**CS 682: COMPUTER VISION**

**Abhishek Bodas**

**G01160204**

**HW2**

**Website:** <http://mason.gmu.edu/~abodas/vision/>

Username: CS682

Password: abodas682

**Ex 1.1)**

Open an image from the dialog box by entering the name or selecting the file. Creating a 11x11 window and on the mouse cursor movement, displaying outer border, coordinates, BGR values, mean and standard deviation of the pixel at current cursor position.

**Code:**

#Script to display BGR values,intensity,mean,standard deviation,outer border of pixel at mouse move

import cv2

import easygui

#To open image from dialog box

path = easygui.enterbox() #"way.png" or as specified in movethecursor() #To open image from dialog box

image = cv2.imread(path,1) #To read the image

def **movethecursor**(event,x,y,flags,param): #Function to compute values at mousemove

global x\_axis, y\_axis

if event == cv2.EVENT\_MOUSEMOVE: #checks mouse move condition

y\_axis,x\_axis = y,x

image=cv2.imread(*"way.png"*,1)

copy=image.copy()

Bluecolor=copy[y,x,0] #To compute coordinates and channels of image

Greencolor=copy[y,x,1]

Redcolor=copy[y,x,2]

intensity=(int(Bluecolor)+int(Greencolor)+int(Bluecolor))/3 #To compute intensity of pixel

window = cv2.getRectSubPix(copy, (11, 11), (x, y))

rect=cv2.rectangle(copy, (x\_axis - 5, y\_axis - 6), (x\_axis + 6, y\_axis + 5), (0,0,0),1) #To create a rectangular window

mean,std= cv2.meanStdDev(window)

cv2.imshow(*'image'*, rect)

resizedwindow = cv2.resize(window, (window.shape[0]\*50,window.shape[1]\*50)) #To increase window size for better view

cv2.imshow(*'cursor window'*, resizedwindow)

print(*"Coordinates y:"*,y,*"x:"*,x,*"Blue:"*,Bluecolor,*"Green: "*,Greencolor,*"Red: "*,Redcolor,*"Intensity:"*,int(intensity),

*"bluemean:"*,int(mean[0]),*"greenmean:"*,int(mean[1]),*"redmean:"*,int(mean[2]),*"bluestd:"*,int(std[0]),*"greenstd:"*,int(std[1]),*"redstd:"*,int(std[2]))

cv2.namedWindow(*'image'*)

cv2.imshow(*'image'*,image)

cv2.setMouseCallback(*'image'*,movethecursor) #Mouse handler for the window

key=cv2.waitKey(10000) & 0xFF #Mask for 64-bit systems

if key==27: #Press Escape key to close the image window

cv2.destroyAllWindows()

elif key==ord(*'q'*): #Press 'q' key to quit the image window

cv2.destroyAllWindows()

elif key==ord(*'e'*):#Press 'e' key to exit the image window

cv2.destroyAllWindows()

