CS-682 Homework 2

Praveen Menon

G01158999

website http://mason.gmu.edu/~pmenon/cs682/

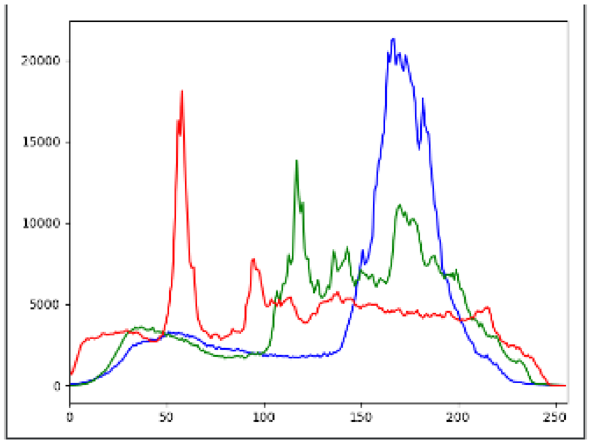
Question1

section1. Use a dialog box to input the image name. Try the exercise with jpeg images too to see the effects of compression.

To create a dialogue box I used askopenfile from tkinter.filedialog. This package gave me an option the select the file from the system folder structure.

Section 2. Once the image is selected from the dialog box I used cv2.imread and cv2.calcHist to calculate the histogram.

To plot the histogram for all three images I used plot from plt package where I mentioned the color as ‘r’, ‘g’, ‘b’. The graph and the input image is shown below



```
import numpy as np
import cv2
from matplotlib import pyplot as plt
from tkinter.filedialog import askopenfilename
filename = askopenfilename()

# Load the colour image assignment part 1
original_img = cv2.imread(filename)

img = cv2.imread('home.jpg')
color = ('b','g','r')

for i, col in enumerate(color):
    original_hist = cv2.calcHist([original_img], [i], None, [256], [0, 256])
    plt.plot(original_hist, color=col)
    plt.xlim([0, 256])
plt.show()
```

Section3. First I used cv2.EVENT_LBUTTONDOWN to select the pixel. To position the 11 X 11 box on the image I used cv2.rectangle. I used cv2.putText to print the data around the box. On selecting the pixel on the image a 11 X11 box will be visible and the mean, intensity and standard deviation is calculated and printed as requested. I also implemented the setMouseCallBack within the while loop so the we can select the box multiple times

