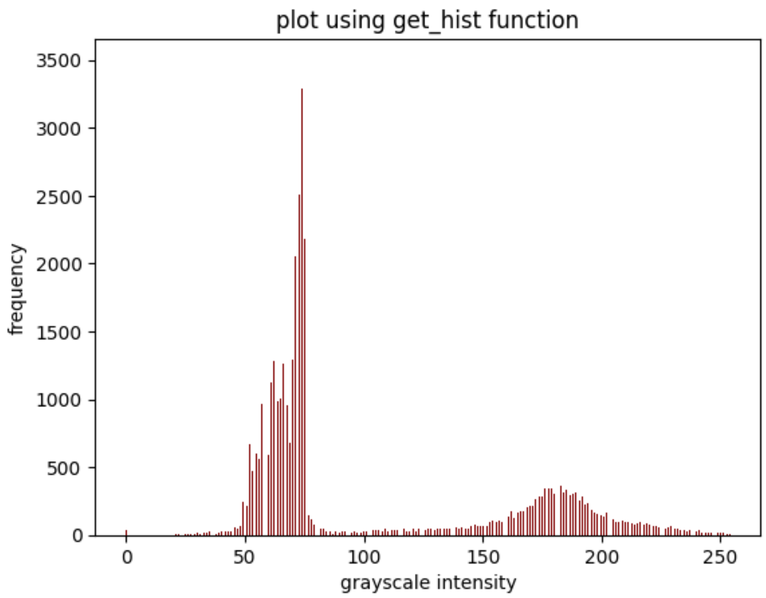
Report- Assignment 1

1. **Histogram Computation: Compute the histogram of the image coins.png, by finding the frequency of pixels for each intensity level {0, 1, . . . , 255}. Show the histogram by plotting frequencies w.r.t. intensity levels. Comment on what you observe. Also, find the average intensity of the image using this histogram. Verify the result with the actual average intensity**

RESULTS:

```
time taken= 0.05159139633178711
Average calculted from histogram = 103.30500158906722
Average calculted over all pixel = 103.30500158906722
```



INFERENCES:

- The majority of the intensities lie between 50 and 80 with peak near 75. Therefore it is the background colour of the image
- The background is black and the coins are of intensity around 175.

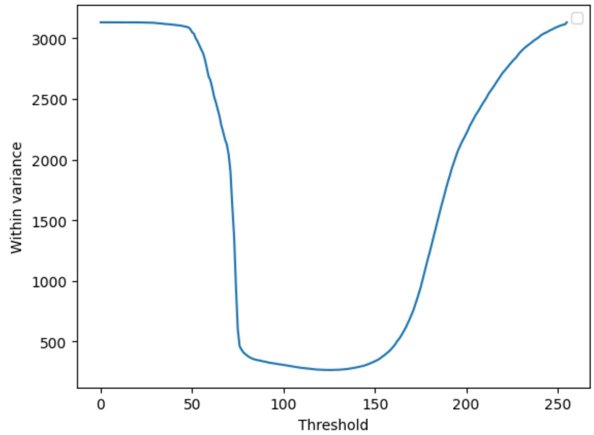
2. **Otsu's Binarization: In the class, we showed that σ 2 w(t) + σ 2 b (t) = σ 2 T , where t is the threshold for binarization. Binarize the image coins.png by finding the optimal threshold t by:**
   (a) Minimizing the within class variance σ 2 w(t) over t.
   (b) Maximizing the between class variance σ 2 b (t) over t.
   Verify that both methods are equivalent. Compare the time taken by each of the approaches.

