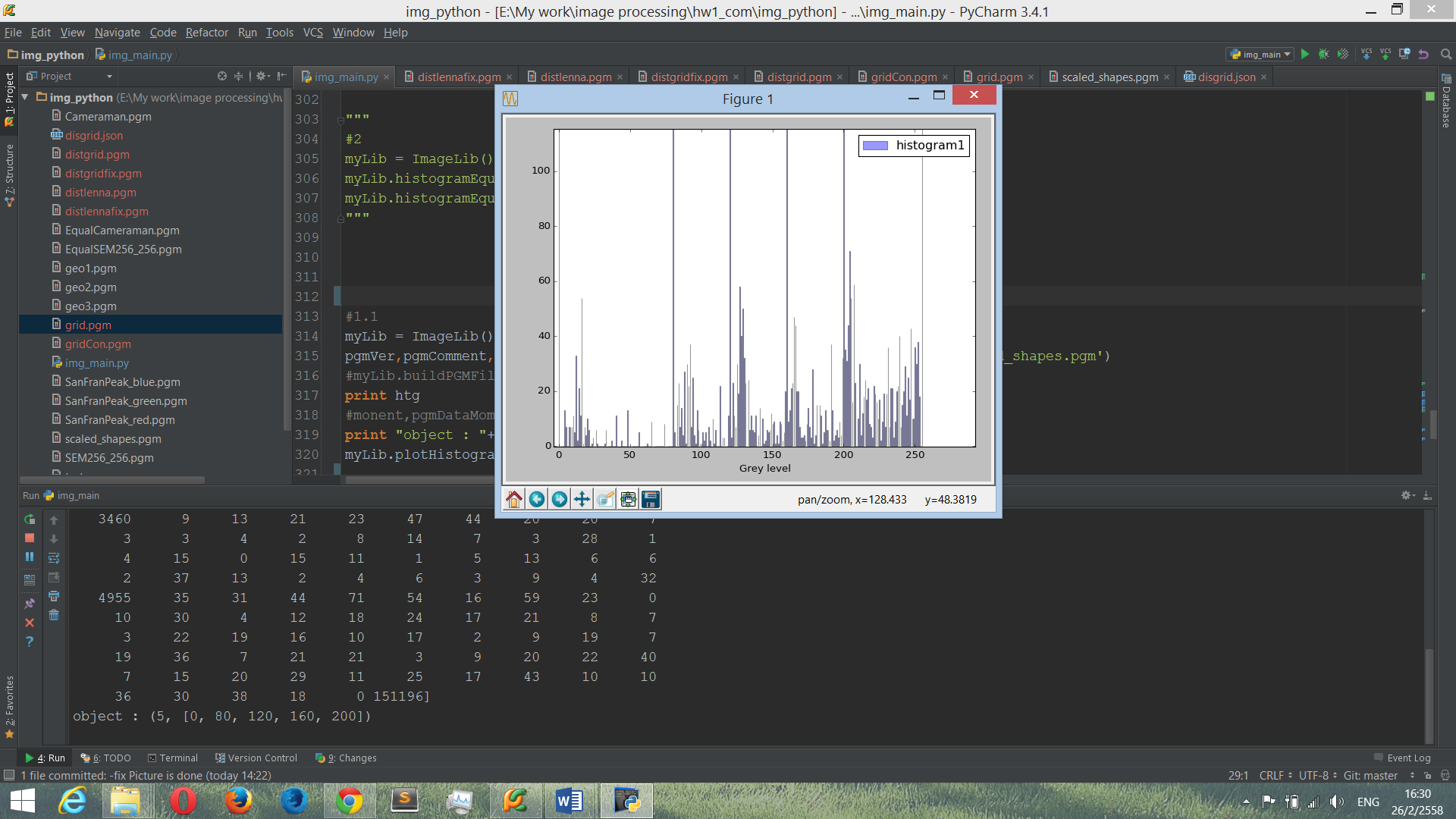
Homework Report

Digital Image Processing: Computer Assignment 1

1. Histogram and Object Moment



รูปที่ 1.1 Histogram

**Example code from my library**

myLib = ImageLib()

pgmVer,pgmComment,pgmSize,pgmGreyscale,pgmData,htg = myLib.readPGMImage('scaled\_shapes.pgm')

*# pgmData is array that contain grey scale data of each pixel in picture*

*# htg is array that contain histogram data of each grey level*

print "object : "+ str(myLib.countingObject(htg,1000))

*# return object counting array. Second parameter is threshold value.*

myLib.plotHistogramFromArray(htg) *# call plot histogram function*

รูปที่ 1.1 คือ histogram ได้จากรูป scaled\_shapes.pgm ซึ่งถ้าเรากำหนด Threshold ให้ว่าถ้าจำนวน histogram ของ Grey level ที่มีขนาดมากกว่า 1000 pixel เราจะพบว่า มีกราฟแท่งที่สูงกว่า Grey Level อื่นอยู่ 6 แท่ง นั้นเราสามารถสรุปได้ว่า Grey level ที่สูงกว่าแท่งอื่นแบบกระโดดมานั้น ประกอบด้วย Object อยู่ 5 แท่งที่มี Grey level ดังนี้ [0, 80, 120, 160, 200] และเป็น Background อยู่ 1 แท่งและมี Grey level คือ [ 255 ]

**Example code from my library**

myLib = ImageLib()

pgmVer,pgmComment,pgmSize,pgmGreyscale,pgmData,htg = myLib.readPGMImage('scaled\_shapes.pgm')

print myLib.pqMoment(0,2,pgmData,pgmSize,pgmGreyscale,0) *# return pq-moment*

print myLib.centralMoment(2,0,pgmData,pgmSize,pgmGreyscale,0) *# return central moment*

print myLib.scaleInvariantMoment(2,0,pgmData,pgmSize,pgmGreyscale,0)+myLib.scaleInvariantMoment(0,2,pgmData,pgmSize,pgmGreyscale,0),80) # return quantity

Object 1 Grey Level: 0



Result

* Center of mass : 116.130408533 , 85.512980479
* Central moment : (2,0) 1100035.49527 , (0,2) 6345057.41276
* Quantity : 0.301531111245

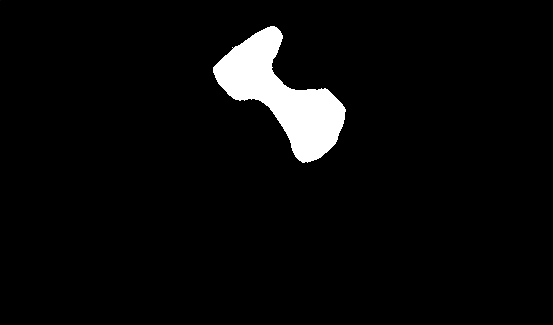
Object 2 Grey Level: 80



Result

* Center of mass : 189.080306699 , 215.044995964
* Central moment : (2,0) 2456890.03793 , (0,2) 4941000.9659
* Quantity : 0.301193318142

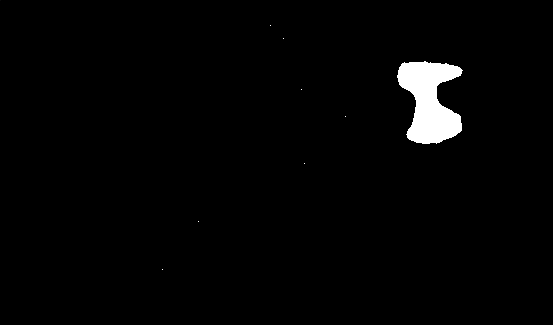
Object 3 Grey Level: 120



Result

* Center of mass : 280.547748705 , 95.1498206933
* Central moment : (2,0) 8460663.08434 , (0,2) 7875173.00226
* Quantity : 0.288181948056

Object 4 Grey Level: 160



Result

* Center of mass : 428.010404624 , 100.979768786
* Central moment : (2,0) 975991.625434 , (0,2) 2194070.58382
* Quantity : 0.26479854065

