CS682 Homework 2 Report

Name: Rashika Koul
Webpage: http://mason.gmu.edu/~rkoul2/CS682/ **Username:** cs682 **Password:** 120196

Exercise 1.1:

1. The image was chosen using the askopenfilename() method from the tkinter.filedialog library and read using the cv2.imread() method.
2. The histograms of all the three color channels of the image were calculated using cv2.calcHist() method.

   ```
   b = cv2.calcHist([image],[0],None,[256],[0,256])
   g = cv2.calcHist([image],[1],None,[256],[0,256])
   r = cv2.calcHist([image],[2],None,[256],[0,256])
   ```

3. (a) The 11*11 square window was created and updated dynamically using the cv2.rectangle() method every time the event was cv2.EVENT_MOUSEMOVE. (b) The location of the mouse was obtained using the current x and y coordinates and their R, G, B values were the values in the channels 2, 1, 0 respectively. (c) The intensity value was calculated by adding the R, G, B values at the current x and y coordinates and then dividing it by 3. (d) The mean and standard deviation was calculated using the cv2.meanStdDev() method.

```
def click(event, x, y, flags, param):
    global mouseX, mouseY, image, clone
    iv = 0.0
    if event == cv2.EVENT_MOUSEMOVE:
        image = clone.copy()
        cv2.rectangle(image, (x-6,y-6), (x+6,y+6), (0, 255, 0), 1)
        mouseX, mouseY = x,y
        mean, sd = cv2.meanStdDev(image[(y-5):(y+6),(x-5):(x+6)])
        iv= (int(image[mouseY][mouseX][2])+int(image[mouseY][mouseX][1])
+int(image[mouseY][mouseX][0]))/3.0
        print("(x="+str(mouseX)+", y="+str(mouseY)+") ~ R:"+str(image[mouseY][mouseX]
[2])+" G:"+str(image[mouseY][mouseX][1])+" B:"+str(image[mouseY][mouseX][0])+" \nIntensity
value="+str(iv))
        print("Mean= R:"+str(mean[2][0])+" G:"+str(mean[1][0])+" B:"+str(mean[0][0]))
        print("Standard deviation= R:"+str(sd[2][0])+" G:"+str(sd[1][0])+" B:"+str(sd[0][0])
+"\n")

clone = image.copy()
cv2.namedWindow("image")
cv2.setMouseCallback("image", click)
while True:
    cv2.imshow("image", image)
    key = cv2.waitKey(1) & 0xFF
    if key == 27:
        break
```

4. The variance and standard deviation of all the three channels (R, G, B) of the homogeneous portions of the image should be zero and that of non-homogeneous portions can be greater than zero.

Exercise 1.2:

1. The the RGB channel image was converted to indexed image and then the histogram was calculated as follows:
   new_image = ((image[:,:,2] >> 5) << 6) + ((image[:,:,1] >> 5) << 3) + (image[:,:,0] >> 5)
   hist, bins = np.histogram(new_image, 512,[0,512])
2. The methods for calculating Histogram Intersection and Chi-square were implemented as follows:

```
def intersection(hist_1, hist_2):
    minima = np.minimum(hist_1, hist_2)
    maxima = np.maximum(hist_1, hist_2)
    intersection = np.true_divide(np.sum(minima), np.sum(maxima))
    return intersection
def chi_square(hist_1, hist_2):
    chi = 0.0
    for h1,h2 in zip(hist_1,hist_2):
        if h1+h2>=5:
            chi = chi + (((h1-h2)**2)/(h1+h2))
    return chi
```

3. All the image histogram pairs were compared as follows:

```
for h1,h2 in itertools.combinations(range(len(histograms)), 2):
    all_chi[h1][h2] = chi_square(histograms[h1],histograms[h2])
    all_chi[h2][h1] = all_chi[h1][h2]
    all_int[h1][h2] = intersection(histograms[h1],histograms[h2])
    all_int[h2][h1] = all_int[h1][h2]
```

The Histogram intersection and Chi-square values were normalised as follows:

```
min_max_scaler = preprocessing.MinMaxScaler(feature_range=(0, 255))
normalized_all_chi = min_max_scaler.fit_transform(all_chi)
normalized_all_int = min_max_scaler.fit_transform(all_int)
```

The normalized values of the 2 histograms were displayed using plt.imshow().