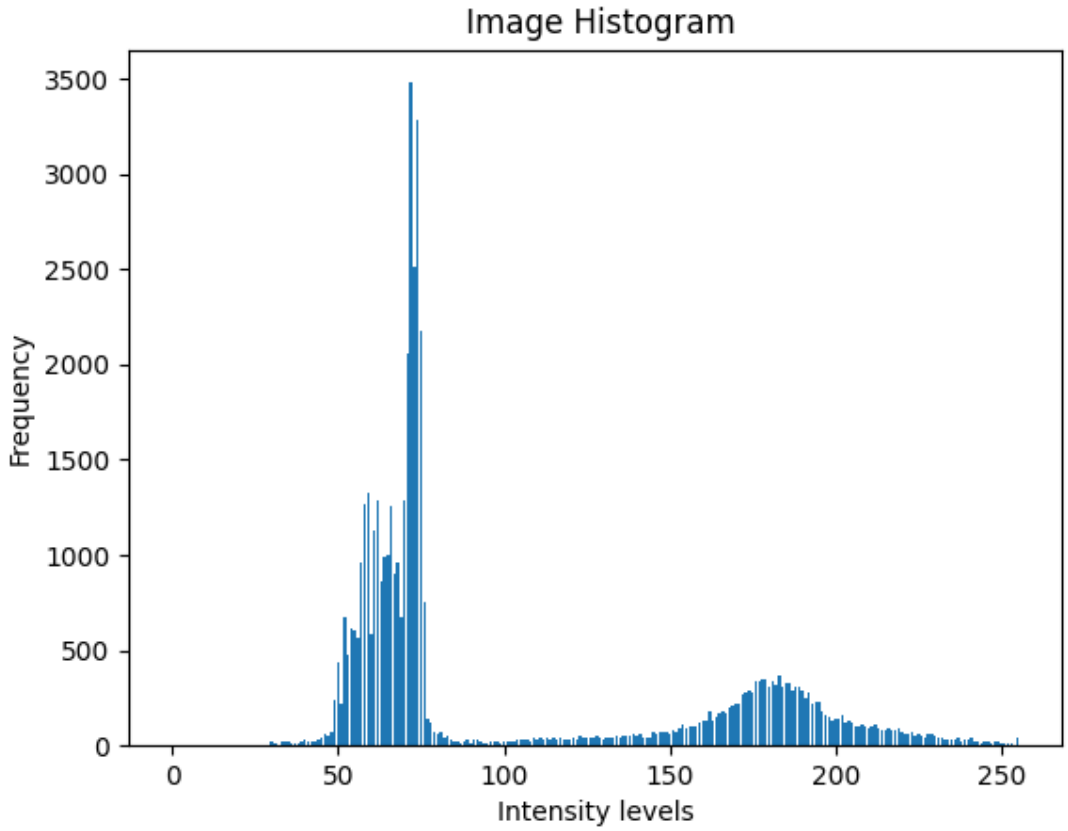
E9 241 – Assignment 1

Adithya Ponnuraj – 20420

QUESTION 1

The histogram of the image coins.png is attached below:



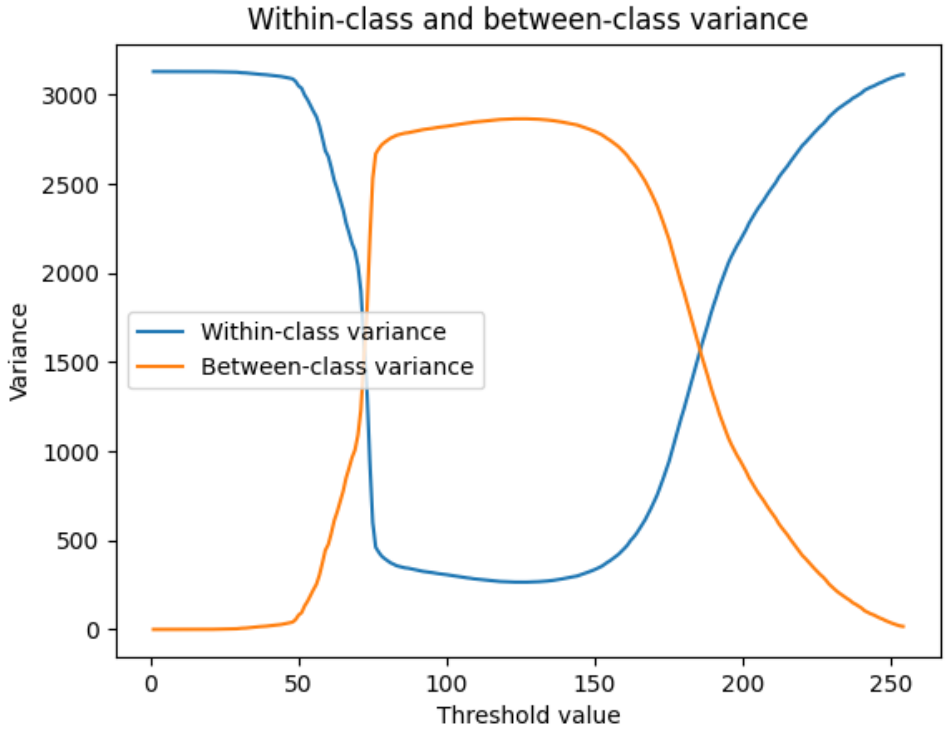
When we find the average value of the intensity of the image by using the histogram (i.e., avg_intensity = np.sum(np.arange(0, 256) * histogram) / np.sum(histogram) and by using the image (i.e., np.sum(img) / img.size), we get both values to be 103.30500158906722, as reflected in the output of the code.