Object 5 Grey Level: 200



Result

* Center of mass : 391.438748739 , 227.531382442
* Central moment : (2,0) 4792344.16024 , (0,2) 3007131.87003
* Quantity : 0.317671394937

1. Point Operations

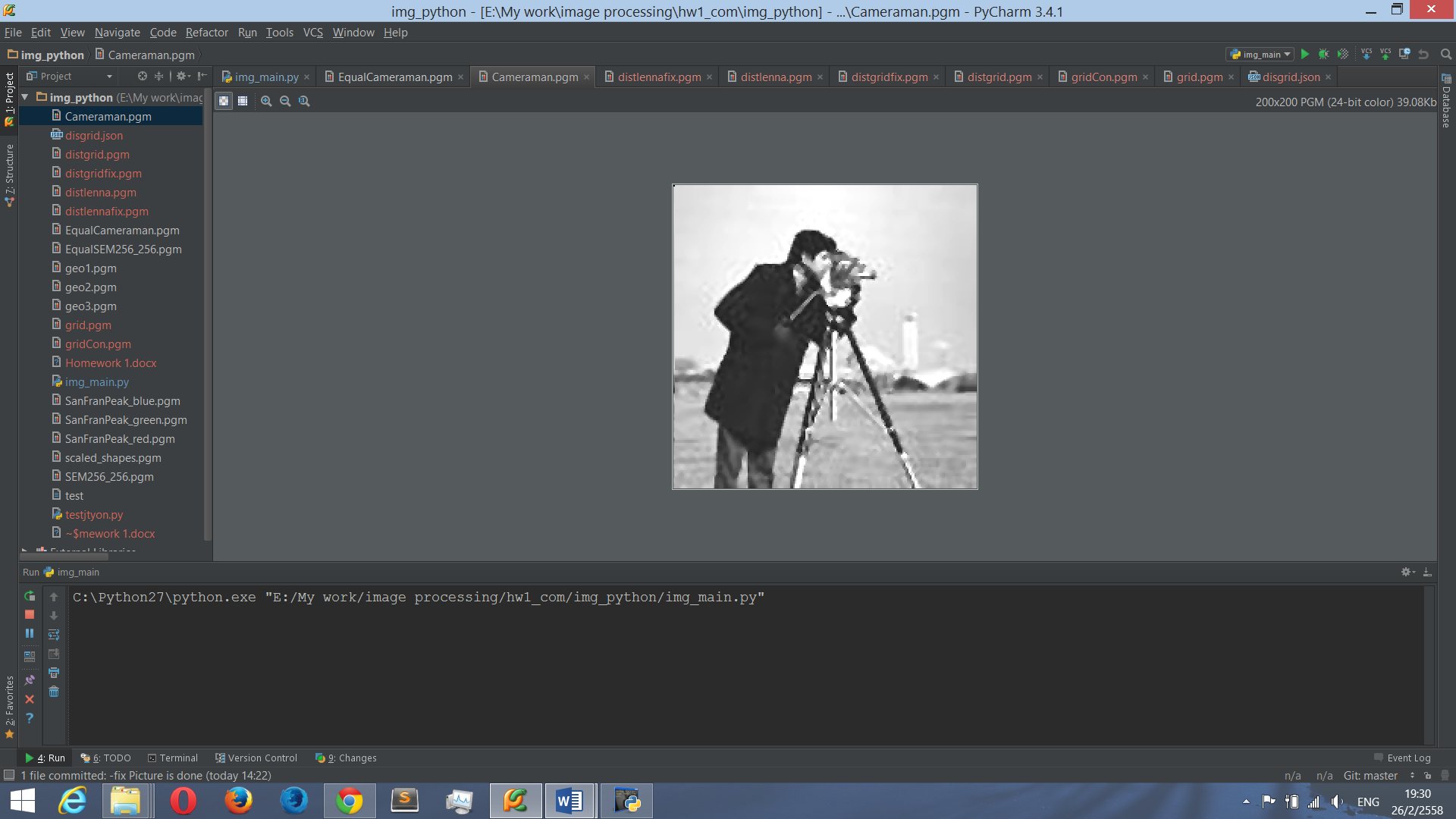
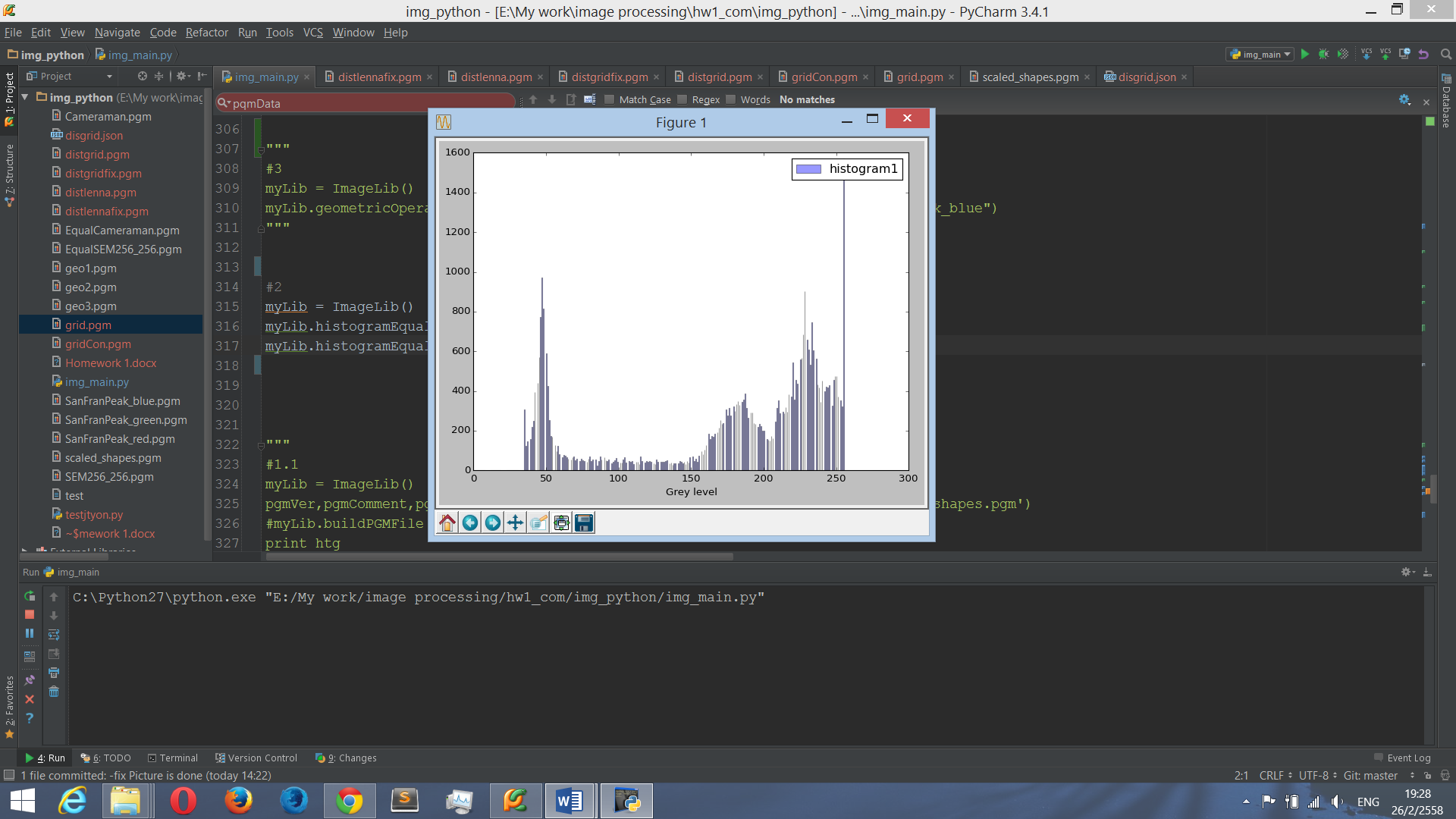
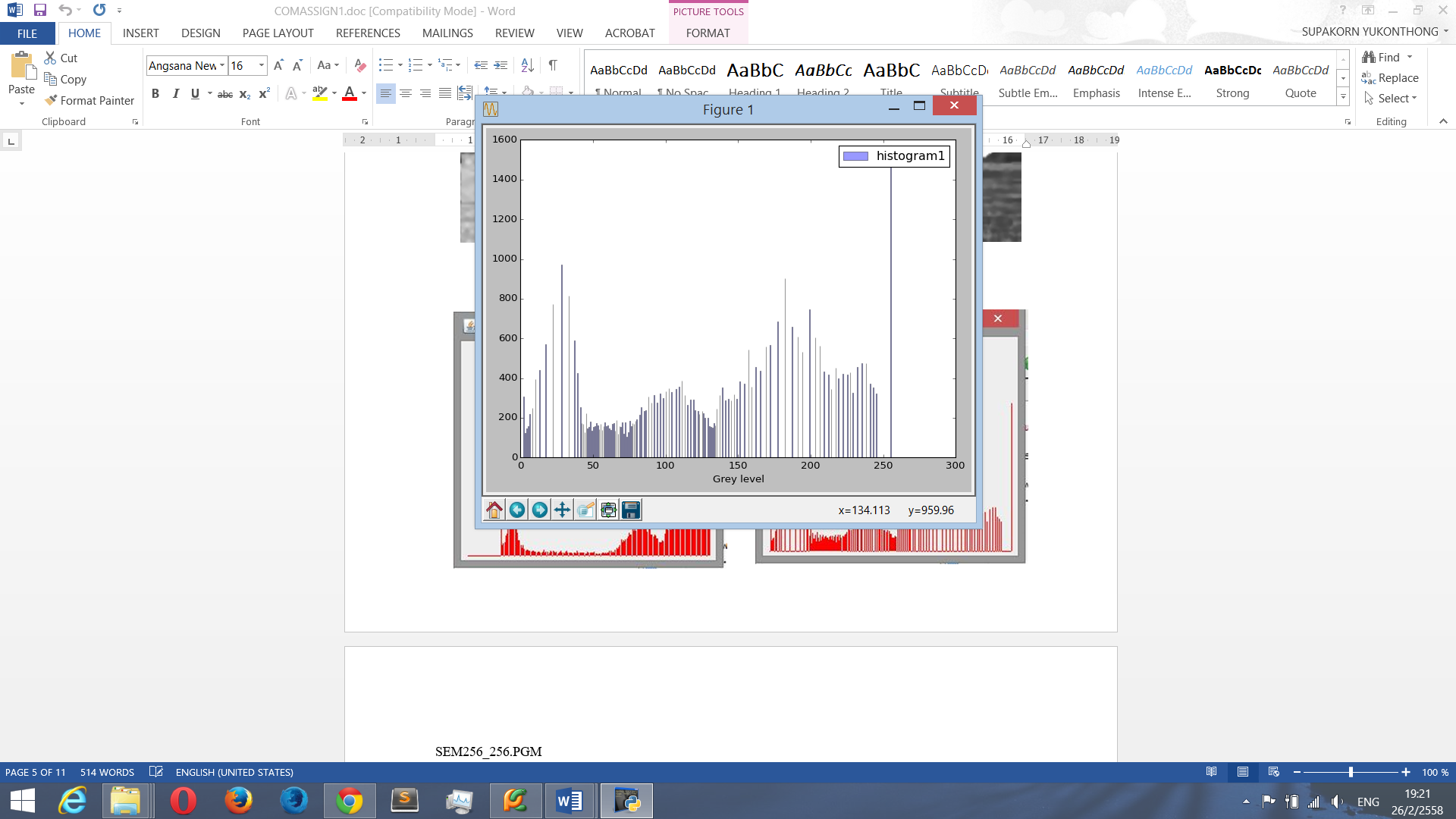
**Example code from my library**

