E9 241 Digital Image Processing

Assignment – 01 Report

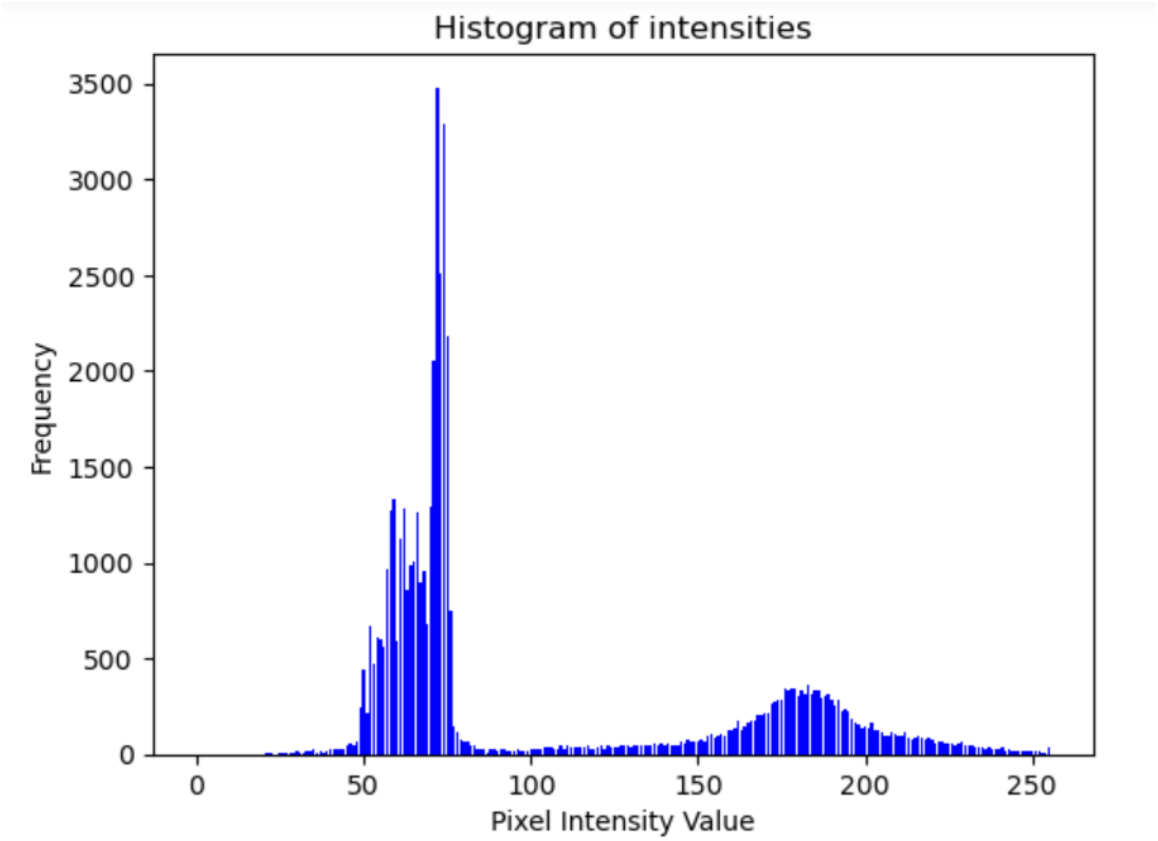
Name: Bhavana Mittapalli

Sr No: 06-18-01-10-51-24-1-24114

Department: Computational and Data Sciences

Question-1, Histogram

The implemented function takes gray scale image as input and gives a list or vector of size 256 with frequencies at each intensity level. The Histogram achieved after finding the intensity frequencies of the coins image is as follows



The average intensity from histogram is 103.30500158906722

And the actual average intensity of the image is 103.30500158906722

Observations:

The histogram successfully visualizes the frequencies of intensities of the coins image.

We also observed that the actual average intensity of the image and calculated average intensity of the image from the histogram is the same. This concludes that the calculated histogram analyzes the gray scale image intensities.

