**CS 682 Home Work II**

**KRISHNA SINDHURI VEMURI**

**G01024066**

**Part 1**

The tkinter and os libraries are used for the file dialog using which desired image can be loaded.

import tkinter as tk

import numpy as np

import cv2

from tkinter import filedialog

import os

from matplotlib import pyplot as plt

I have defined a function mouse\_move that creates a new image window of size 13\*13(11\*11 is 2 pixels curving on all sides) when left button is clicked. The window shows the outer border of the pixel that is selected. The location of the pixel, the RGB values at the pixel location, the intensity(R+G+B/3) at that pixel location, the mean and standard deviation of the 11\*11(without curving ) are calculated and printed.

def mouse\_move(event,x,y,flags,param):

while(1):

if event == cv2.EVENT\_LBUTTONDOWN:

crop\_img = img[y-5:y+5,x-5:x+5]

constant= cv2.copyMakeBorder(crop\_img,2,2,2,2,cv2.BORDER\_CONSTANT,value=(255,255,255))

cv2.namedWindow('cropped',cv2.WINDOW\_NORMAL)

cv2.imshow('cropped', constant)

pix=[x,y]

print('The location of the pixel',pix)

b,g,r=img[x,y]

R=np.uint16(r)

G=np.uint16(g)

B=np.uint16(b)

print ('The R G B values of the pixel are',R,G,B)

print ('The intensity of the pixel are',float((R + G + B) / 3))

mean,sdeviation=cv2.meanStdDev(crop\_img)

bm,gm,rm=mean

bs,gs,rs=sdeviation

print ('The mean of the 11 \* 11 window is',rm,gm,bm)

print ('The standard deviation of the 11 \* 11 window is',rs,gs,bs)

if cv2.waitKey(20) & 0xFF == 27:

break

I have defined my\_filetypes that only accepts the following kinds of image types as input.

my\_filetypes = [('png', '.png'),('bnp', '.bnp'),('tiff', '.tiff'),('jpg','.jpg')]

answer = filedialog.askopenfilename(initialdir=os.getcwd(),

title="Please select an image:",

filetypes=my\_filetypes)

img = cv2.imread(answer)

cv2.imshow('image',img)

cv2.waitKey(0)

By using cv2.split() the color spaces are differentiated using which the histograms are built

channels = cv2.split(img)

colors = ("b", "g", "r")

for(channel, c) in zip(channels, colors):

histogram = cv2.calcHist([channel], [0], None, [256], [0, 256])

plt.plot(histogram, color = c)

plt.xlim([0, 256])

plt.show()

The mouse\_move function is called on the image defined. While true, open the image and close the image window when close button the image is clicked.

cv2.namedWindow('image',cv2.WINDOW\_NORMAL)

cv2.setMouseCallback('image',mouse\_move)

while(1):

cv2.imshow('image',img)

cv2.resizeWindow('image',500,500)

cv2.waitKey(0)

if cv2.waitKey(20) & 0xFF == 27:

break

cv2.destroyAllWindows()

**Part 2**

The glob library is used to access the images in a folder.

import cv2

import numpy as np

from matplotlib import pyplot as plt

import glob

This function calculates the histogram intersection by taking two 512 bin histograms as input and performing the following function



def calculate\_intersection(h1,h2):

sum\_min=0

sum\_max=0

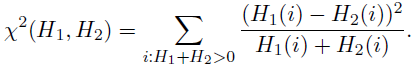
for i in range(0,512):

sum\_min += min(h1[i],h2[i])

sum\_max += max(h1[i],h2[i])

return float(sum\_min/sum\_max)

This function calculates the chi-squared measure by taking two 512 bin histograms as input and performing the following function



def calculate\_chi\_square(h1,h2):

chi\_square\_distance = 0

for i in range(0,512):

if (h1[i] + h2[i]) > 5:

chi\_square\_distance += (((h1[i] - h2[i])\*\*2)/float(h1[i] + h2[i]))

return chi\_square\_distance

The value of the intersection is scaled by multiplying it with 255 and the resulting integer value is stored as result.

def color\_intersection(intersection):

return int(255\*intersection)

To scale the chi-square measure result, I am dividing the result with 2000 because I could not find a better way to do it.

def color\_chi\_square(chi\_square):

return int(chi\_square/2000)

This function used to create an index using the [((r/32)\*64)+((g/32)\*8)+(b/32)] formula, this index is used to plot histograms

def create\_index(img):

image = cv2.imread(img)

b,g,r = cv2.split(image)

B= np.uint16(b)

G= np.uint16(g)

R= np.uint16(r)

index = (((R>>5)<<6)+((G>>5)<<3)+(B>>5))

return index

The following code plots histograms for all the 99 images, stores the histogram values of each image in the list all\_hist. Also, all the histograms are saved.

all\_hist=[]

for img in glob.glob('ST2MainHall/\*.jpg'):

hist,bins = np.histogram(create\_index(img),512,[0,511])

plt.plot(hist)

plt.savefig(str(img)+'\_histogram.png')

plt.clf()

all\_hist.append(hist)

print("Saved all histograms")

The matrices store the histogram intersection and chi-squared measure results for each combination of images.

intersection\_matrix = np.zeros((99,99))

chi\_square\_matrix = np.zeros((99,99))

#For all image combinations of the images we calculate the histogram intersection and chi\_square measure

#Also the resultant value is changed to a color value using the color\_intersection and color\_chi\_square function

for i in range(0,99):

for j in range(0,99):

intersection = calculate\_intersection(all\_hist[i],all\_hist[j])

intersection\_matrix[i][j]= color\_intersection(intersection)

chi\_square = calculate\_chi\_square(all\_hist[i],all\_hist[j])

chi\_square\_matrix[i][j]= color\_chi\_square(chi\_square)

print("Histogram Intersection Done")

#Save and plot the Histogram Intersection Comparison

plt.imshow(intersection\_matrix,cmap='rainbow',interpolation='nearest')

plt.colorbar()

plt.savefig('Intersection\_Comparison.png')

plt.title('Intersection Comparison')

plt.show()

plt.clf()

#Save and plot the Chi-Square Comparison

print("Chi-Square Measure Done")

plt.imshow(chi\_square\_matrix,cmap='rainbow',interpolation='nearest')

plt.colorbar()

plt.savefig('Chi\_Square\_Comparison.png')

plt.title('Chi Square Comparison')

plt.show()

plt.clf()

The final results are changed to color using the cmap=’rainbow’ function and thus we can see the color results after scaling the values. The visualization of the histogram intersection , shows the similarities and dissimilarities in the images. The value 255 represents perfect match and the values 0 represents perfect mismatch. Similarly, the visualization of the chi-squared measure represents 0 for match and 255 for mismatch. These visualizations are stored as .png files.