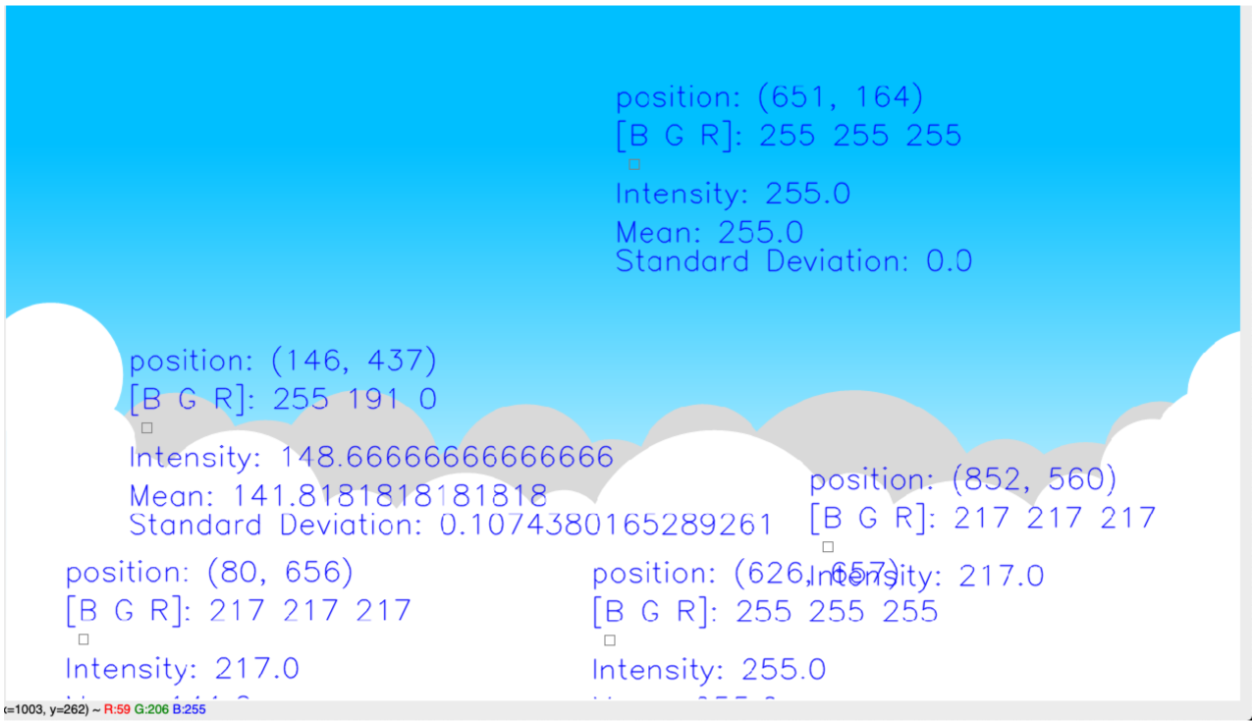
First I picked the R G B values for the pixel before calculating the intensity.
I have used grayscale to calculate the mean and standard deviation as per the formula in from the textbook.

```
import cv2
import math
import numpy as np

x1, y1 = 0, 0

original_img = cv2.imread("sky.png")

gray_image = cv2.cvtColor(original_img, cv2.COLOR_BGR2GRAY)
```

```
def on_mouse_click(event, x, y, flags, param):
    global x1, y1
    if event == cv2.EVENT_LBUTTONDOWN:
        cv2.rectangle(original_img, (x + 5, y + 5), (x - 5, y - 5), (128, 128, 128), 1)

        # Position
        cv2.putText(original_img, 'position: (' + str(x) + ', ' + str(y) + ')', (x - 20, y - 60), cv2.FONT_HERSHEY_SIMPLEX,
1, (255, 0, 0), 0, cv2.LINE_AA)

        color = original_img[y1, x1]

        # color
        cv2.putText(original_img, '[B G R]: ' + ' '.join(str(v) for v in color), (x - 20, y - 20),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 0, cv2.LINE_AA)

        # intensity
        intensity = 0
        for c in color:
            intensity = intensity + int(c)
        cv2.putText(original_img, 'Intensity: ' + str(intensity / 3), (x - 20, y + 40), cv2.FONT_HERSHEY_SIMPLEX, 1,
(255, 0, 0), 0, cv2.LINE_AA)

        # Mean
        mean = 0
        for xm in range(x - 5, x + 6):
            for ym in range(y - 5, y + 6):
                mean = mean + gray_image[xm, ym]
        mean = mean / (11 * 11)
        cv2.putText(original_img, 'Mean: ' + str(mean), (x - 20, y + 80), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0),
0, cv2.LINE_AA)

        # Standard Deviation

        std = 0
        for x_std in range(x - 5, x + 6):
            for y_std in range(y1 - 5, y1 + 6):
                s_std = std + ((gray_image[x_std, y_std] - mean) * (gray_image[x_std, y_std] - mean))
        std = math.sqrt(s_std / (11 * 11))

        cv2.putText(original_img, 'Standard Deviation: ' + str(std), (x - 20, y + 110), cv2.FONT_HERSHEY_SIMPLEX, 1,
(255, 0, 0), 0, cv2.LINE_AA)

        x1, y1 = x, y



cv2.namedWindow('Image')

while 1:
    cv2.setMouseCallback('Image', on_mouse_click)
    cv2.imshow('Image', original_img)
    k = cv2.waitKey(5) & 0xFF
    if k == 27:
        break



cv2.destroyAllWindows()
```

Section4. Observation.

We can observe that the standard deviation for a point is closer to zero all the rgb values will be more similar. When it is exactly 0 the 3 values of R, G, B will be 255 which mean the values are homogeneously distributed at that pixel as shown in the image below



```
position: (651, 164)
[B G R]: 255 255 255
Intensity: 255.0
Mean: 255.0
Standard Deviation: 0.0

position: (146, 437)
[B G R]: 255 191 0
Intensity: 148.66666666666666
Mean: 141.81818181818181818
Standard Deviation: 0.1074380165289261

position: (852, 560)
[B G R]: 217 217 217
Intensity: 217.0

position: (80, 656)
[B G R]: 217 217 217
Intensity: 217.0

position: (626, 657)
[B G R]: 255 255 255
Intensity: 255.0

=1003, y=262) ~ R:59 G:206 B:255
```