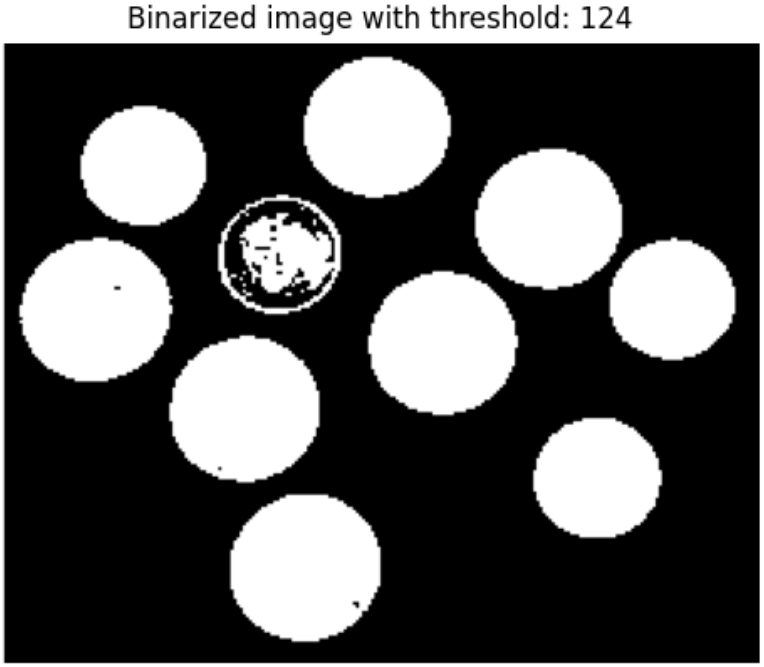
QUESTION 2

In this question, we perform Otsu's binarization by minimizing the within class variance and maximizing the between class variance. First, we plot the values of variance vs. threshold for all threshold values between 0 and 255 for both types of variances, as shown in the attached figure. The runtimes for both of these methods were calculated by using the inbuilt 'time' package, which gives a more accurate representation of the answer in jupyter notebooks than in python. The two runtimes are as follows:

- Runtime of within-class variance method: 0.002321
- Runtime of between-class variance method: 0.000167

Therefore, we see that maximising the between-class variance is faster than minimizing the within-class variance and hence, in the following questions and their implementations of Otsu's binarization, we use this method. Both methods arrived at a common value of optimum threshold of 124 for coins.png. The binarized image is also attached below:





QUESTION 3

For this question, we used the inverse depth map to extract the foreground. The same Otsu's binarization protocol implemented in question 2 was used to binarize the inverse depth map. From this, the foreground of the depth map corresponding to the foreground of the IIScText.png was extracted (i.e., an image copy of only the foreground elements was made) and was superimposed (i.e., wherever there was information of the text to be superimposed, the background image's data was overwritten on those pixels with the superimposed image's data, leading to the final image), which is attached below:

QUESTION 4

For this question, we use connected component analysis to find all the connected components in the binarized (again, using the Otsu's implementation from Question 2) image of the given quote.png image. There are three punctuations in the given text, which are then filtered out by setting a minimum threshold pixel area above which a character can be deemed a valid character. After trial and error, this threshold was chosen to be 275. The output of the function gives us a total valid character count of 64, which (as confirmed by word and inspection) is the correct number of characters in the given text.

Connected Components