myLib = ImageLib()

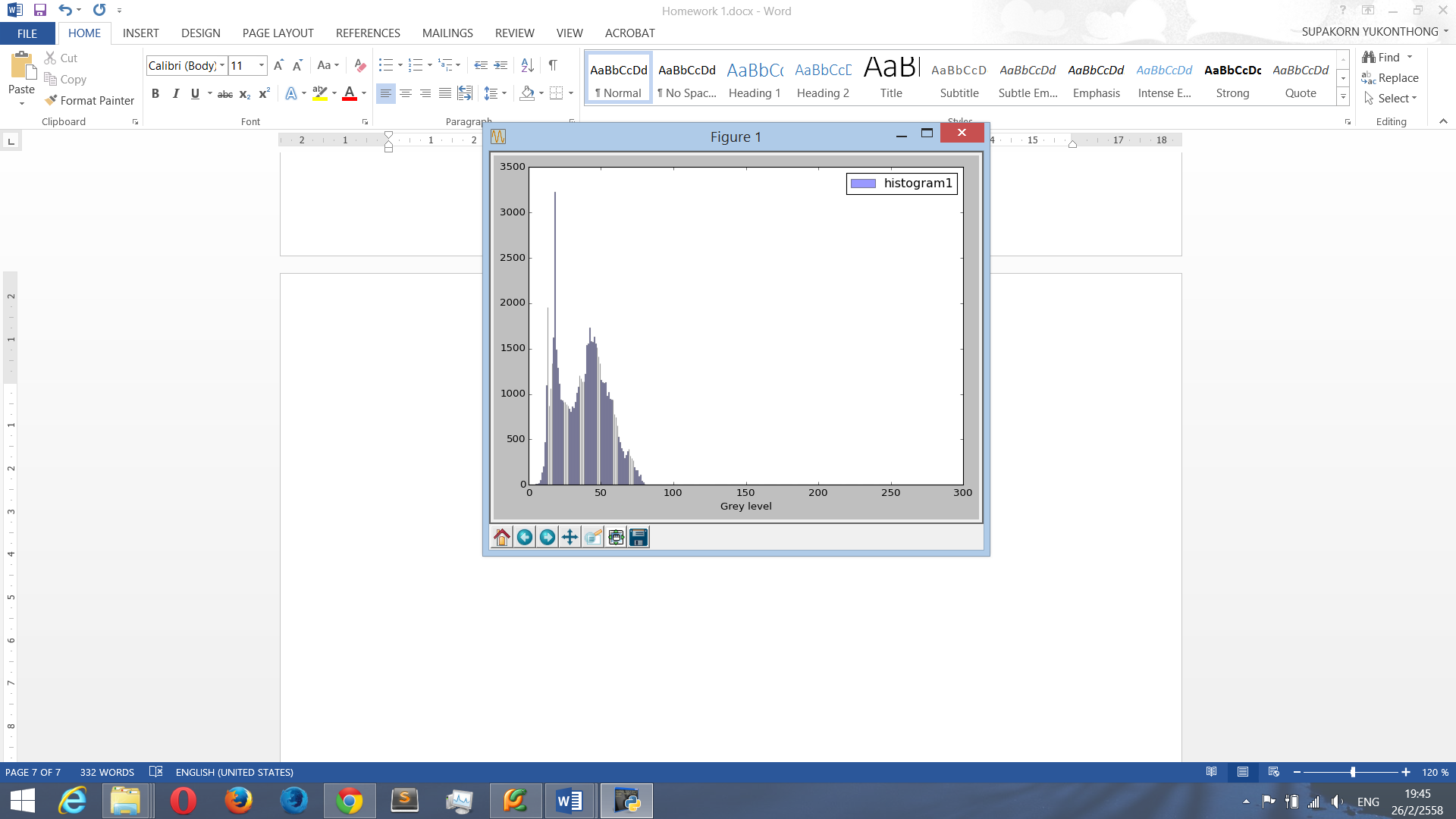
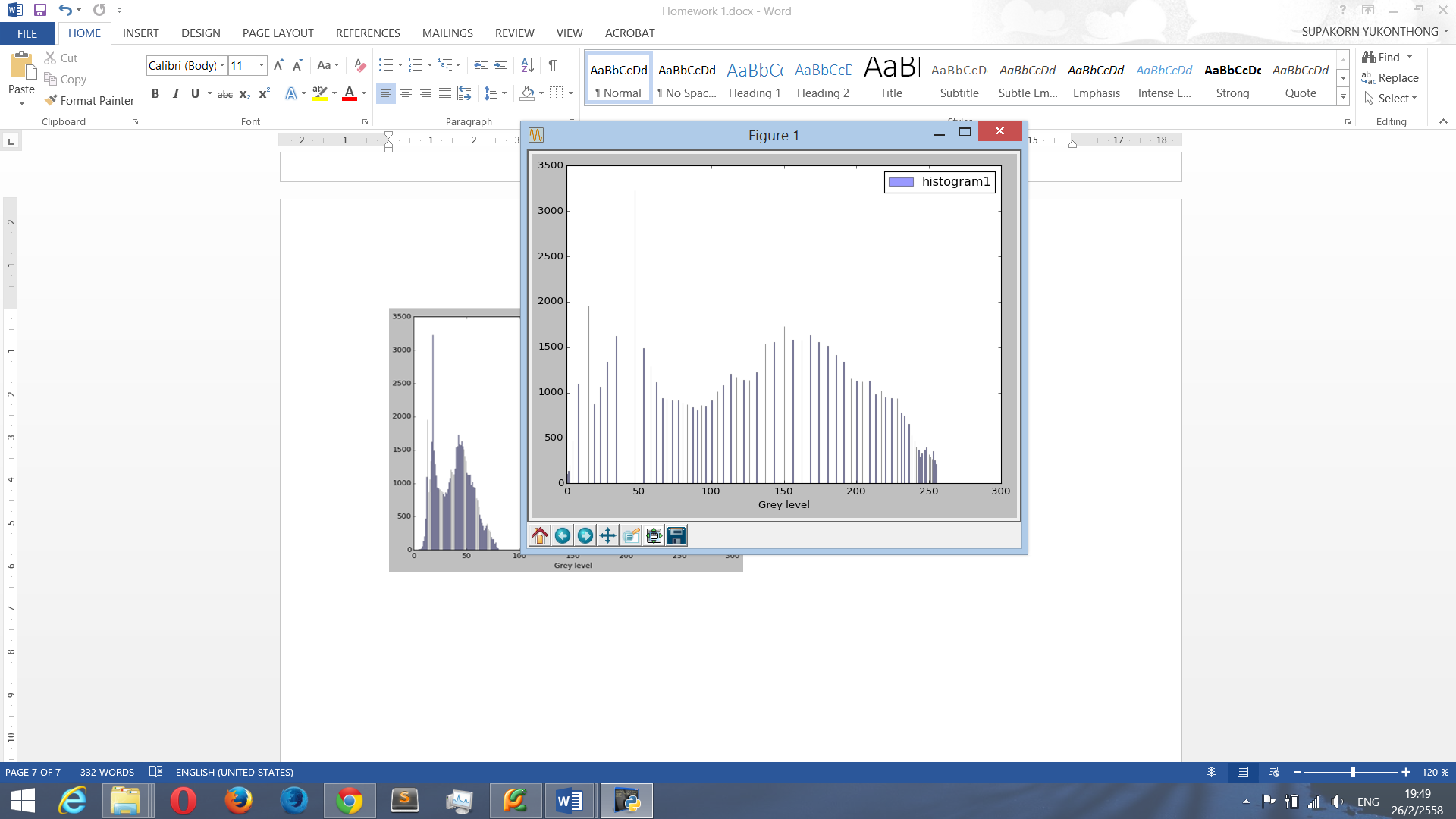
myLib.histogramEqualization("EqualCameraman","Cameraman")