elif key==ord(*'x'*):#Press 'x' key to cancel the image window

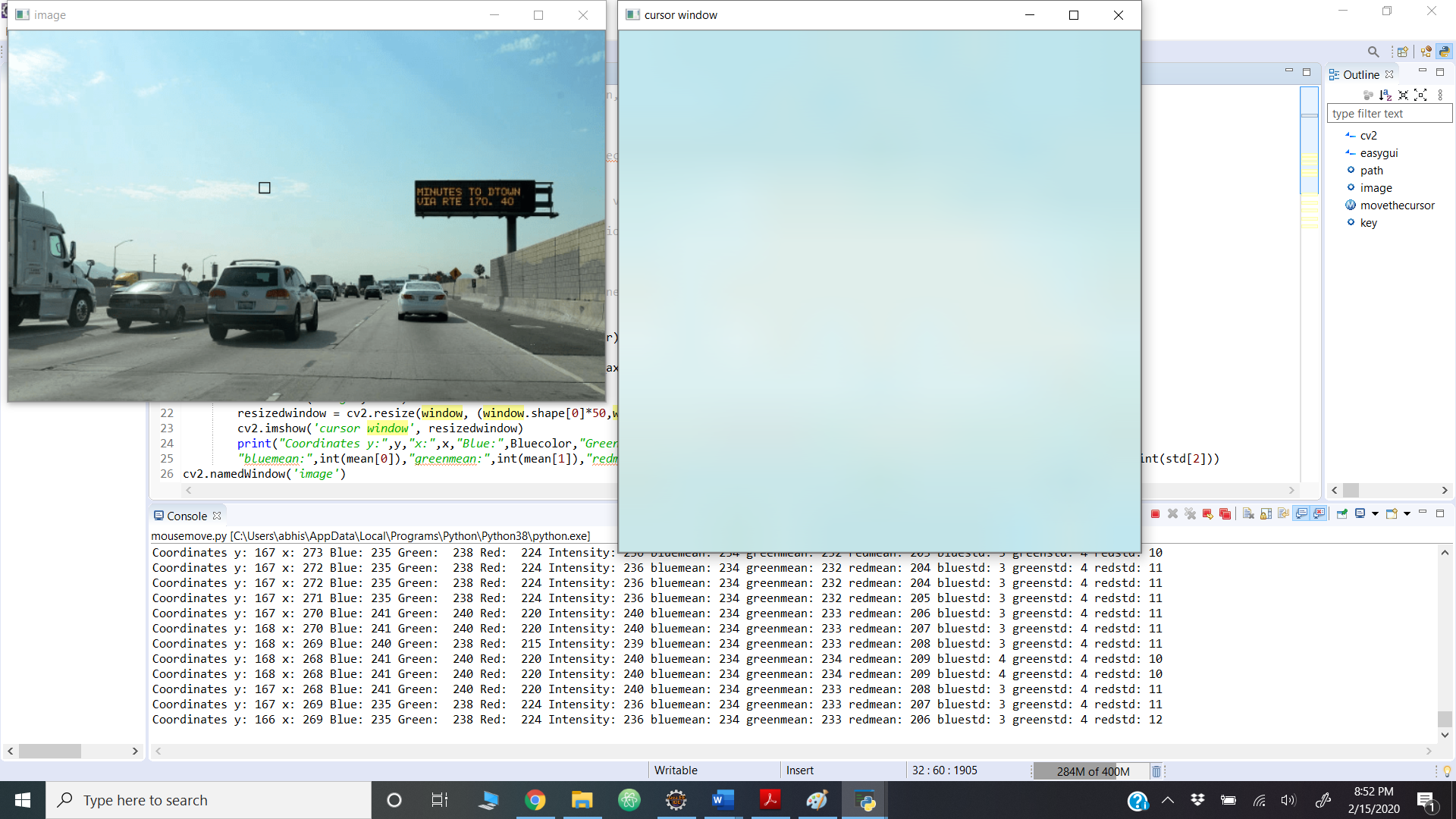
cv2.destroyAllWindows()