RESULTS:

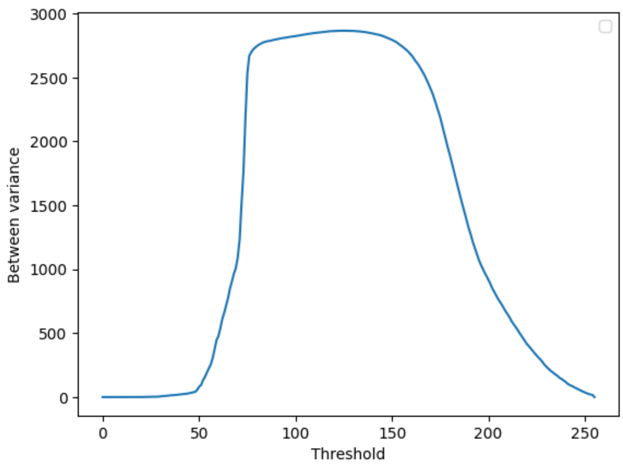
Verifying that maxima of σ 2 b(t) and minima of σ 2 w(t) occur at same value of t.

```
time taken for between variance calculation = 0.3152885437011719
time taken for within variance calculation = 0.23804402351379395
variance between class = 2865.7017638569228
theshold for max between class variance = 125
variance within class = 265.1024571148032
theshold for min within class variance = 125
```

Plot of σ 2 w(t) :



Plot of σ 2 b(t) :

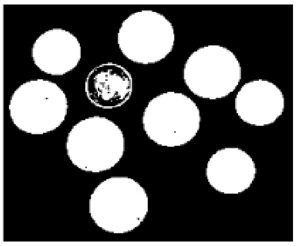


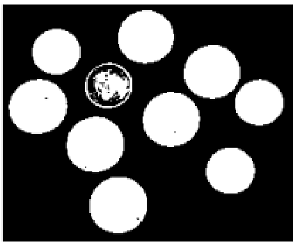
INFERENCES:

- We can see the thresholds found by minimizing within class variance and maximizing between class variance are identical and thus it is evident that they must be identical procedures.

$$\sigma_w^2(t) + \sigma_b^2(t) = \sigma_T^2$$

- It is intuitive from above equation, we know the total variance is independent of the threshold, if we maximize one of the functions, the other in that case would be the minimum. Thus it is mathematically consistent
- The time taken by both the processes are identical.





3. **Depth based Extraction: The image IIScTextDepth.png is an inverse depth map of IIScText.png. A depth map indicates the depth of an object from the camera for each pixel. Particularly, an inverse depth map has a higher value when the object is nearer to the camera and a lower value when it is farther apart. Binarize the inverse depth map IIScTextDepth.png and use that information to extract the text in IIScText.png and display it over the background image IIScMainBuilding.png. The expected image is shown below.**

   REPORT:

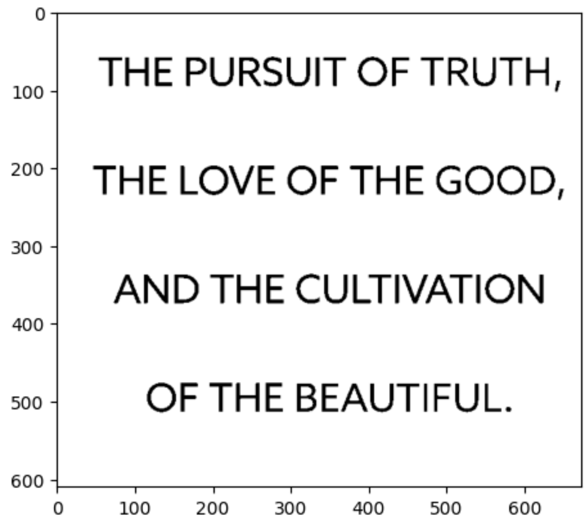
INFERENCES:

- In this process first the IIScTextDepth.png is Binarized using otsu's binarization algorithm. The binarized image then is used to extract the information from the IIScText.png and paste on the IIScMainBuilding.png
- The question helps us to understand intuitively what a depth map is used for. We can use depth map to basically extract information of the background and foreground (in this case) using elementary methods like binarization. Depth map is a method to decipher depth in a 2D image.