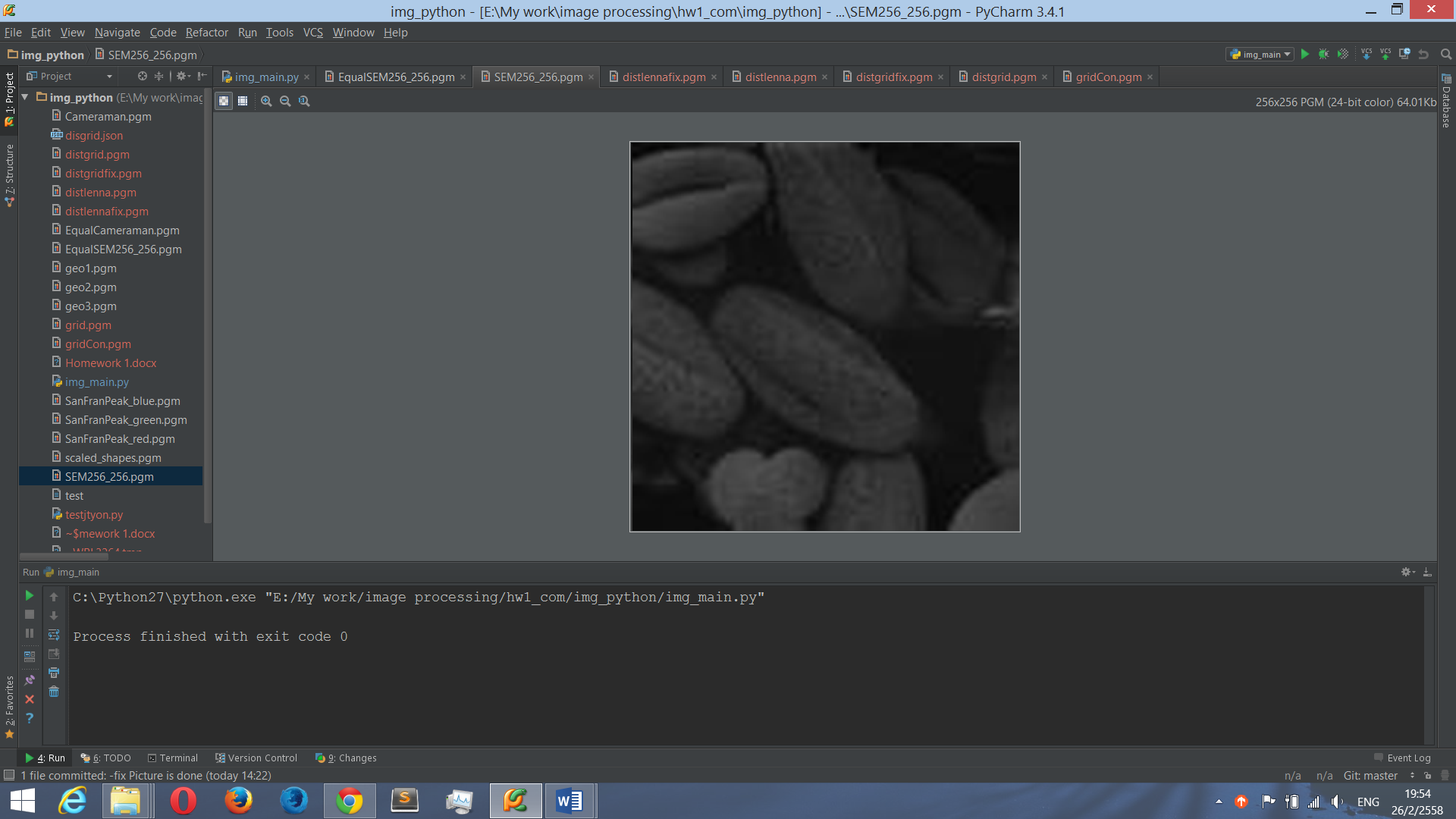
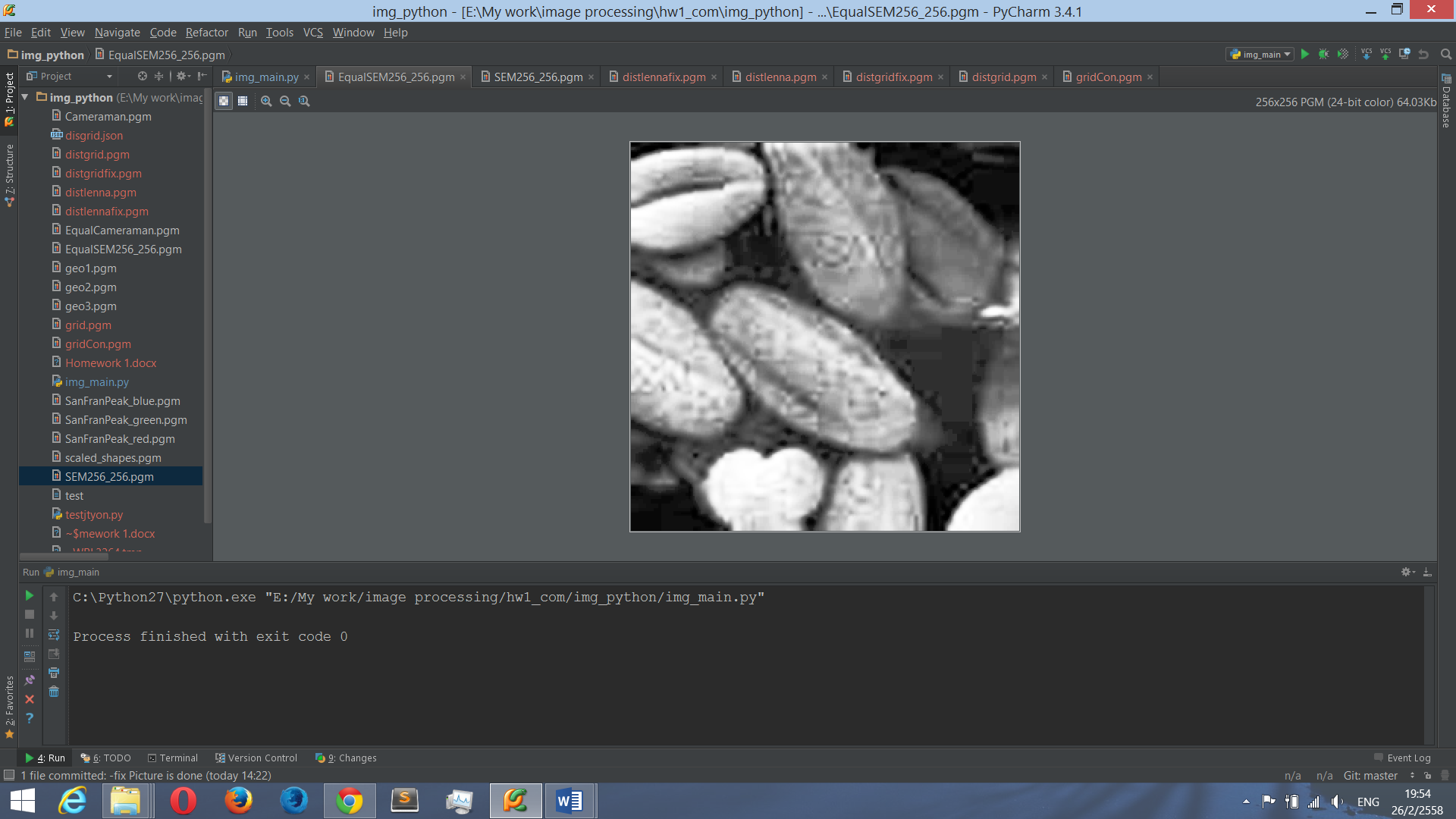
myLib.histogramEqualization("EqualSEM256\_256","SEM256\_256")