cv2.destroyAllWindows() #To destroy windows anyway

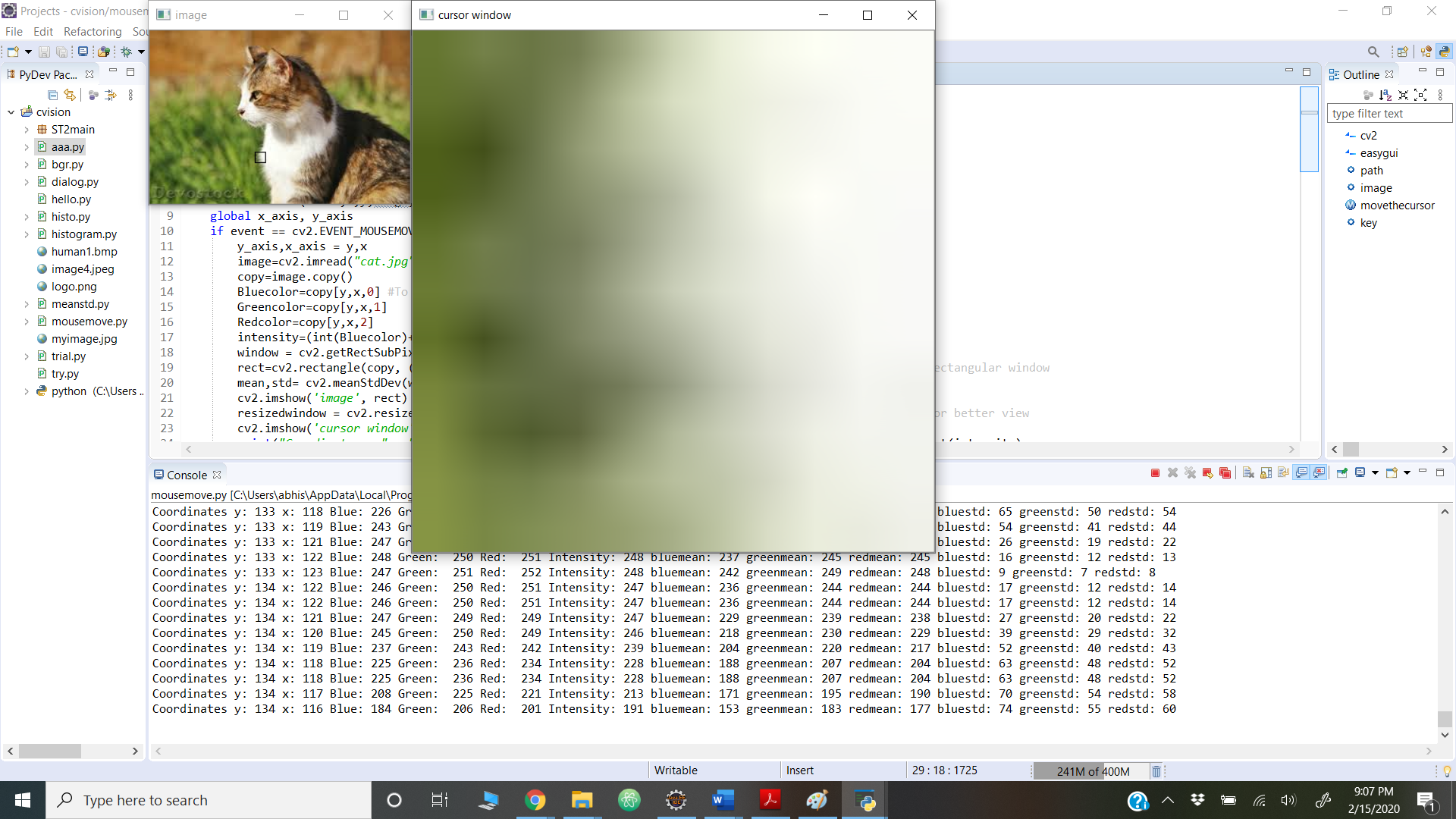
cv2.setMouseCallback(*'image'*,movethecursor)

**Output images:**

**PNG:**



**JPEG:**



**Code:**

Color Histogram of all 3 channels of a .png and .jpeg image on mouse movement

#Script to display color histogram of pixel at mouse move

import cv2

import easygui

from matplotlib import pyplot as plt #to import pyplot module

#To open image from dialog box

path = easygui.fileopenbox() #"way.png" or as specified in movethecursor()

image = cv2.imread(path,1) #To read the image

def **ihistogram**(image): #Function to calculate and display histogram

color = (*'b'*,*'g'*,*'r'*) #Tuple of colors

for i,col in enumerate(color): #To plot histogram of all 3 channels

hist = cv2.calcHist([image],[i],None,[256],[0,256]) #To calculate histogram

plt.plot(hist,color = col)

plt.xlim([0,256]) #To set the x limits of left and right

plt.text(0.5, 24000,*"HISTOGRAM"*, bbox=dict(facecolor=*'yellow'*, alpha=0.5)) #To display text on histogram

plt.show()

def **movethecursor**(event,x,y,flags,param): #Function to compute values at mousemove

global x\_axis, y\_axis

if event == cv2.EVENT\_MOUSEMOVE: #checks mouse move condition

y\_axis,x\_axis = y,x

image=cv2.imread(*"cat.jpg"*,1)

copy=image.copy()

window = cv2.getRectSubPix(copy, (11, 11), (x, y))

