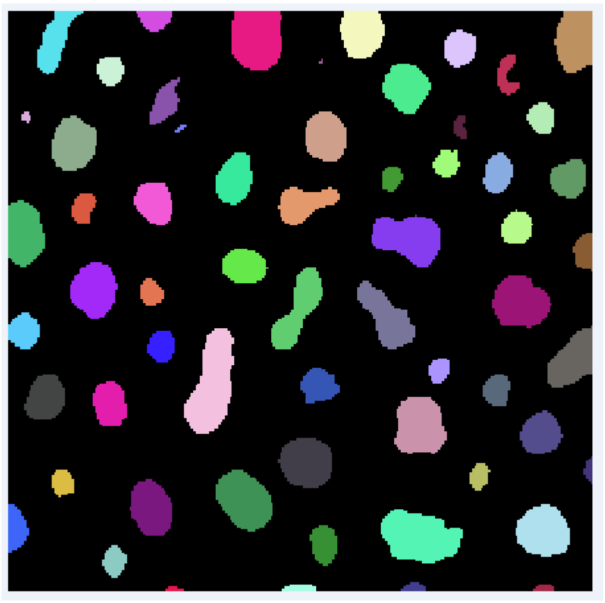


Figure 3. Regions coloured

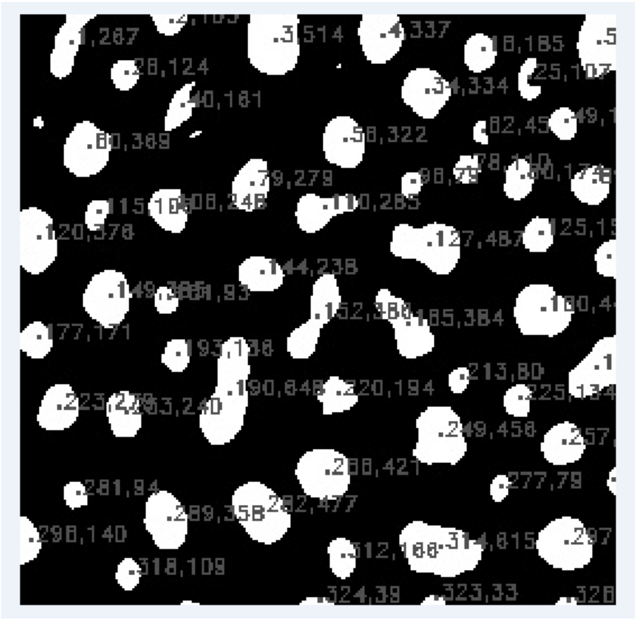


Figure 4. Regions labelled with centroid and area

Important points explained:

1. To check left and top values of first row and first columns (according to the 2 neighbour windows), I padded the new image which will be updated with region numbers with 0 using np.zeros().
2. In the condition when left and top both are region values, and both are different, I assigned top pixel value to current pixel as well as left pixel and extend the region position with all the positions of left pixel value. To clear the position values of left pixel, I used regions.clear() command.
3. To remove the unnecessary regions, I popped out keys which has zeros values in it.
4. To demonstrate colour regions to verify the number of regions detected, I used random function to assigned colour value to each region.
5. To label each region detected, I used cv2.putText() command.

2. <u>Image Compression</u>

In this part, I have implemented run-length coding to represent the image. Here is the example of run-length coding.

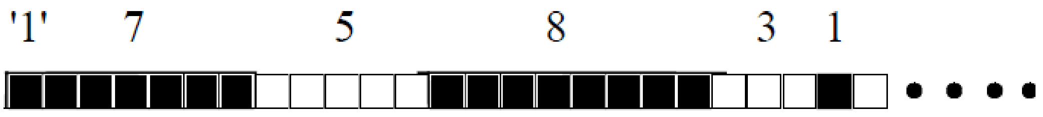


Figure 5. Run-length coding

I have returned the run-length code in a list which used in decode_image() function to retrieve back. I successfully decoded the original image. Here is the output of decoded image.

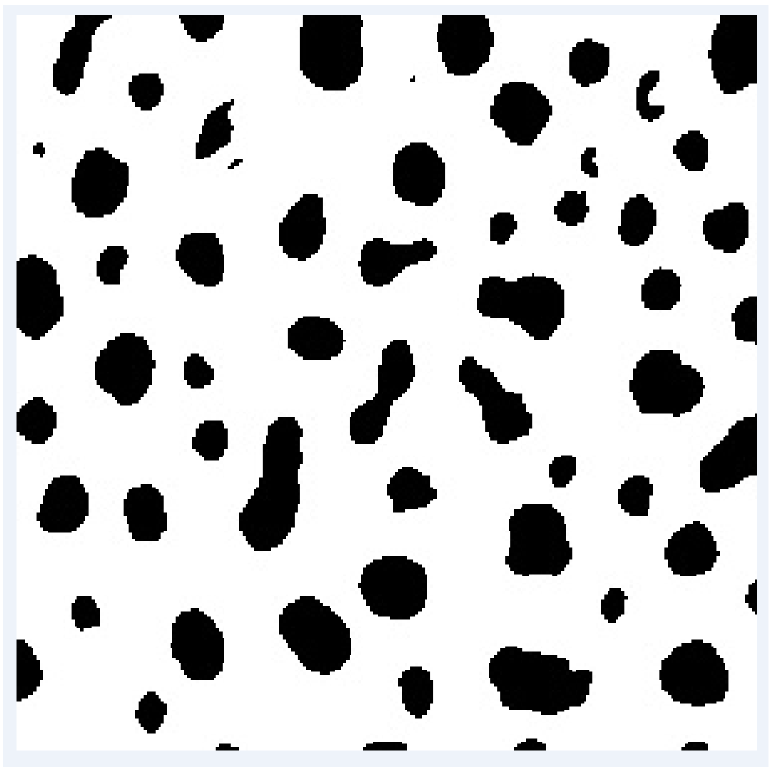


Figure 6. Decoded image from Run-length coding

Problem faced:

For finding run-length code, I did not face any issue. I tried to solve the requirements using list.

I have successfully completed all the tasks. Code can be run using command:

python dip_hw2_region_analysis.py -i <image-name>