Name: Anowarul Kabir
G01146736
Site: http://mason.gmu.edu/~akabir4/cs682/
username: akabir4
pass: akabir4gmu135

Problem 1:
Running command “python problem_1_solution.py” will open a dialog box to select an image, and the selected image will be shown immediately.

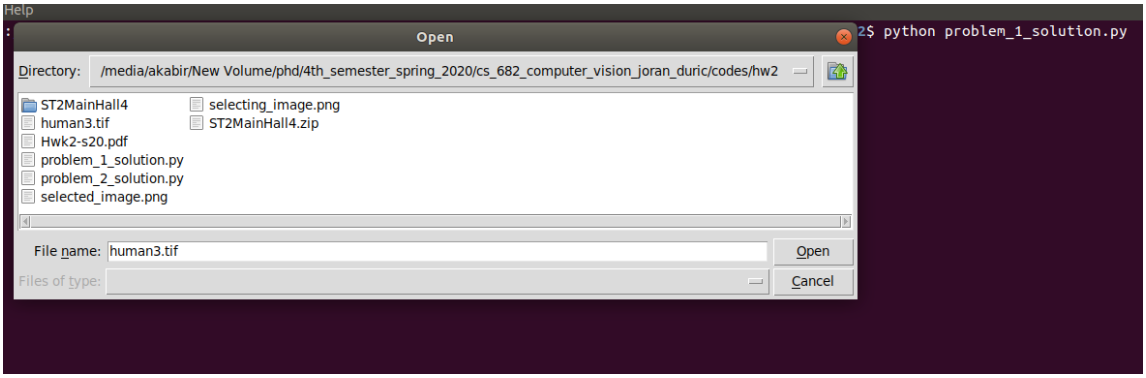

Figure 1: Image selection from dialog box

Figure 2: Showing selected image

Figure 3 shows color histograms for each channel of the selected image.

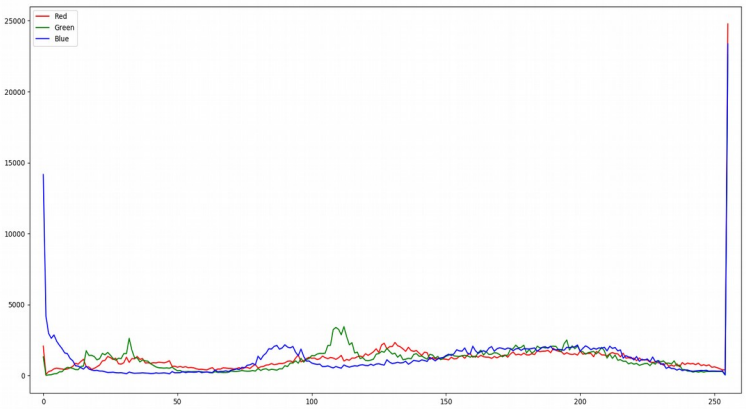

Figure 2: Color histogram for each channel

Here are some example images that shows the window of size 13x13 with outer border and 11x11 without outer border. It also shows the reference point P in the form of (x, y), intensity value, mean and standard deviation. Mean and standard deviation are computed based on 11x11 window.