rect=cv2.rectangle(copy, (x\_axis - 5, y\_axis - 6), (x\_axis + 6, y\_axis + 5), (0,0,0),1) #To create a rectangular window

cv2.imshow(*'image'*, rect)

resizedwindow = cv2.resize(window, (window.shape[0]\*50,window.shape[1]\*50)) #To increase window size for better view

cv2.imshow(*'cursor window'*, resizedwindow)

cv2.imshow(*'h'*,ihistogram(window))

cv2.namedWindow(*'image'*)

cv2.imshow(*'image'*,image)

cv2.setMouseCallback(*'image'*,movethecursor) #Mouse handler for the window

key=cv2.waitKey(5000) & 0xFF #Mask for 64-bit systems

if key==27: #Press Escape key to close the image window

cv2.destroyAllWindows()

elif key==ord(*'q'*): #Press 'q' key to quit the image window

cv2.destroyAllWindows()

elif key==ord(*'e'*):#Press 'e' key to exit the image window

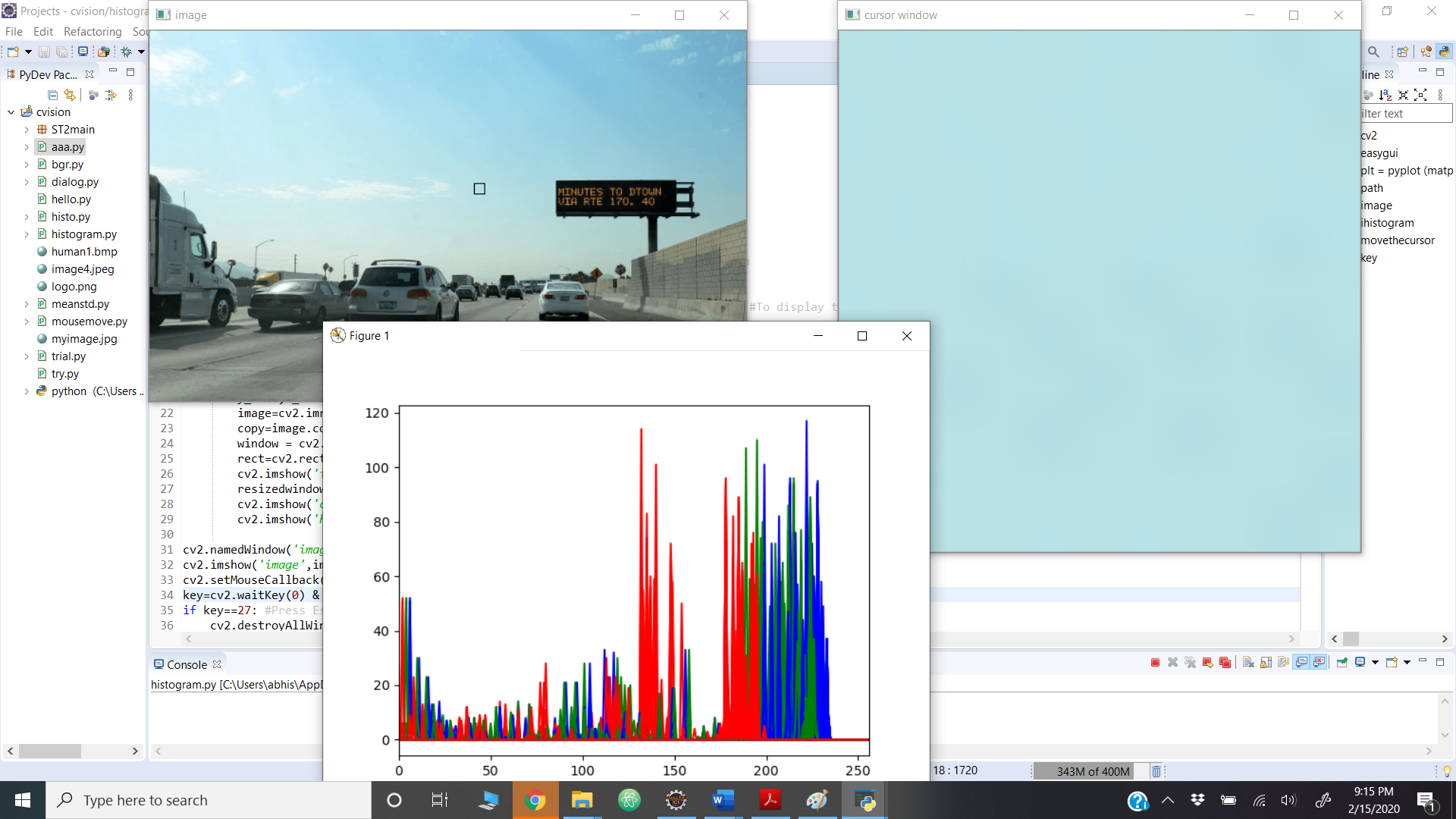
cv2.destroyAllWindows()