Question 2

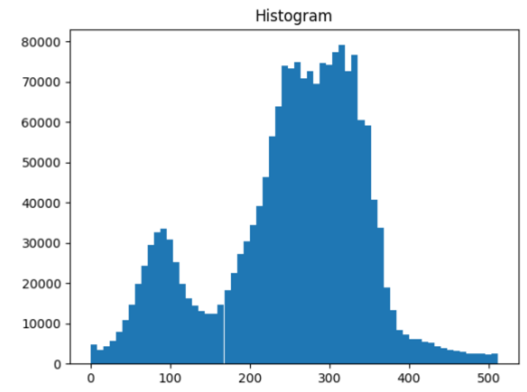
Section A

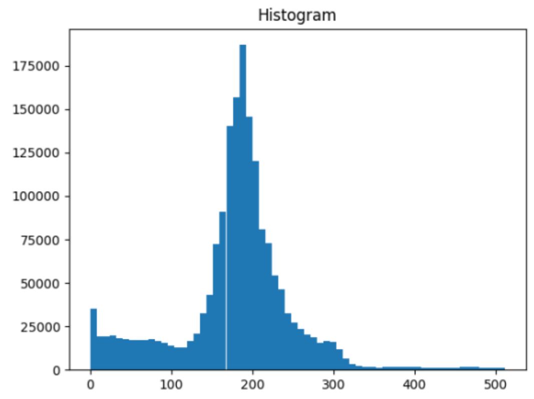
For this part I used the glob package to access the image and appended the image values into a variable original_image.
Later I loop through the original image and calculate the histogram based on the formula given in the question. I used the plt package to calculate the histogram using the command

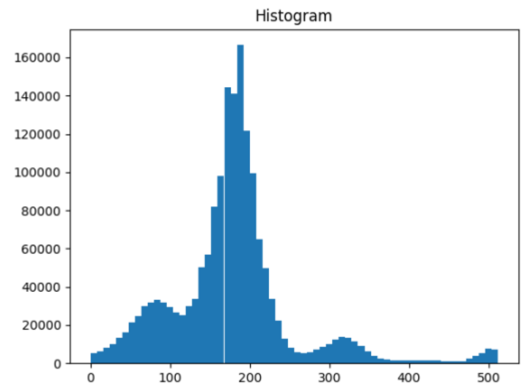
    plt.hist(array, bins= list(range(0,520,8))

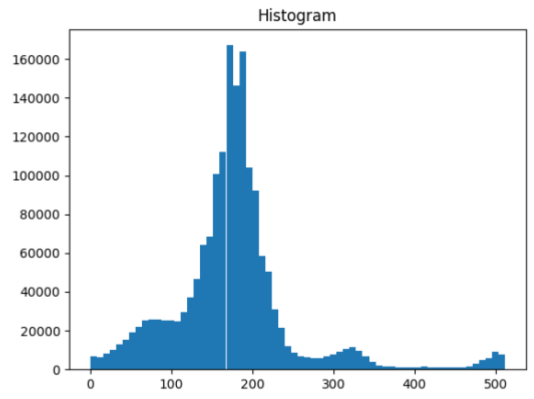
I later saved the histogram graph for each image inside an output folder HistOutput so whicle running this file, please make sure the folder “HistOutput” is available.

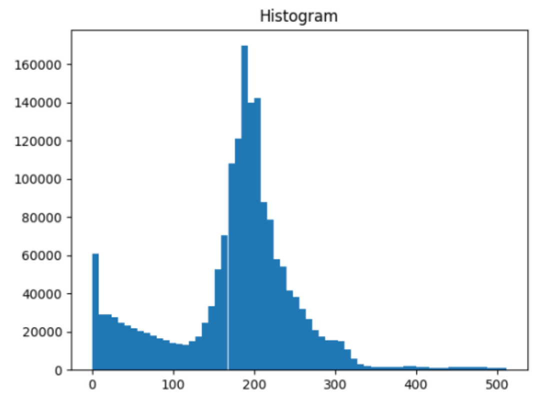
Some of the output for the images are shown below. From left to right these are the output for the first 6 images in the input folder ST2MainHall4.

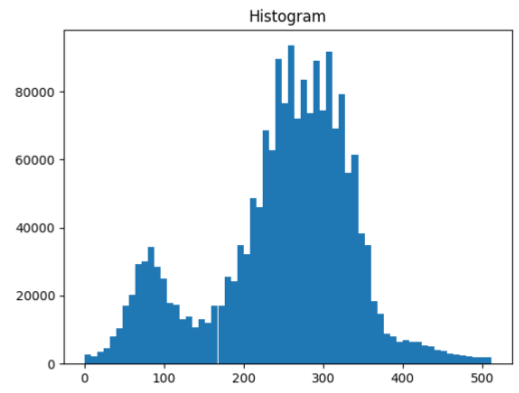












Section B
For this section I read all the images using the glob function as before and calculated the histogram intersection and chiSquare for each images and stored it into and array. I looped over 0 to 99 as there are 99 images and plotted both histogram and chiSquare.