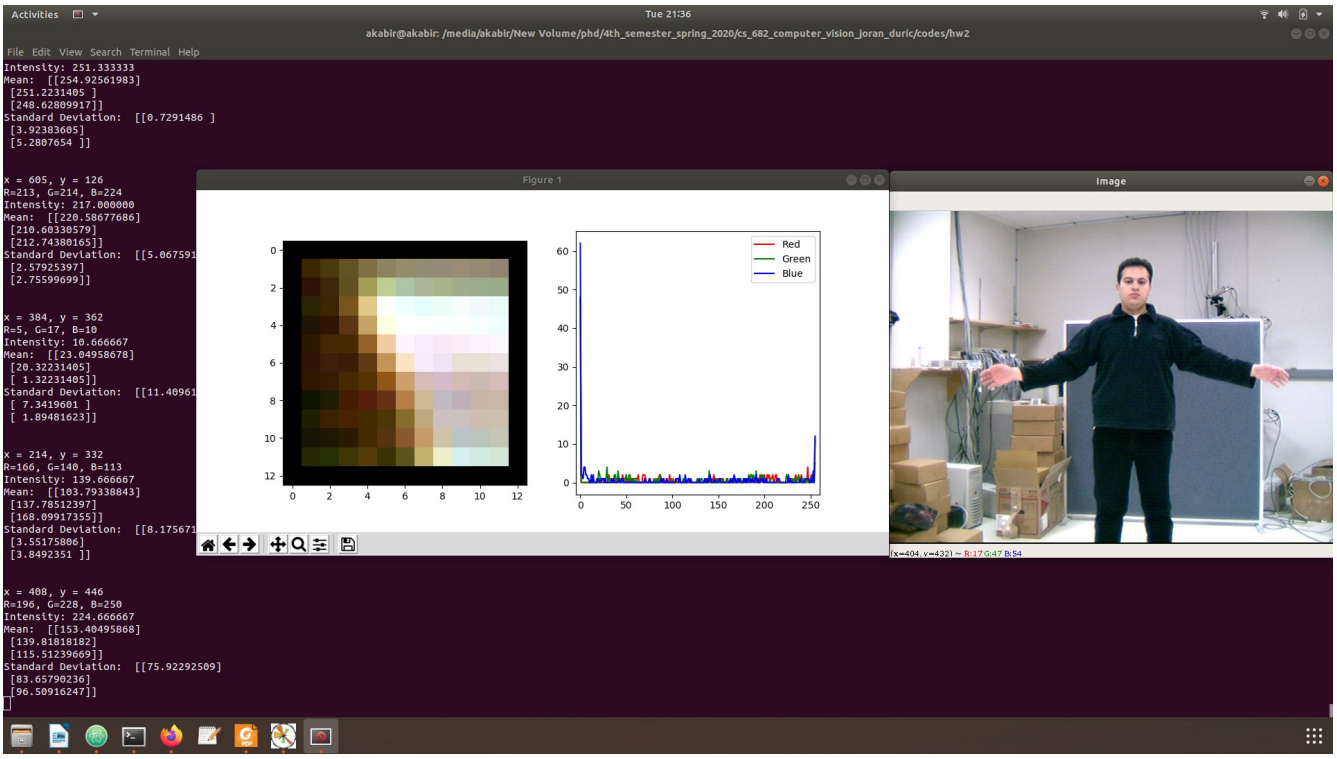

Figure 3: Homogeneous window

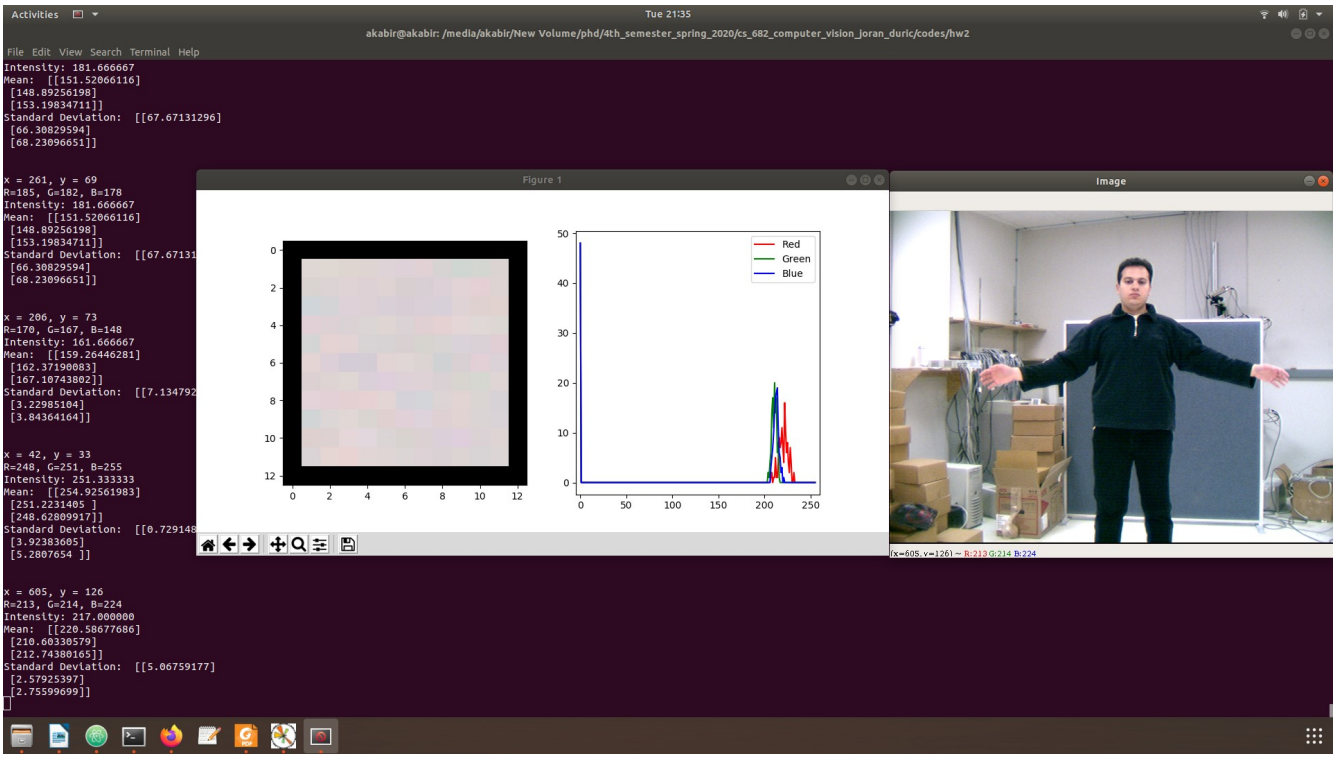

Figure 4: Inhomogeneous window

Figure 4 and 5 shows homogeneous and inhomogeneous window based on channel value distribution. In Figure 4 histogram, we can see all image values are distributed around all pixel values. On the contrary, in Figure 5 image values are distributed around close to single value. So the standard deviation is very little in comparison to Figure 4.

Problem 2:

A. To compute the color histogram in respect to 512-bins, I have written the following line of code which convert each pixel value into an index. Before doing that I changed the data type of image from *uint8* to *uint16*.

$$indexes = ((img[:, :, 0] >> 5) << 6) + ((img[:, :, 1] >> 5) << 3) + (img[:, :, 2] >> 5)$$