4. **Connected Components: Binarize the image quote.png and count the total number of characters excluding punctuations using connected component analysis.**

REPORT:

Binarized image:



Region matrix:

Details of sets of regions:

```
length of list =  67
height of all regions = [36 36 36 36 36 36 36 36 35 36 35 35 36 35 36 37 37 13 36 36 36 36 36 36
 36 36 36 36 36 37 37 37 37 35 13 36 36 36 36 36 36 36 36 36 36 36 36 37
 36 37 36 36 36 36 36 36 35 36 35 35 35 36 35 37 35  5 35]
Average height of pixels =  34.80597014925373
```

Result:

```
no. of letters in Image =  64
time taken= 12.135975837707520
```

INFERENCES:

- This question helped to get an idea of how regions can be made in an image. This knowledge opened up new doors to what and how an image can be manipulated.
- In this approach height to the regions was used to differentiate between letter and comma but, more sophisticated method can be devised for an image where varieties of symbols are present.

5. **(Optional Bonus Question) - MSER: Maximally Stable Extremal Regions (MSER) correspond to regions of connected components which, when thresholded around a certain threshold, are stable in terms of the size of the component. Determine the number of characters in Characters.png based on MSER.**

   Think about why finding connected components over an Otsu binarized image will not work well in this scenario.

   REPORT:

   Characteristic features of all regions formed with their info:

```
info of image- format:-[[jth element of marix, ith element of marix, height, width]]
[[1029  350  187  157]
 [ 209  350  187   56]
 [ 613  350  187  203]
 [1250  529  187  204]
 [1218  308  189  158]
 [1244  338  187  204]
 [1240  307  187  204]
 [1149  322  189   26]
 [1215  328  189  159]
 [1149  346  189   38]
 [ 259  350  189  159]
 [1052  349  187  204]
 [ 202  350  189   44]
 [1220  421  187  204]
 [1132  340  189   57]
 [ 282  349  187  204]
 [1119  308  189   57]
 [1188  333  187  204]
 [1097  324  189   57]
 [1182  400  188  204]
 [1085  319  189   57]
 [1080  308  189   58]
 [ 157  308  189   58]
 [1143  309  188  204]
 [1146  345  188  204]
 [ 241  325  188  204]
 [1052  308  189   58]
 [ 282  291   71  204]
 [ 230  313  198  204]
 [1027  309  188  204]
 [ 230  269  111  204]
 [ 230  289  151  204]
 [ 230  282  136  204]
 [ 708  331  189  212]
 [ 737  346  194  211]
 [ 385  350  189  211]
 [ 731  375  195  105]
 [ 711  385  195  105]
 [ 723  404  195  104]
 [ 727  417  195  102]
 [ 773  350  195  101]
 [ 699  432  188  212]
 [ 730  440  188  212]
 [ 782  350  194  119]]
```

Information about their stability of regions and more:

```
stablity array
[111  99  19   0   1   1   0   0   0   0   0   0   0   1   0   4   1   0
   0   0   0   0   2   0   0   0   0   0   1   1   0   0   0   1   0  69
   0   1   1   0   5   0   0  42]
increments in binarization threshold value =  1
Number of stable regions =  5
time taken =  854.0407636165619
```

INFERENCES:

- This question helps us to utilize the region forming algorithm and use it to find regions which might be not possible to find using otsu's binarization
- Otsu's binarization will not be able to binarize the image. Because the letters have both in black and white pixels, they both cannot showed using a single threshold value.
- In this question, (a) and (b) steps are done

```
# (a) Sweep over all thresholds.
for t in range (0, 255, inc):
  # (b) For each threshold, determine connected components in the image.
  img_m = np.round(io.imread(img))
  img_info = find_region_MSER(img_m, t)
```

```
def find_region_MSER(img, th):
  bin_img = binarize_img_MSER(img, th)      #bin_img[0]->binarized image
                                            #bin_img[1]->mejority pixel black or white
  img = bin_img[0]
  blk = bin_img[1]
  x = nos_MSER(img, blk)
  return x
```