Before After









ผมเลือก Histogram Equalization ซึ่งเป็นหนึ่งใน Point Operation วิธีการนี้จะเกลี่ยค่า Histogram ให้เท่าๆกัน ทุก grey scale โดยใช้หลักการของ CDF มาช่วยในการหาความหนาแน่นของความน่าจะเป็นที่จะเกิดขึ้น หลังจากได้ค่า Histogram ใหม่มาแล้วก็ใช้วิธีปัดเศษ เพื่อให้ค่า pixel เก่าถูกปัดเข้า Grey level ใหม่ที่ได้มา

ดังนั้นจะสังเกตได้ว่าภาพ Cameraman.pgm นั้นมีความสว่างมากเกินไป หลังจากนำไปผ่าน Histogram Equalization ผลที่ได้จากการทำ จะทำให้ภาพมืดลง และเห็นลายละเอียดของภาพมากขึ้น

ส่วนภาพ SEM256\_256.pgm นั้นมืดมากจนไม่เห็นลายละเอียดชัดเจนว่าเป็นรูปอะไร หลังจากนำไปผ่าน Histogram Equalization ผลที่ได้จากการทำ จะทำให้ภาพสว่างขึ้น และเห็นลายละเอียดของวัตถุมากขึ้น

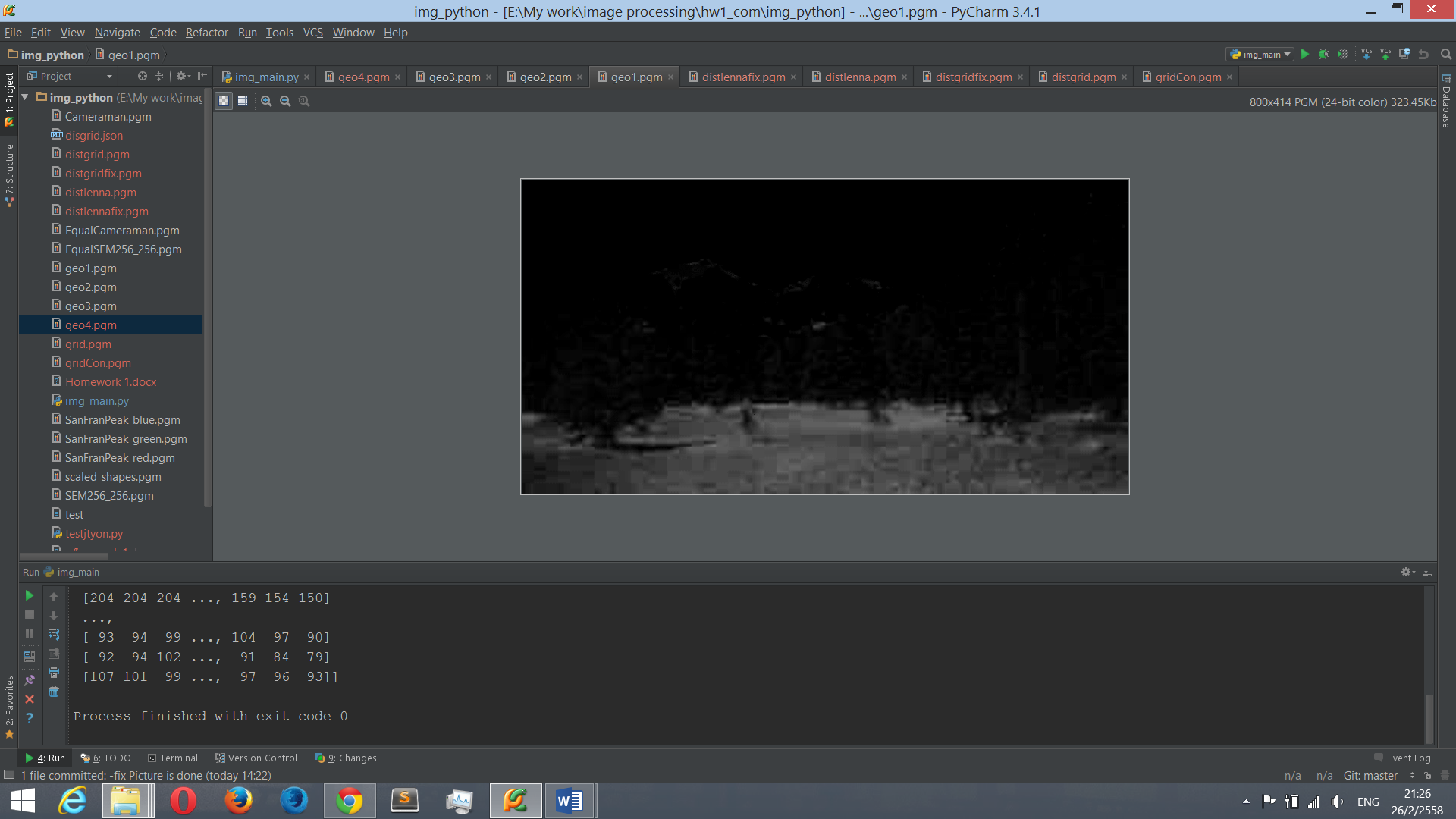
1. Algebraic Operations

เป็นการดำเนินการคณิตศาสตร์กับ Pixel ของแต่ละสีโดยตรง

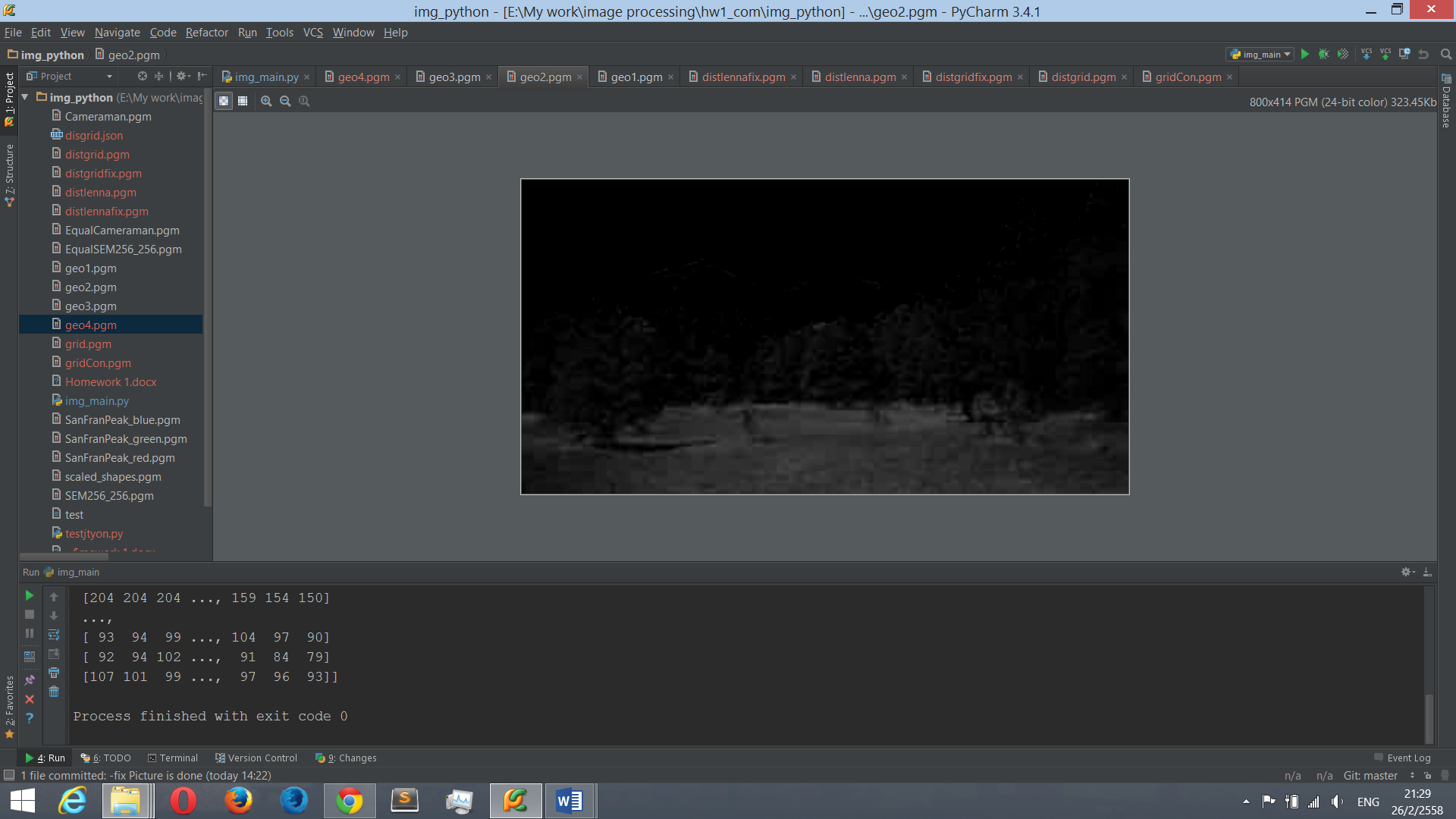
**Example code from my library**

myLib = ImageLib()

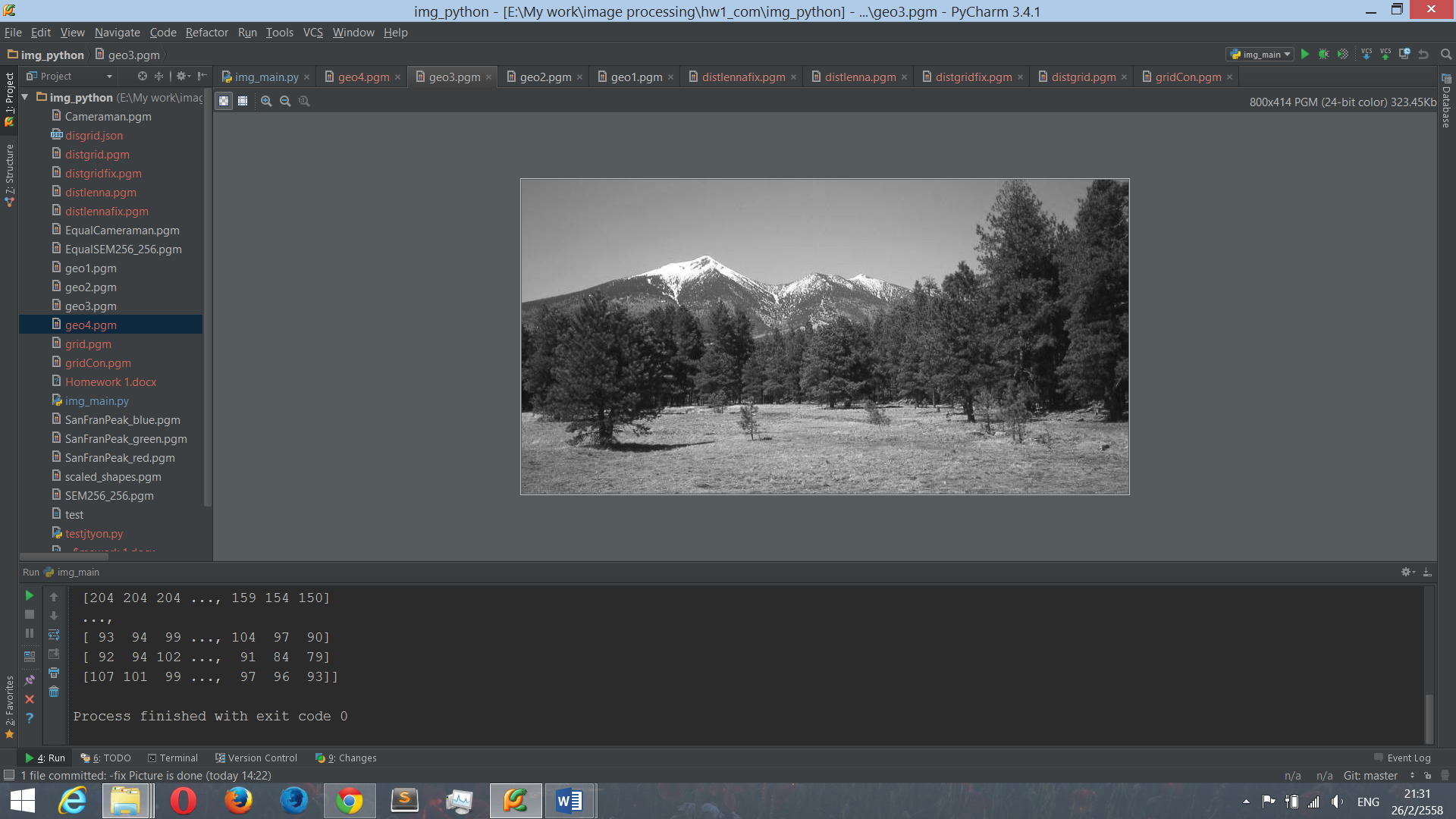
myLib.geometricOperationsImage("SanFranPeak\_red","SanFranPeak\_green","SanFranPeak\_blue")