B. To do this I wrote two functions called hist_intersection() and chi_squared_measure() based on given equation.

C. Figure 6 shows comparisons among all images using histogram intersection (left) and chi-square measure (right).

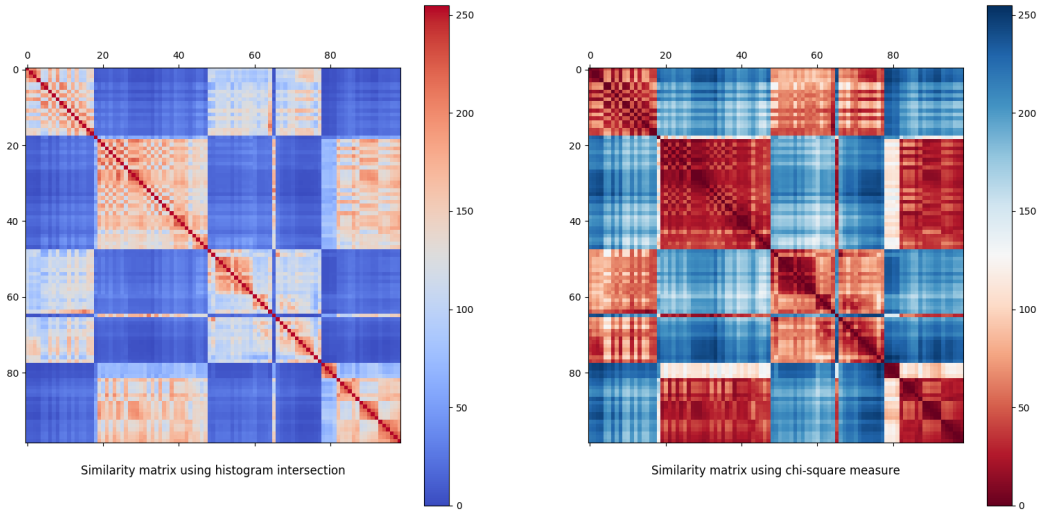

Figure 5: Comparison among images

To run and generate similarity matrix, my solution takes about 13 seconds.

Problem 1 Solution code:

```
from tkinter import *
from PIL import Image
from PIL import ImageTk
from tkinter import filedialog

import numpy as np
import cv2
import matplotlib.pyplot as plt

def draw_histogram(img):
    color = ('r', 'g', 'b')
    color_names = ['Red', 'Green', 'Blue']
    for i, col in enumerate(color):
        histr = cv2.calcHist([img], [i], None, [256], [0, 256])
        plt.plot(histr, color=col, label=color_names[i])
        plt.xlim([-5, 260])
    plt.legend()
    plt.show()

# the following lines are necessary for draw_img_and_hist() function
plt.ion()
fig = plt.figure(figsize=(10, 3))
ax1 = fig.add_subplot(1, 2, 1)
ax2 = fig.add_subplot(1, 2, 2)

def draw_img_and_hist(img):
    ax1.imshow(img)
    ax2.cla()
    color = ('r', 'g', 'b')
    color_names = ['Red', 'Green', 'Blue']
    for i, col in enumerate(color):
        histr = cv2.calcHist([img], [i], None, [256], [0, 256])
        ax2.plot(histr, color=col, label=color_names[i])
        plt.xlim([-5, 260])
    plt.legend()
    plt.draw()
    plt.pause(0.001)

def on_click(event, x, y, flags, params):
    print("\nPress any key to stop.\n")
    if event == cv2.EVENT_MOUSEMOVE:  # EVENT_LBUTTONDOWN
        print('x = %d, y = %d' % (x, y))  # x = cols, y = rows
        B = int(I[y, x, 0])
        G = int(I[y, x, 1])
        R = int(I[y, x, 2])
        print('R=%d, G=%d, B=%d' % (R, G, B))
        intensity = (R + G + B) / 3
        print("Intensity: %f" % (intensity))
        rows, cols = I.shape[:2]
        w_rows, w_cols = 13, 13
        k_row, k_col = int(w_rows / 2), int(w_cols / 2)
        if(x - k_col < 0 or y - k_row < 0 or x + k_col > cols or y + k_row > rows):
            print("out of boundary\n")
            return
        else:
            window = I[y - k_row:y + k_row + 1, x - k_col:x + k_col + 1]
            window[:1, :] = 0  # 1st row
            window[12:, :] = 0  # last row
            window[:, :1] = 0  # 1st col
            window[:, 12:] = 0  # last col
            draw_img_and_hist(window)

        only_window = window[1:12, 1:12]
        # cv2.imshow("Window", only_window)
```

```
        mean, stddev = cv2.meanStdDev(only_window)
        print("Mean: ", mean)
        print("Standard Deviation: ", stddev)
        # draw_histogram(only_window)

def main():
    global I
    root = Tk()
    root.withdraw()  # to hide small tk window
    path = filedialog.askopenfilename()
    I = cv2.imread(path)
    cv2.imshow("Image", I)
    # draw_histogram(I)
    cv2.setMouseCallback('Image', on_click)

main()
plt.show()
cv2.waitKey(0)
```