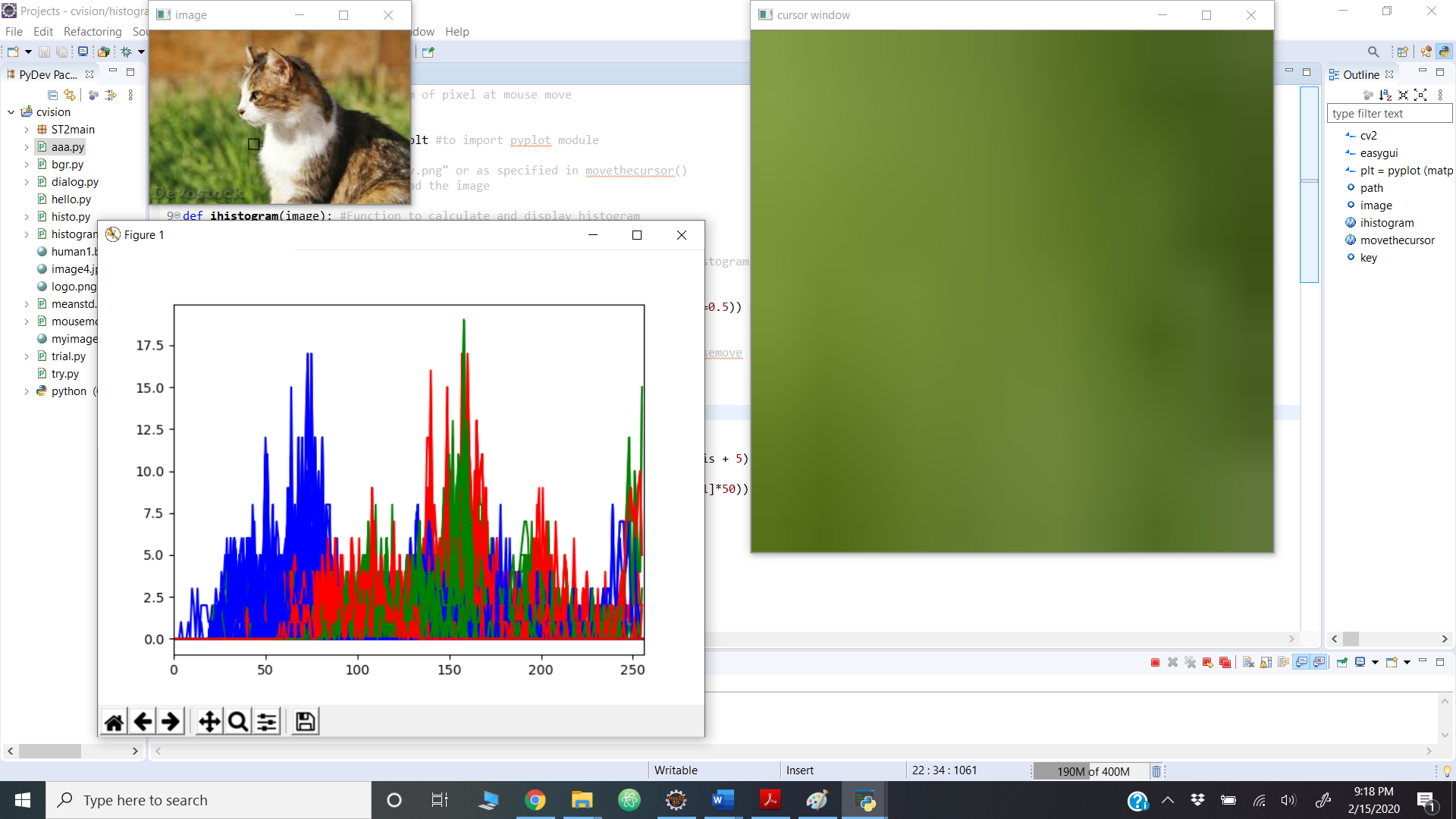
elif key==ord(*'x'*):#Press 'x' key to cancel the image window

