CS682 Homework 3 Report

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

Name: Rashika Koul

1. To compute gray level edges the images were first smoothed and then converted to gray scale. Then the sobel operator was applied to get x and y gradients. To compute r, g, b edges the color image was split into r, g, b channels and x and y gradients was calculated for each channel using sobel operator.

```
image = cv2.blur(image,(5,5))
gray_image = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
grx = cv2.Sobel(gray_image,cv2.CV_64F,1,0,ksize=3)
gry = cv2.Sobel(gray_image,cv2.CV_64F,0,1,ksize=3)
b = image[:,:,0]
g = image[:,:,1]
r = image[:,:,2]
bx = cv2.Sobel(b,cv2.CV_64F,1,0,ksize=3)
by = cv2.Sobel(b,cv2.CV_64F,0,1,ksize=3)
gx = cv2.Sobel(g,cv2.CV_64F,1,0,ksize=3)
gy = cv2.Sobel(g,cv2.CV_64F,0,1,ksize=3)
rx = cv2.Sobel(r,cv2.CV_64F,1,0,ksize=3)
ry = cv2.Sobel(r,cv2.CV_64F,0,1,ksize=3)
```

2. To compute the magnitude and angle of the gray images, cv2.cartToPolar() method was used. The angle obtained was then divided by 10 and rounded off before plotting the histogram using value 1.

```
magg, angleg = cv2.cartToPolar(grx, gry, angleInDegrees=True)
angleg = np.uint8(np.rint(np.divide(angleg,10)))
hist, bins = np.histogram(angleg,36)
gray_histogram.append(hist)
```

3. To compute the angle for color images, cv2.cartToPolar() method was used. This angle was also divided by 10 and rounded off. To compute the magnitude for color images the given formula was used and all magnitudes < 5 were ignored. The histogram was plotted using the magnitudes for each angle.

```
magc,anglec = cv2.cartToPolar(bx+gx+rx,by+gy+ry, angleInDegrees=True)
anglec = np.uint8(np.rint(np.divide(anglec,10)))
magc = abs(bx)+abs(gx)+abs(rx)+abs(by)+abs(gy)+abs(ry)
magc[magc<5] = 0
histc, binsc = np.histogram(anglec,36, weights = magc)
color_histogram.append(histogram)
```

4. 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
```

5. 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().

Bonus Questions:

1. Quiver function was used to visualise the gradients of grey image as follows:

```
X, Y = np.meshgrid(np.arange(1600), np.arange(1200))
plt.figure()
plt.subplot(221), plt.quiver(X[::50,::50], Y[::50,::50], np.uint8(grx[::50,::50]), np.uint8(gry[::50,::50]),
units='width'), plt.title('Scene 1 Gray Image Orientation')
plt.show()
```

2 and 3. The Eigen vectors and Eigen values were calculated and gradient magnitude and orientation was calculated using lumpy arrays as follows:

```
a = rx**2 + gx**2 + bx**2
b = rx*ry + gx*gy + bx*by
c = ry**2 + gy**2 + by**2
lamda = 1/2*((a+c)+np.sqrt((a+c)**2 - 4*(b**2-a*c)))
x = np.divide(-b,a-lamda)
mage, anglee = cv2.cartToPolar(x, np.ones(x.shape), angleInDegrees=True)
mage = lamda
anglee = np.uint8(np.rint(np.divide(anglee,10)))
histe, binse = np.histogram(anglee,36, weights = mage)
eigen_histogram.append(histograme)
```