In next part regions are formed,

```
# Making region matrix r[][], and taking account of connected regions in r_lst = [{},{},...]
mn = num_img.shape
for i in range (int(0.3*mn[0]), int(0.7*mn[0]), 1):
  for j in range (int(0.1*mn[1]), int(0.9*mn[1]), 1):
    if (num_img[i][j]== colr and i>0 and j>0):
      if (num_img[i-1][j]== colr):
        r[i][j] = r[i-1][j]
        if (num_img[i][j-1]==colr):
          if(r[i-1][j]!=r[i][j-1]):
            # print("log: found 2 existing region")
            update_region_MSER(0, r_lst, i, j, r[i][j-1], r[i-1][j], True)
            r[i][j-1] = r[i-1][j]
      if (num_img[i][j-1]==colr and num_img[i-1][j]!=colr):
        r[i][j] = r[i][j-1]
      if (num_img[i][j-1]!=colr and num_img[i-1][j]!=colr):
        update_region_MSER(count, r_lst, i, j, 0, 0, False)
        r[i][j] = count
        count += 1
```

In this set of codes the geometry of the regions are determined,

```
for i in range (int(0.3*mn[0]), int(0.7*mn[0]), 1):
  for j in range (int(0.1*mn[1]), int(0.9*mn[1]), 1):
    ptr=0
    for x in r_lst:
      if r[i][j] in x:
        pxls[ptr] +=1
        if (wd_f[ptr]>j): wd_f[ptr]=j
        pxl_wd[ptr] = j - wd_f[ptr]

        if (ht_f[ptr]>i): ht_f[ptr]=i
        pxl_ht[ptr] = i -ht_f[ptr]
      if pxl_ht[ptr]>200 or pxl_wd[ptr]>250:
        r_lst = np.delete(r_lst, ptr, axis=0)
        pxls = np.delete(pxls, ptr)
        pxl_ht = np.delete(pxl_ht, ptr)
        pxl_wd = np.delete(pxl_wd, ptr)
        continue
      ptr+=1
```

In this step the region geometries are compiled in array and others discarded.

The array is then returned

```
for x in r_info:
  if pxls[ptr]<10000 or pxl_ht[ptr]<50 or pxl_ht[ptr]>200 or pxl_wd[ptr]<25 or pxl_wd[ptr]>250:
    r_info = np.delete(r_info, ptr, axis=0)
    pxls = np.delete(pxls, ptr)
    pxl_ht = np.delete(pxl_ht, ptr)
    pxl_wd = np.delete(pxl_wd, ptr)
    continue
  codx = wd_f[ptr]+int(pxl_wd[ptr]/2)
  cody = ht_f[ptr]+int(pxl_ht[ptr]/2)
  r_info[ptr][0] = codx
  r_info[ptr][1] = cody
  r_info[ptr][2] = pxl_ht[ptr]
  r_info[ptr][3] = pxl_wd[ptr]
  ptr+=1

return r_info
```

The returned information is used to “measure stability” of the regions and update count[] of all the region info in img_info[] keeping record of all previous regions found.

```
  # (b) For each threshold, determine connected components in the image.
  img_m = np.round(io.imread(img))
  img_info = find_region_MSER(img_m, t)

  # (c) A connected component is termed an MSER if the size of the component does not change
  #     much (within "tol= 10") for a small perturbation "inc= (5-10)" in the choice of the threshold.
  #     Determination of the stable threshold for each connected component.
  if(isinstance(img_info, np.ndarray)):
    for x in img_info:
      prsnt = False
      ptr = 0
      for y in info:
        if(np.allclose(x, y, atol=tol)):
          count[ptr]+=inc
          prsnt = True
          break
        if(prsnt): break
        ptr+=1
      if(prsnt==False):
        info = np.append(info, [x], axis = 0)
        count = np.append(count, [0])

return info, count, inc
```

- We have scanned the image 3 times for binarization, region formation and geometry calculation. We only scan the matrix column from 0.3 to 0.7 times column length and row from 0.1 to 0.9 times row length, these lengths were chosen by using observation. This was done to improve runtime of the program.
- The limit for height, width and no. of pixels to be cutoff and removed were also selected by observation.
- This question helped to broaden the knowledge of how an image matrix can be manipulated and information be extracted like height, width and center points of region which defines the characteristics of that region.