Question-2, Otsu's Binarization

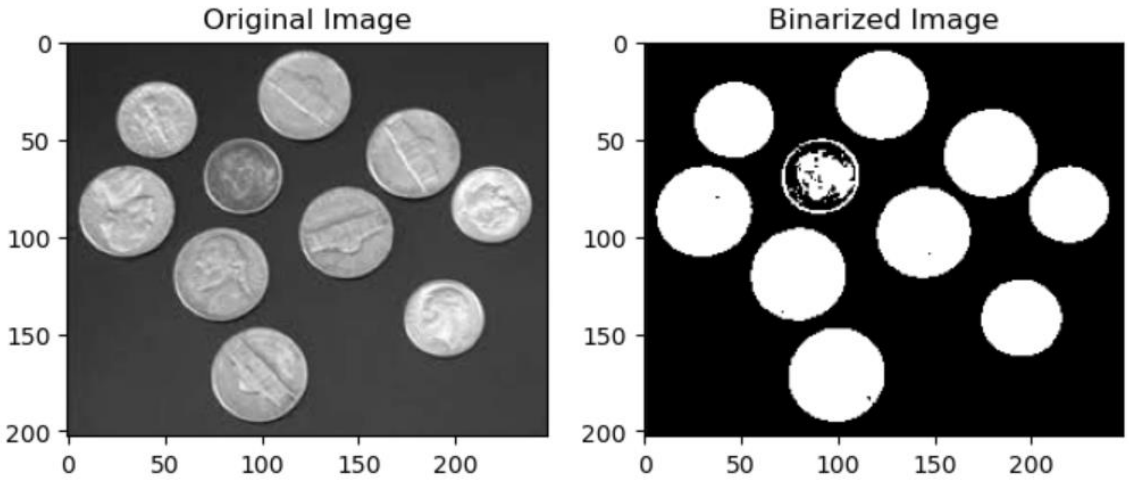
Ostu's binarization is a method of finding Threshold for binarizing the gray scale image. Ostu's binarization calculates threshold in two ways, one is minimizing the within class variance and one by maximizing the between class variance. Because the sum of the both is a constant we can do it in either ways. But in both ways, we get the same threshold.

Observations:

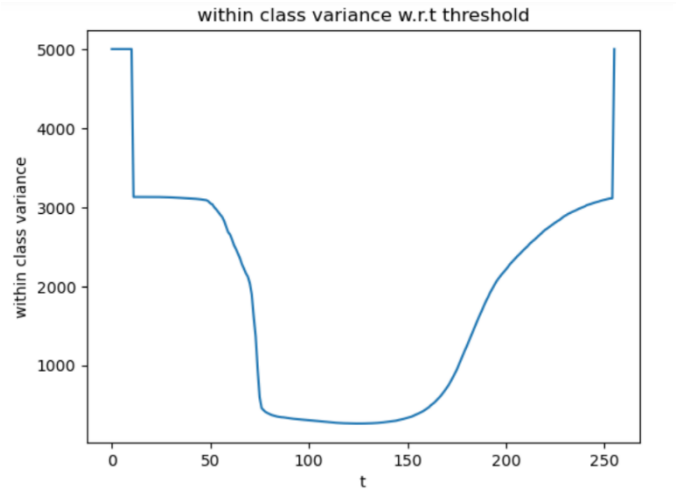
We need to Binarize the given coins image using Ostu's Binarization.

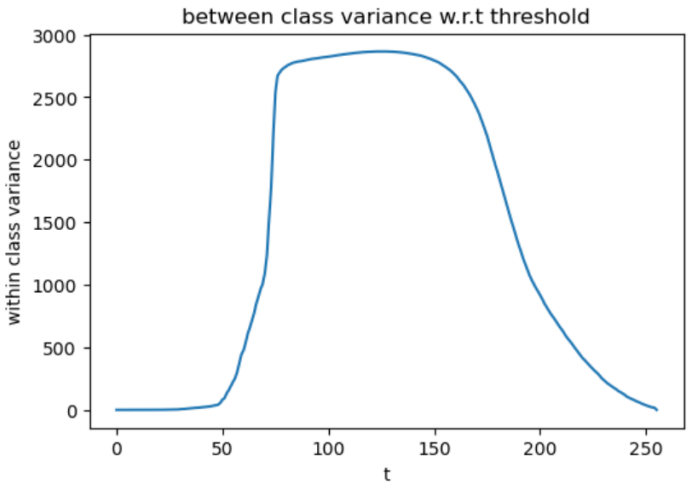
The Optimal threshold by minimizing the within class variance: 125