cv2.destroyAllWindows()

cv2.destroyAllWindows() #To destroy windows anyway

cv2.setMouseCallback(*'image'*,movethecursor)

**Output-images:png**

**Jpeg:**

**HOMOGENOUS VS NON-HOMOGENOUS**

By comparing the above 2 histograms of png and jpeg images, it can be observed that the histogram of png image shows homogenous distribution of image values, whereas the histogram of jpeg image a shows non-homogenous distribution of image values. It can be determined by the logic that in homogeneity, the color channels of histograms appear to be distinct, there is a certain level of uniformity. Whereas in non-homogeneity, the color channels appear to be scattered. The uniformity is not maintained.

**Ex1.2)**

Extracting all the images from a directory and building a colored 512-bin histograms with bit shifted operation on color values. Writing 2 histogram comparison functions: Intersection and chi squared and using these two functions, displaying a linearly scaled plot of image pairs.

**Code:**

#Script to display 512 bin color histogram, intersection matrix and chi squared matrix

import cv2

import os #To import os module

import glob #To import glob module

import numpy as np

from matplotlib import pyplot as plt

directory = *"C:/Users/abhis/OneDrive/Desktop/Projects/cvision/ST2main"* #Location of Unziped folder ST2MainHall4

path = os.path.join(directory,*'\*g'*)

allimages = glob.glob(path)

imagelist = []

for i in allimages:

eachimage = cv2.imread(i)

imagelist.append(eachimage) #List of all images

histlist=[]

for index, eachimage in enumerate(imagelist):

(height, width, channels) = imagelist[index].shape

blue=imagelist[index][:,:,0]

green=imagelist[index][:,:,1]

red=imagelist[index][:,:,2]

index= np.zeros((height,width), dtype=*'uint8'*)

index= ((red>>5)<<6) + ((green>>5)<<3) + ((blue>>5)) #bit shifts

indexhist=cv2.calcHist([index], [0],None, [512], [0,512])

histlist.append(indexhist) #List of all histograms

for h in histlist:

plt.plot(h)

plt.xlim(0)

plt.text(400, 700000,*"COLOR HISTOGRAM"*, bbox=dict(facecolor=*'yellow'*, alpha=0.5)) #To display text on histogram

plt.show()

def **hintersection**(h1,h2): #histogram comparison function by calculating intersection

hmin = np.sum(np.minimum(h1, h2))

hmax = np.sum(np.maximum(h1, h2))

return float(hmin/hmax)

def **hchisquared**(h1,h2): #histogram comparison function by calculating chi square

hchi=0

for i in range(0,len(h1)):

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

hchi+=(((h1[i]- h2[i])\*\*2)/float(h1[i]+h2[i]))

return hchi

#Using above functions to compare image pairs

intmatrix = np.zeros((99, 99), dtype=*'uint8'* )

chimatrix = np.zeros((99, 99), dtype=*'float64'* )

for i1, h1 in enumerate(histlist):

for i2, h2 in enumerate(histlist):

intmatrix[i1][i2] = hintersection(h1, h2)\*255

plt.imshow(intmatrix)

plt.colorbar()

plt.text(40,0,*"Intersection"*, bbox=dict(facecolor=*'yellow'*, alpha=0.5)) #To display text on histogram

plt.show()

for i1, h1 in enumerate(histlist):

for i2, h2 in enumerate(histlist):

chimatrix[i1][i2] = hchisquared(h1, h2)

chimatrix\*=255.0/chimatrix.max()