Problem 2 Solution code:

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
import os
import random
import time

start = time.time()


def draw_color_histogram():
    # 'ST2MainHall4/ST2MainHall4001.jpg'
    directory = 'ST2MainHall4/'
    all_hists = []
    label = []
    for filename in os.listdir(directory):
        print(filename)
        temp_img = cv2.imread(directory + filename)
        # temp_img = cv2.imread('ST2MainHall4/ST2MainHall4001.jpg')
        img = np.zeros(shape=temp_img.shape, dtype=np.uint16)
        img = img + temp_img
        indexes = ((img[:, :, 0] >> 5) << 6) + \
            ((img[:, :, 1] >> 5) << 3) + (img[:, :, 2] >> 5)
        # hist,bins = np.histogram(indexes.ravel(),512,[0,512])
        all_hists.append(indexes.ravel())
        label.append(filename)

    bins = np.linspace(-5, 520, 15)
    # print(bins)
    plt.hist(all_hists, bins=bins)
    # plt.legend(loc="upper right")
    plt.title('Histograms')
    plt.show()


def scale(X, x_min, x_max):
    mat_min = X.min()
    mat_max = X.max()
    nom = (X - mat_min) * (x_max - x_min)
    denom = mat_max - mat_min
    return x_min + nom / denom
```

```
def compute_color_histogram(temp_img):
    img = np.zeros(shape=temp_img.shape, dtype=np.uint16)
    img = img + temp_img
    indexes = ((img[:, :, 0] >> 5) << 6) + \
        ((img[:, :, 1] >> 5) << 3) + (img[:, :, 2] >> 5)
    hist, bins = np.histogram(indexes.ravel(), 512, [0, 512])
    return hist, bins


def hist_intersection(hist1, hist2):
    sum_of_min, sum_of_max = 0, 0
    for i in range(hist1.shape[0]):
        sum_of_min += min(hist1[i], hist2[i])
        sum_of_max += max(hist1[i], hist2[i])
    return sum_of_min / sum_of_max


def chi_squared_measure(hist1, hist2):
    chi_square = 0
    for i in range(hist1.shape[0]):
        if hist1[i] + hist2[i] > 0:
            chi_square += ((hist1[i] - hist2[i])**2) / (hist1[i] + hist2[i])
    return chi_square


directory = 'ST2MainHall4/'
all_hists = []
for filename in os.listdir(directory):
    print(filename)
    temp_img = cv2.imread(directory + filename)
    hist, _ = compute_color_histogram(temp_img)
    # print(hist.shape)
    all_hists.append(hist)

num_of_imgs = len(all_hists)
# confusion_matrix_using_hist_intersection
cmuhi = np.zeros(shape=(num_of_imgs, num_of_imgs))
# confusion_matrix_using_chi_squared_measure
cmucsm = np.zeros(shape=(num_of_imgs, num_of_imgs))
for i in range(num_of_imgs):
    for j in range(num_of_imgs):
        cmuhi[i, j] = hist_intersection(all_hists[i], all_hists[j])
        cmucsm[i, j] = chi_squared_measure(all_hists[i], all_hists[j])

cmuhi = scale(cmuhi, 0, 255)
cmucsm = scale(cmucsm, 0, 255)

fig, axes = plt.subplots(ncols=2, figsize=(20, 20))
ax1, ax2 = axes

im1 = ax1.matshow(cmuhi, cmap='coolwarm')
im2 = ax2.matshow(cmucsm, cmap='RdBu')

ax1.set_title('Similarity matrix using histogram intersection', y=-.1)
ax2.set_title('Similarity matrix using chi-square measure', y=-.1)

fig.colorbar(im1, ax=ax1)
fig.colorbar(im2, ax=ax2)

end = time.time()
print("Time: ", end - start)
plt.show()
cv2.waitKey(0)
```