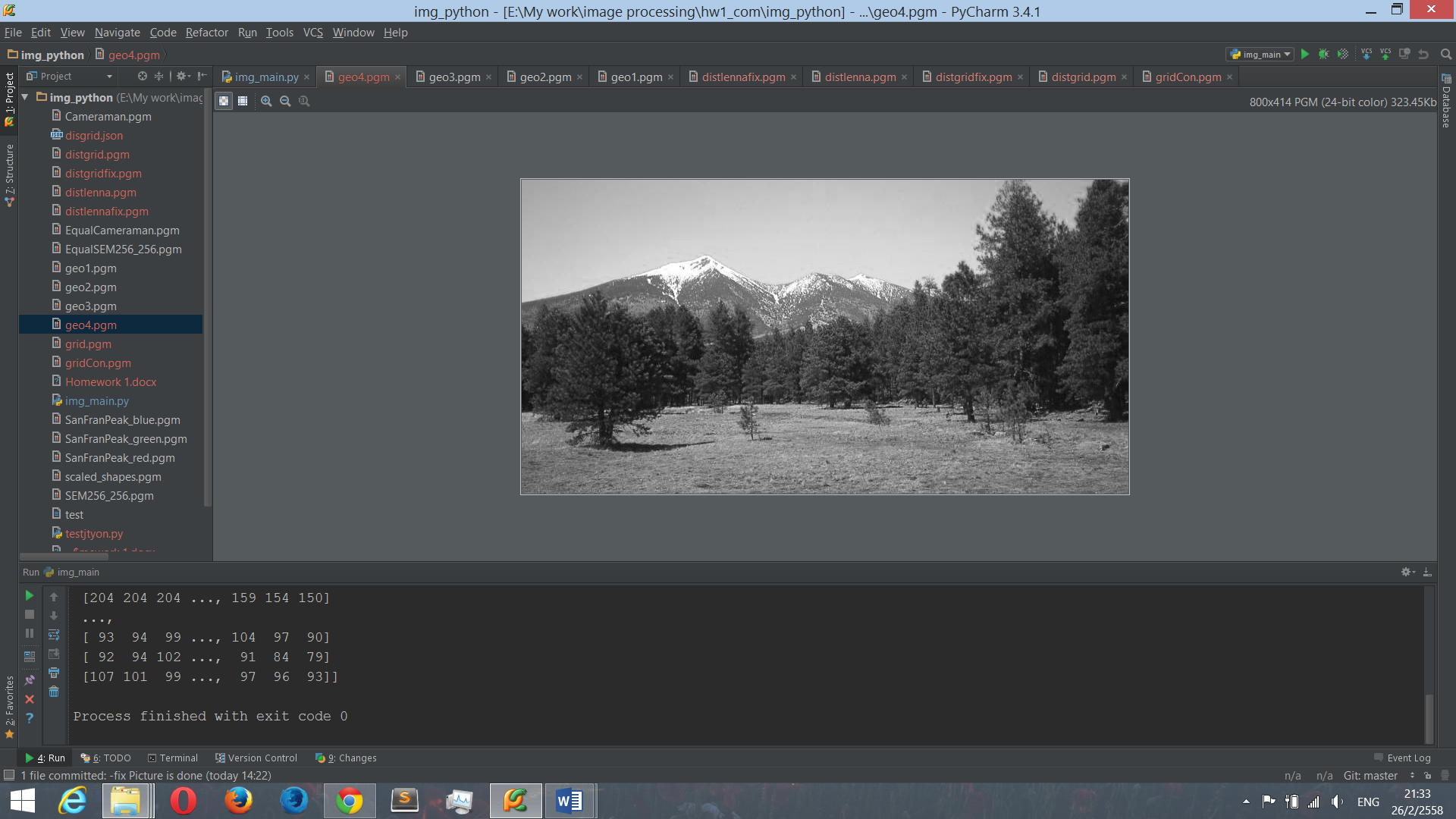
2g-r-b (excess green)



red-blue difference



gray-level (intensity)



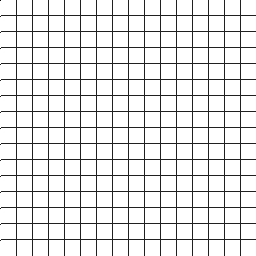
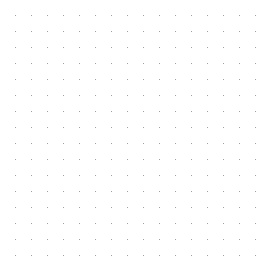
( ( (r+g)/2 ) + (2\*b) )/3 เพิ่มโทนสีฟ้า

1. Geometric Operations

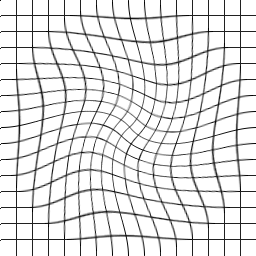
โจทย์กำหนดให้แปลงรูปเบี้ยวให้กลับมาเป็นรูปปกติ โดยเลือกใช้วิธี Control Grid Interpolation โดยผมมีขั้นตอนดำเนินการดังนี้

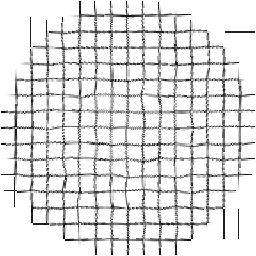
1. เนื่องจากเรามี File ที่มี ภาพ Grid ก่อนที่จะเบี้ยว ดังนั้นเราต้องหาตำแหน่งของแต่ละจุดนั้นคือ x1,y1,x2,y2,x3,y3,x4,y4 ของแต่ละ Grid ย่อย ผมเลือกใช้ Convolution กับ Kernel

เราจะได้จุดพิกัดที่ตัดกันมีค่า Grey Level น้อยสุด จึงทำให้เราเห็นจุดสีเทาๆ และจุดที่ไม่ใช้จุดตัดจะมีค่า Grey Level สูงทำให้เราเห็นเป็นสีขาว หลังจากนั้นนำมาเติมจุดทางขวาและด้านล่างสุดให้ครบจำนวน Grid



1. หลังจากนั้นก็นำค่าแต่ละตำแหน่งของแต่ล่ะ Grid ไปหาค่า w1,w2,w3,w4 เพื่อนำไปแทนสมการ
2. หลังจากนั้นก็สร้าง Array ชุดใหม่ขึ้นมาจำลองว่าเป็นรูปดี นำตำแหน่งในรูปดีที่เราต้องการไปแทนสมการ เราจะได้ตำแหน่ง Grey Level ของรูปเสียเพื่อนำไปแทนในรูปดี





**Example code from my library**

myLib = ImageLib()

kernel = np.array([[0,1,0],[1,1,1],[0,1,0]])

pgmDataCon,pgmCon = myLib.convolutionWithKernel("grid",kernel)

normalGridPosition = myLib.findPixelPosition(pgmDataCon,256,256)

unNormalGridPosition = myLib.readJsonPixelPosition("disgrid.json")

xWeight,yWeight = myLib.findWeight(normalGridPosition,unNormalGridPosition)

myLib.fixBadPicture(xWeight,yWeight,"distgrid",normalGridPosition)

โค้ดทั้งหมดของผมครับ

import numpy as np

import matplotlib.pyplot as plt

import math

import json

class ImageLib:

def readPGMImage(self,path): #Use for PGM File reading

file = open(path, "rb")

pgmVer = file.readline().split()

pgmComment = []

while True:

pgmComment\_eachline = file.readline()

if(pgmComment\_eachline[0]=="#"):

pgmComment.append(pgmComment\_eachline)

else:

break

pgmSize = pgmComment\_eachline.split()

pgmGreyscale = file.readline().split()

pgmDataList = []

htg = np.zeros((256),dtype=np.int32)

np.set\_printoptions(suppress=True)

for j in range(int(pgmSize[1])):

pgmDataX = []

for i in range(int(pgmSize[0])):

byte = file.read(1)

chrToInt = ord(byte)

pgmDataX.append(chrToInt)

htg[chrToInt] = htg[chrToInt]+1

pgmDataList.append(pgmDataX)

file.close()

pgmData = np.asarray(pgmDataList,dtype=np.int32)

return pgmVer,pgmComment,pgmSize,pgmGreyscale,pgmData,htg

#pgmData is data pixel that i get from pgm file under grey level value(numpy array).