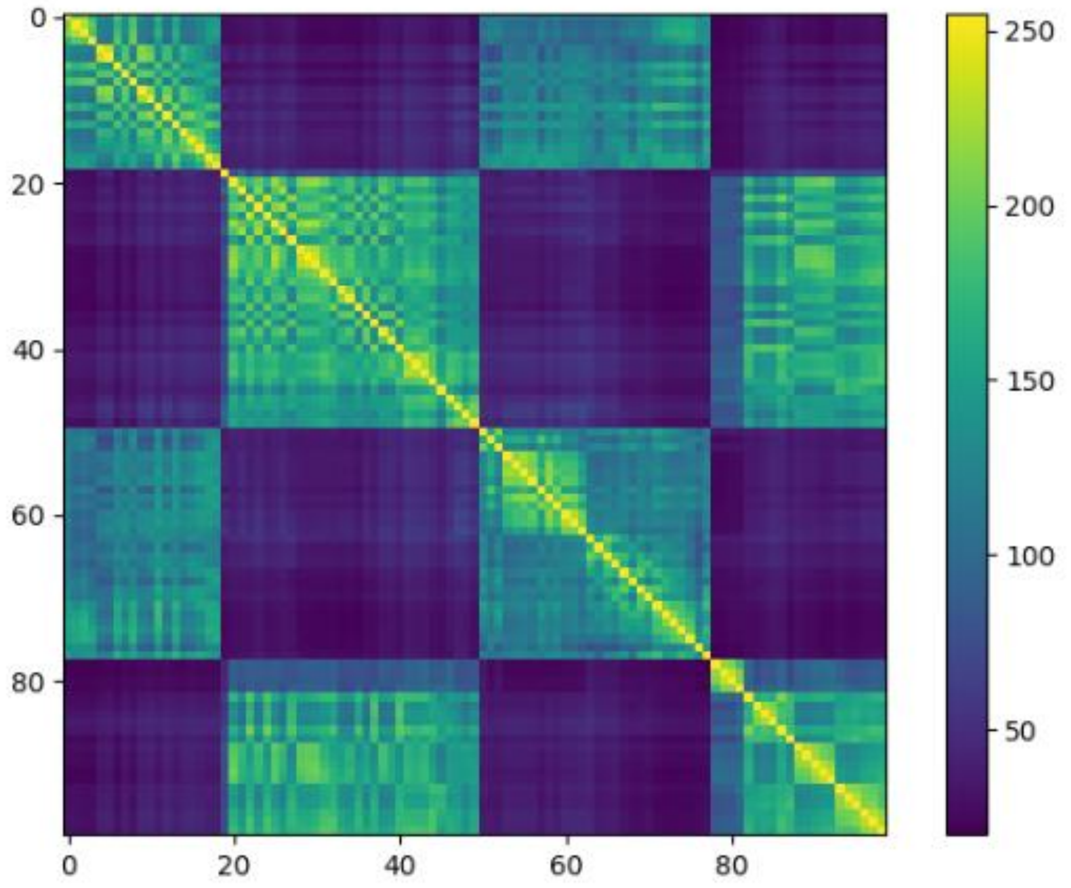
To calculate the histogram I used numpy's histogram command and the histogram intensity and chi square is calculated as the given formula in the question.

For histogram intensity the diagonal is1.0 and for chiSquare the diagonal is 0.0.