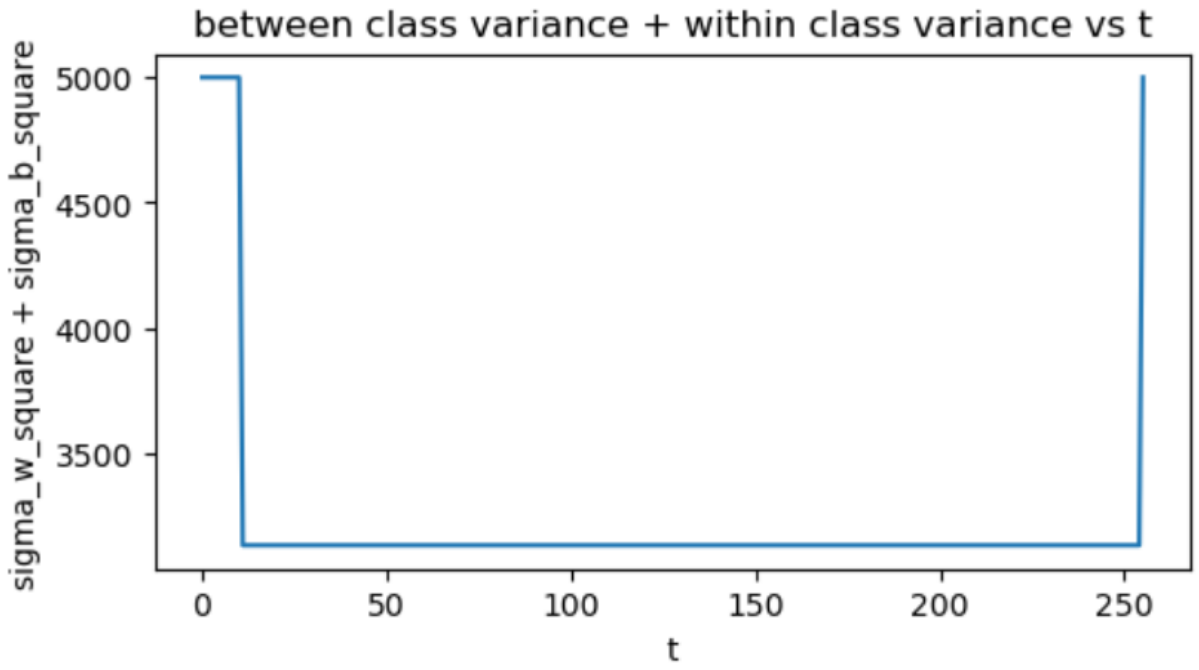
The Optimal threshold by maximizing the between class variance: 125



Plots of $\sigma^2 w(t)$, $\sigma^2 b(t)$ and $\sigma^2 w(t) + \sigma^2 b(t)$ with respect to threshold





