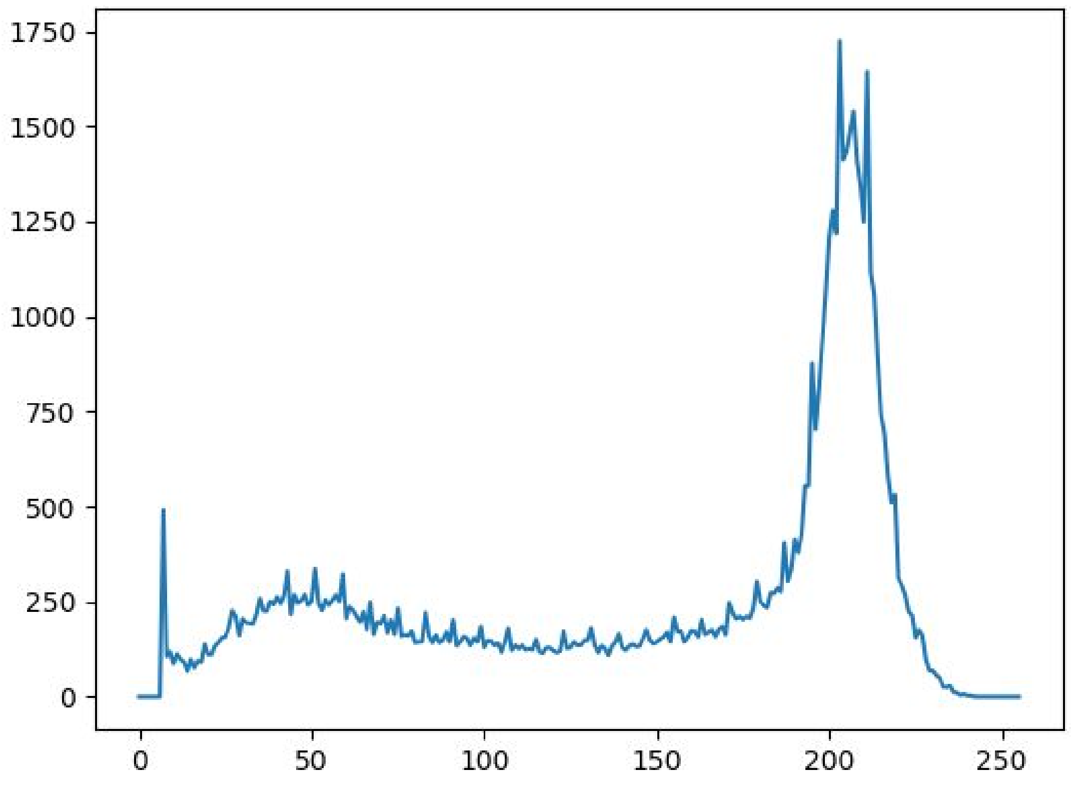
COSC 6380 - Computer Vision
Ashna Shah - 1638044
Assignment 2 Report

This assignment is mainly about region counting and image compression of binary image. There is one main file named dip_hw2_region_analysis and three other files in which there are methods to perform various actions.

1) Binary_image.py
    a) Method compute_histogram
        i) This method takes input a gray scale and counts the frequency of various intensity values in the image.
        ii) It returns a list of frequencies as values at indexes which are intensity value from 0-255.

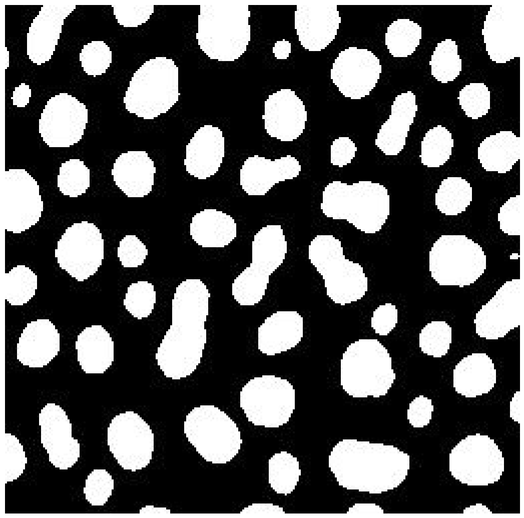


Histogram example of image cells.png

    b) Method find_optimal_threshold
        i) It takes input the list of frequencies that we created in previous method which would help us find optimal threshold.
        ii) So, first we initialize threshold to half of max intensity value.