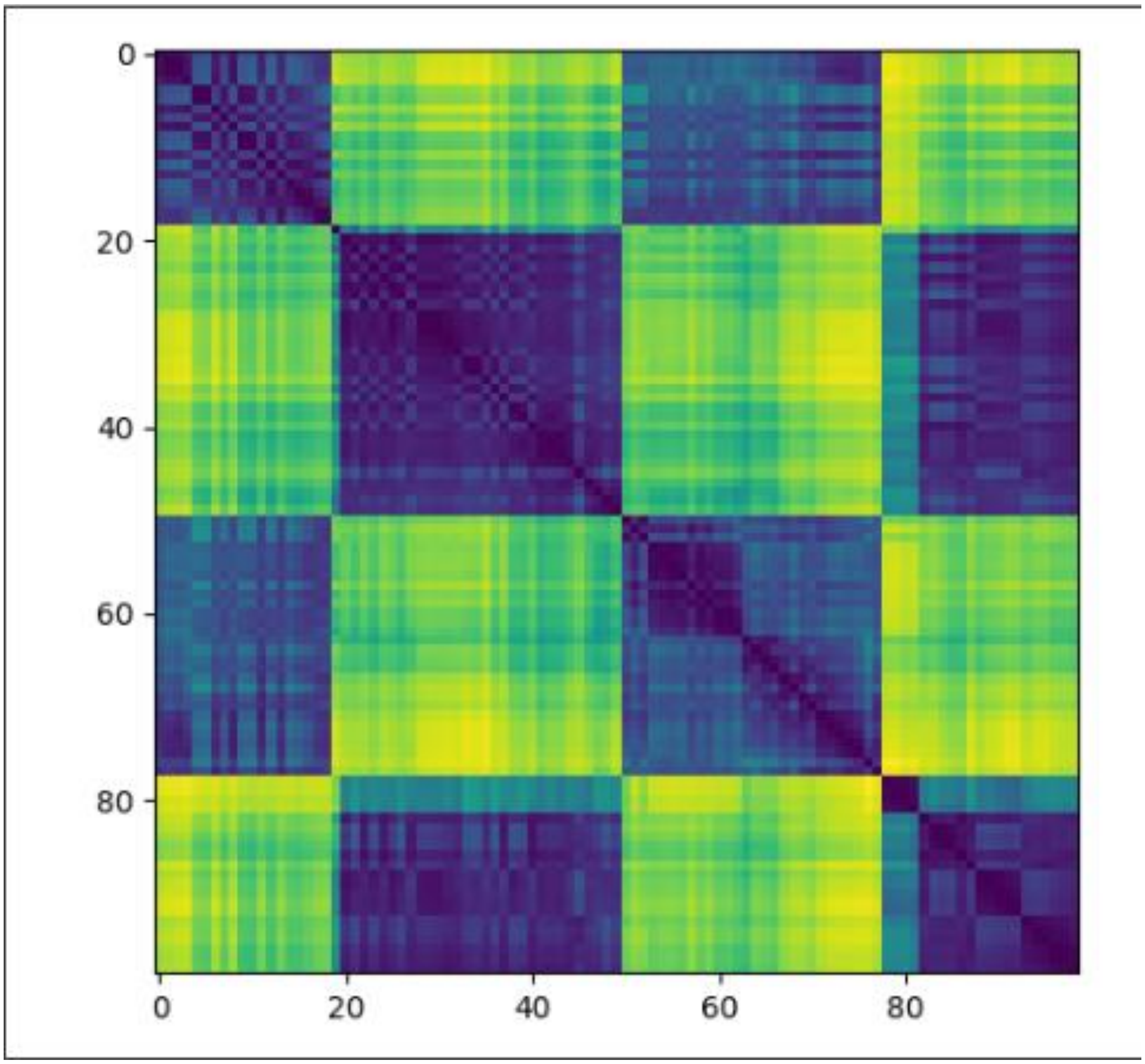
From the image we can identify the for intersection high value show higher similarity and for chi sqr lower value show higher similarity.

I used the pyploy command to plot the intensity and chi square

Intersection Plot



Chi Square Plot



```
import cv2
import glob
import numpy as np
from matplotlib import pyplot as plt


def histogramIntersection(a, b):
   min_val = np.minimum(a, b)
   max_val = np.maximum(a, b)
   intersection = np.true_divide(np.sum(min_val), np.sum(max_val))
   intersection = intersection * 255
   return intersection.astype(np.uint8)


def chiSquared(a, b):
   np.seterr(divide='ignore', invalid='ignore')
   numerator = np.square(np.subtract(a, b))
   denominator = np.add(a, b)
   chi_sq = np.nansum(np.true_divide(numerator, denominator))

   return chi_sq

def main():
   original_image = []
   histogram = []
   histo_val = []
   histo_val_inner = []


   # read images from folder
   for imageFile in sorted(glob.glob("ST2MainHall4/*.jpg")):
      img = cv2.imread(imageFile)
      print(imageFile)
      original_image.append(img)

   # calculate histogram
   for img in original_image:
      b, g, r = cv2.split(img)
```

```
        img = ((r >> 5) << 6) + ((g >> 5) << 3) + (b >> 5)
        img_histo = cv2.calcHist([img], [0], None, [512], [0, 256])
        histogram.append(img_histo)



    # calculate histo intersection and chiSquare for 99 images
    for i in range(0, 99):
        img1 = histogram[i]
        for j in range(0, 99):
            img2 = histogram[j]
            # for histogram
            # histo_val_inner.append(histogramIntersection(img1, img2))
            # for chi squar
            histo_val_inner.append(chiSquared(img1, img2))
        histo_val.append(histo_val_inner)
        histo_val_inner = []

    val = np.array(histo_val)
    plt.imshow(val)
    # fig, ax = plt.subplots()
    # ax.imshow(val)
    plt.colorbar()
    plt.show()



if __name__ == '__main__':
    main()
```