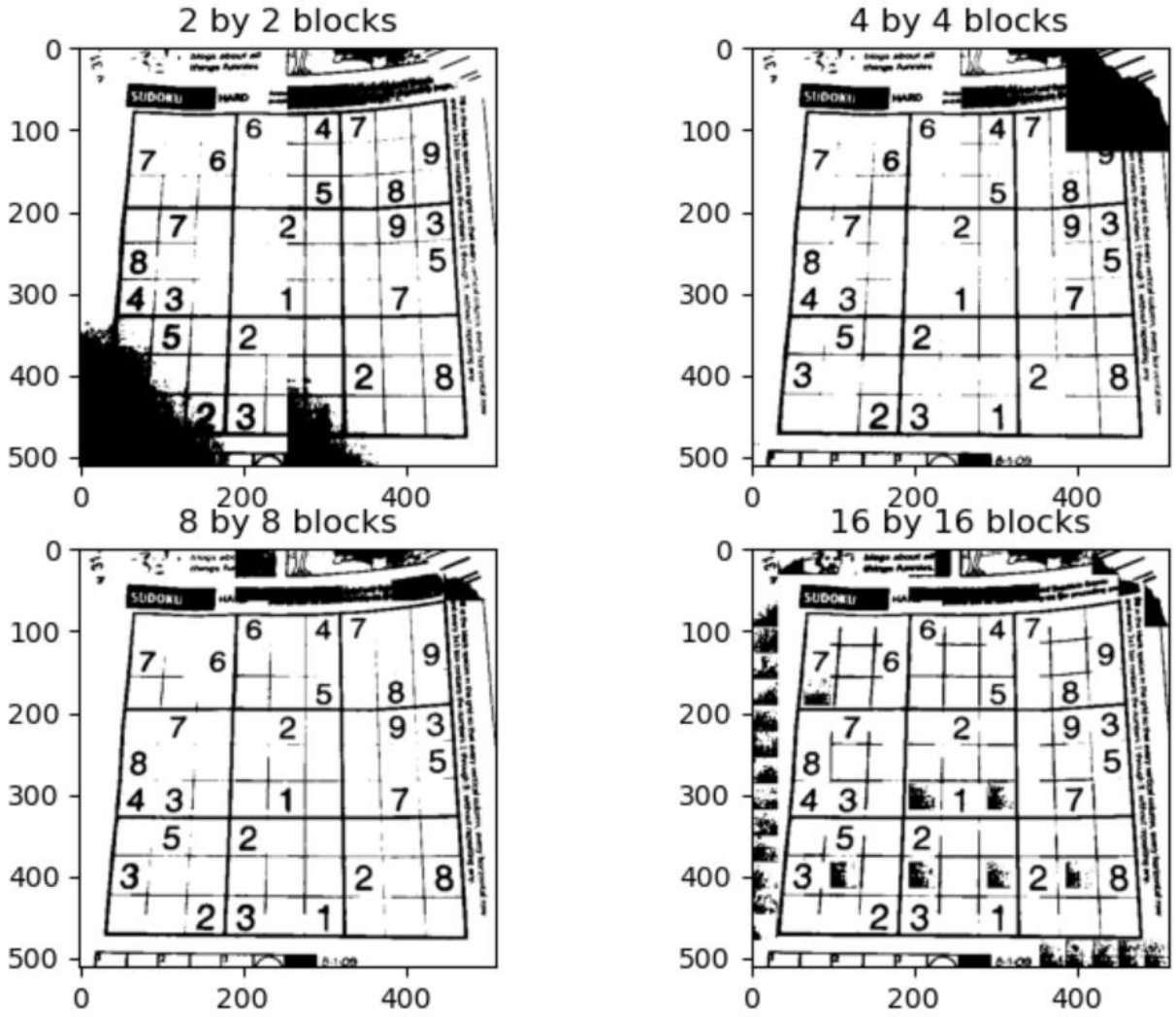


We can see that optimal threshold by minimizing within class variance and maximizing between class variance are same. And from the above plot we can see that sum of within class variance and maximizing between class variance is constant. This is why we can either minimize within class variance and maximize between class variance.

Question-3, Adaptive Binarization

We need to divide image into N*N blocks and then we need to apply Ostu's binarization on each block separately and stitch them back together.

Adapative Binarization images for different N's are



Observations:

Adaptive Binarization: Instead of binarizing the whole image directly we dive the image inro blocks and binarize each block separately and then joins back together.

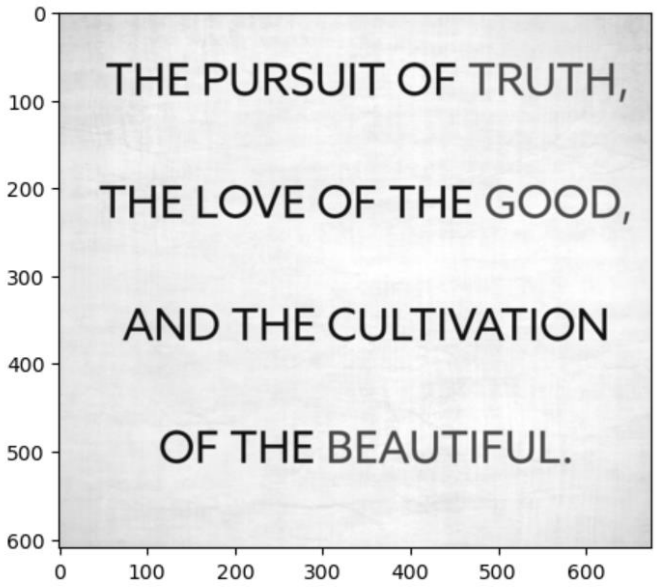
When we try to find threshold for whole image, we consider all pixel's intensities into account and we will try to find threshold which minimizes within class variance. But sometimes there can be a chance where when the pixels are far from each other and they carry different intensity information(irrelevant), in this case one can influence other.

Instead, if we consider a small block near pixel and then binarize, it considers near pixel information and relevant intensity information into account while calculating threshold.

This can give better view of the binarized image. From the plots, we can also see that as we increase blocks, we got better threshold and better image. Clarity of image increased while we increased from 2*2 blocks to 8*8 blocks.

Question-4, Connected Components

We need to find the connected components, that is number of characters of the given quote image without punctuation.



Number of characters for the given quote image excluding punctuations = 64

Algorithm:

The code consists of two components:

1. **give_label_component** : This part of the code takes image and one pixel address as input and gives a label to all the elements of the connected component of given pixel address. It starts from given address and checks all its 8 neighbors until that connected component is covered and label
2. **count_characters**: This part of code use give_label_component and labels all connected components or (characters of the given image). And then next part of the code finds number of unique labels that is no of characters and filters punctuations and returns number of chracters as output.