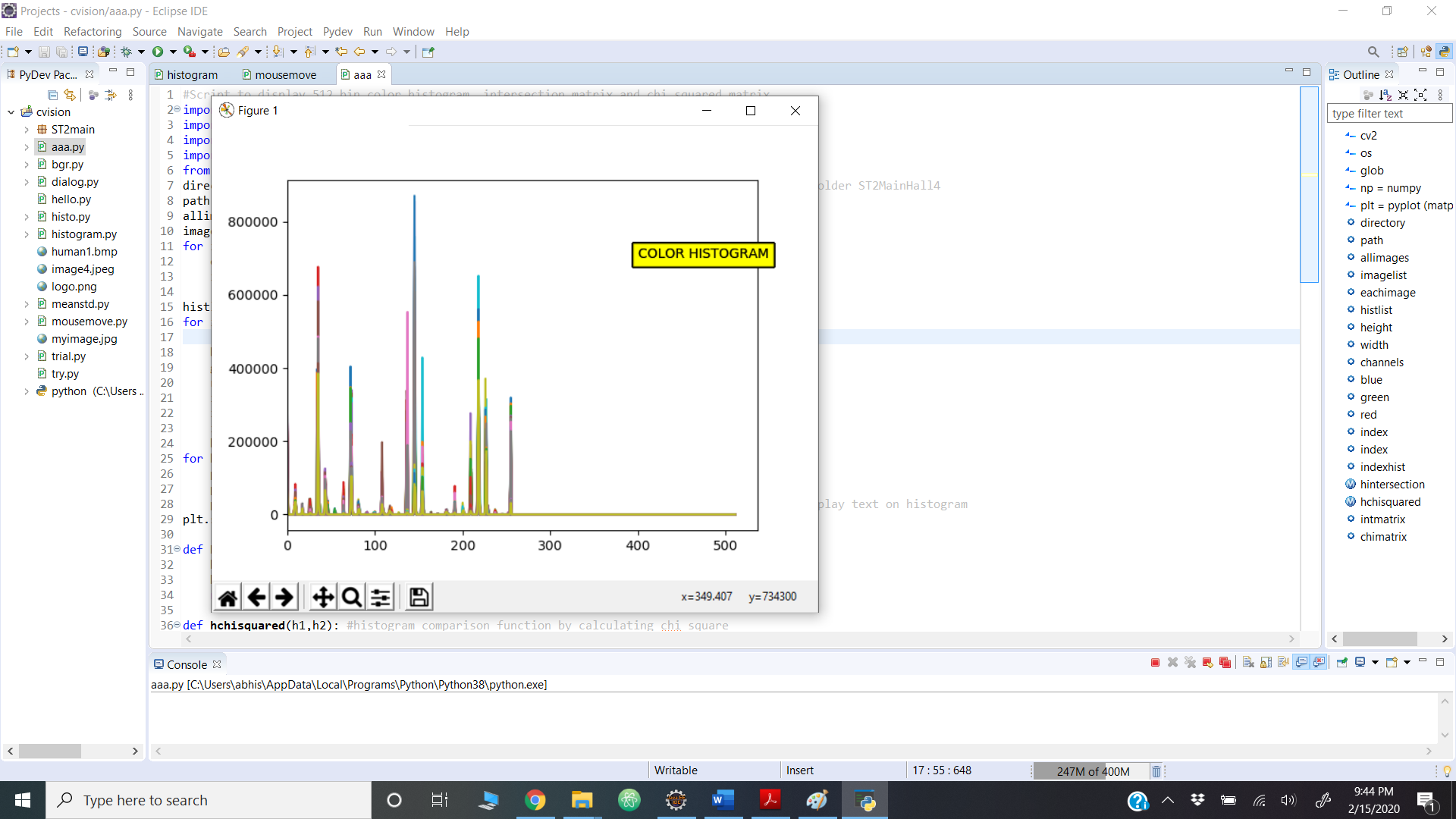
plt.imshow(chimatrix)