iii) After that, we create two lists - below threshold and above threshold and compute mean for both of the lists.
iv) Taking average of means would give us new threshold.
v) We would repeat steps 3 and 4 until difference between previous and new threshold is less than 1.

c) Method binarize
i) It takes input the grayscale image and the optimal threshold for the image.
ii) We simply loop around the image to check if the intensity value is less than threshold, it has to be white(255) else it has to be black(0) and store these values in new array of binary image.


Binary Image of the grayscale image provided.

2) Cell_counting.py
a) Method blob_coloring
i) It takes binary image as input.
ii) First, we make a region of the same size as the binary image.
iii) Looping around the binary image, we compare each pixel with its top and left pixel and assign the region accordingly to the new region.
iv) If top and left pixel are having same intensity value, we first check whether they belong to same region or not.
v) If not we need to add the list of coordinates of that regions into a new region and delete them from previous regions which is done in method called “transfer coordinates”
vi) If top pixel is having same intensity as present one we assign the region of top pixel. Similarly, we do that with left pixel.
vii) If top and left are having same intensity value but different than present pixel then we add new region into the dictionary.

b) Method compute_statistics
i) It takes input a dictionary whose keys are region number and values is a list of the corresponding coordinates of the region.

ii) We compute the total area of the region by taking the length of values in the dictionary.

iii) We also compute centroid by taking

(1) Sum of x-coordinates divided by total area of the region

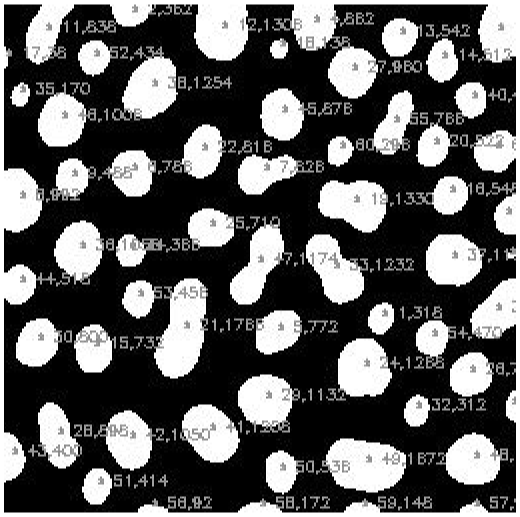
(2) Sum of y-coordinates divided by total area of the region

iv) We return a dictionary “stats” which has key as region number and in value - a tuple of coordinates of centroid and area of that region.

c) Method mark_regions_image

i) It takes input the binary image and the statistics that we computed in method compute_statistics.

ii) So now we simply mark the centroid at the coordinates we obtained with the region number and its area, and return that image



Binary Image with markings on it.

3) Run_Length_Encoding.py

a) Method encode_image

i) It takes binary image as input.

ii) We start scanning image row-by-row and firstly store the intensity of first pixel.

iii) After that we assign a counter which is incremented until we encounter a different intensity value.

iv) So, if we find a different intensity value, we reset the counter and start increasing that counter until again we find different intensity.

v) As this is binary image, we only have two intensity values.
Eg:- 0,7,8,5,...means with 0 as intensity value there are 7 pixels, 8 pixels with intensity value 1 and again 5 pixels with intensity value 0.

b) Method decode_image

i) It takes input as the run_length_code that we computed in previous method.

ii) We loop around that and take the first value which is the initial intensity value and accordingly assign other intensity value to some variable.

iii) In the loop now, we extend the list with these intensity values alternatively and get the binary image back.