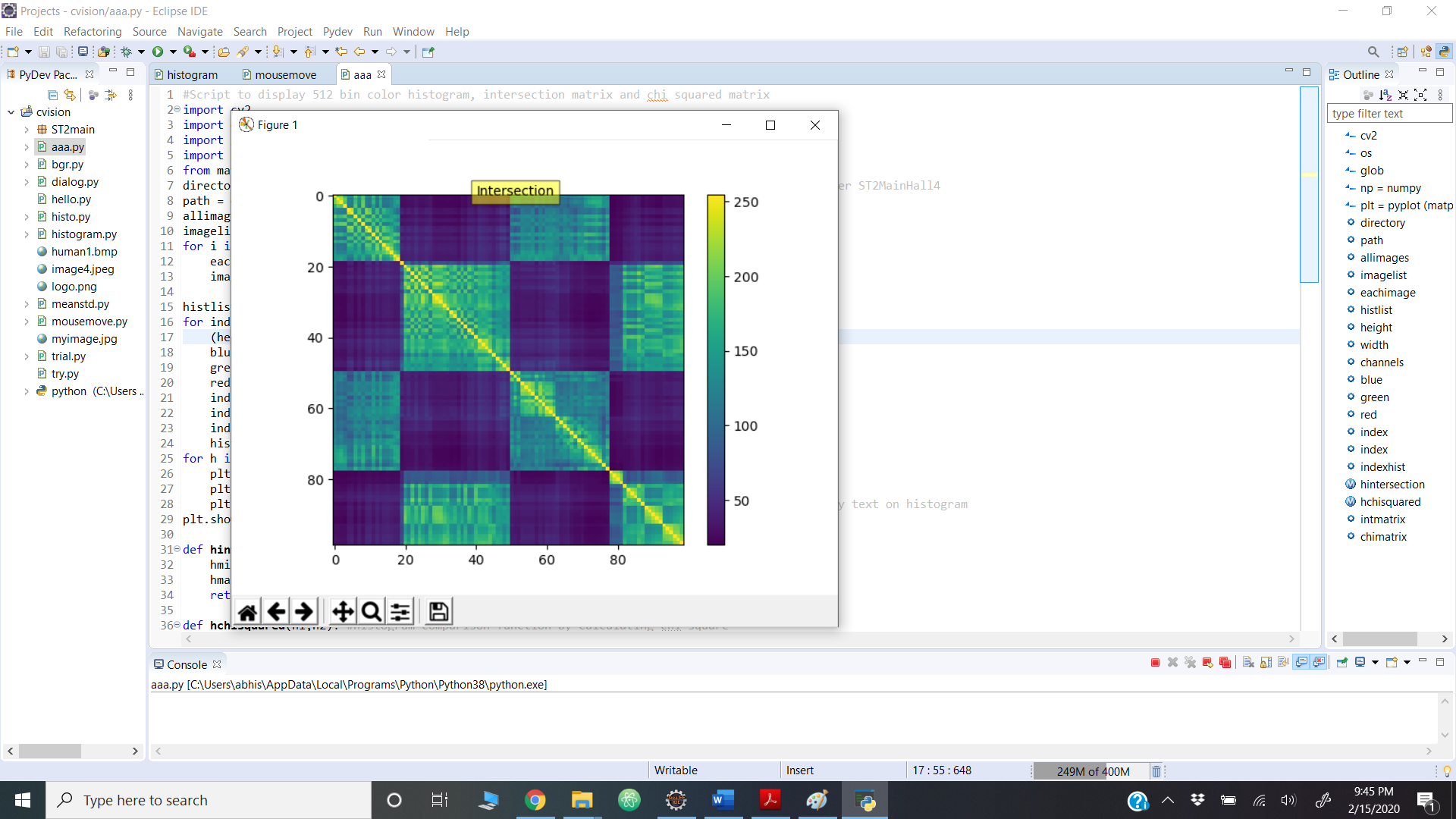
plt.colorbar()

plt.text(40,0,*"Chi squared"*, bbox=dict(facecolor=*'yellow'*, alpha=0.5)) #To display text on histogram

plt.show()

**Output images:**

**Histogram:** 

**Intersection:**

**Chi-squared:**