#pgmSize contain width and height of pixel(list).

#htg is a histogram of image (numpy array).

def plotHistogramFromArray(self,histogram\_arr): #Use for histogram ploting

index = np.arange(256)

bar\_width = 0.35

opacity = 0.4

rects1 = plt.bar(index, histogram\_arr, bar\_width,

alpha=opacity,

color='b',

label='histogram1')

plt.xlabel('Grey level')

plt.legend()

plt.tight\_layout()

plt.show()

def countingObject(self,histogram,threshold\_object): #Use for counting object

countObject = 0

countObjectGreyLevel = []

for i in range(histogram.size):

if histogram[i] >= threshold\_object:

countObject += 1

countObjectGreyLevel.append(i)

countObjectGreyLevel.remove(max(countObjectGreyLevel))

return countObject-1,countObjectGreyLevel # minus 1 for backgroud

#def buildPGMInterestObject(self,inputFileName):

def pqMoment(self,p,q,pgmData,pgmSize,greyLevel,greyLevelSelected):

# Use for pq moment finding

moment = 0

pgmDataMoment = np.zeros((int(pgmSize[1]),int(pgmSize[0])), dtype=np.int32)

for i in range(int(pgmSize[1])):

for j in range(int(pgmSize[0])):

if pgmData[i][j] == greyLevelSelected:

pgmDataMoment[i][j] = 1

else:

pgmDataMoment[i][j] = 0

moment += ((math.pow(j,p))\*((math.pow(i,q))\*pgmDataMoment[i][j]))

#ImageLib.buildPGMFile(self,"testmoment",pgmSize[0],pgmSize[1],greyLevel,pgmDataMoment)

return moment,pgmDataMoment

def centralMoment(self,p,q,pgmData,pgmSize,greyLevel,greyLevelSelected):

centralMoment = 0

moment1,pgmDataMoment = ImageLib.pqMoment(self,1,0,pgmData,pgmSize,greyLevel,greyLevelSelected)

moment2,pgmDataMoment = ImageLib.pqMoment(self,0,1,pgmData,pgmSize,greyLevel,greyLevelSelected)

moment3,pgmDataMoment = ImageLib.pqMoment(self,0,0,pgmData,pgmSize,greyLevel,greyLevelSelected)

xCoor = moment1/moment3

yCoor = moment2/moment3

print "Central of Mass x : " + str(xCoor)

print "Central of Mass y : " + str(yCoor)

for i in range(int(pgmSize[1])):

for j in range(int(pgmSize[0])):

centralMoment += ((math.pow((j-xCoor),p))\*((math.pow(i-yCoor,q))\*pgmDataMoment[i][j]))

return centralMoment

def scaleInvariantMoment(self,p,q,pgmData,pgmSize,greyLevel,greyLevelSelected):

scaleInvariantMoment = 0

centralMomentPQ = ImageLib.centralMoment(self,p,q,pgmData,pgmSize,greyLevel,greyLevelSelected)

centralMoment00 = ImageLib.centralMoment(self,0,0,pgmData,pgmSize,greyLevel,greyLevelSelected)

scaleInvariantMoment = centralMomentPQ/(math.pow(centralMoment00,(1+((p+q)/2))))

return scaleInvariantMoment

def buildPGMFile(self,fileName,width,height,greyLevel,pgmData): #Write PGM File

f = open(str(fileName)+".pgm","wb")

f.write("P5\n");

f.write("# "+str(fileName)+"\n");

f.write(str(width)+" "+str(height)+"\n"+str(greyLevel[0])+"\n");

for i in range(int(height)):

for j in range(int(width)):

if pgmData[i][j]<0:

pgmData[i][j] = 0

elif pgmData[i][j]>int(greyLevel[0]):

pgmData[i][j] = int(greyLevel[0])

f.write(chr(pgmData[i][j]));

f.close()

def histogramEqualization(self,outputFileName,inputFileName):

pgmVer,pgmComment,pgmSize,pgmGreyscale,pgmData,htg = ImageLib.readPGMImage(self,str(inputFileName)+".pgm")

ImageLib.plotHistogramFromArray(self,htg)

imgArea = int(pgmSize[0])\*int(pgmSize[1])

htgScaleAfter = np.zeros(int(pgmGreyscale[0])+1,dtype=np.int32)

propOfA = 0.0

for i in range(htg.size):

propOfA += float(htg[i])/float(imgArea)

#print "propA" + str(propOfA)

fDa = propOfA \* float(pgmGreyscale[0])

htgScaleAfter[i] = round(fDa)

pgmDataAfter = np.zeros((int(pgmSize[1]),int(pgmSize[0])),dtype=np.int32)

for i in range(int(pgmSize[1])):

for j in range(int(pgmSize[0])):

pgmDataAfter[i][j] = htgScaleAfter[pgmData[i][j]]

ImageLib.buildPGMFile(self,outputFileName,pgmSize[0],pgmSize[1],pgmGreyscale,pgmDataAfter)

pgmVer,pgmComment,pgmSize,pgmGreyscale,pgmData,htg = ImageLib.readPGMImage(self,str(outputFileName)+".pgm")

ImageLib.plotHistogramFromArray(self,htg)

def geometricOperationsImage(self,redPgmFileName,greenPgmFileName,bluePgmFileName):

redpgmVer,redpgmComment,redpgmSize,redpgmGreyscale,redpgmData,redhtg = ImageLib.readPGMImage(self,str(redPgmFileName)+".pgm")

greenpgmVer,greenpgmComment,greenpgmSize,greenpgmGreyscale,greenpgmData,greenhtg = ImageLib.readPGMImage(self,str(greenPgmFileName)+".pgm")

bluepgmVer,bluepgmComment,bluepgmSize,bluepgmGreyscale,bluepgmData,bluehtg = ImageLib.readPGMImage(self,str(bluePgmFileName)+".pgm")

print redpgmData

print greenpgmData

print bluepgmData

geo1 = ((2\*redpgmData)-greenpgmData)-bluepgmData

ImageLib.buildPGMFile(self,"geo1",redpgmSize[0],redpgmSize[1],redpgmGreyscale,geo1)

geo2 = (redpgmData-bluepgmData)

ImageLib.buildPGMFile(self,"geo2",redpgmSize[0],redpgmSize[1],redpgmGreyscale,geo2)

geo3 = (redpgmData+greenpgmData+bluepgmData)/3

ImageLib.buildPGMFile(self,"geo3",redpgmSize[0],redpgmSize[1],redpgmGreyscale,geo3)

geo4 = (((redpgmData+greenpgmData)/2)+2\*bluepgmData)/3 #my own option

ImageLib.buildPGMFile(self,"geo4",redpgmSize[0],redpgmSize[1],redpgmGreyscale,geo4)

def convolutionWithKernel(self,inputFileName,kernel):

pgmVer,pgmComment,pgmSize,pgmGreyscale,pgmData,htg = ImageLib.readPGMImage(self,str(inputFileName)+".pgm")

pgmDataCon = np.zeros((int(pgmSize[1]),int(pgmSize[0])),dtype=np.int32)

pgmDataCon.fill(255)

#print pgmData

for i in range(1,int(pgmSize[1])-1):

for j in range(1,int(pgmSize[0])-1):

temp = 0

#XYY

#YYY

#YYY

temp += pgmData[i][j]\*kernel[1][1]

temp += pgmData[i-1][j-1]\*kernel[0][0]

temp += pgmData[i-1][j]\*kernel[0][1]

temp += pgmData[i+1][j+1]\*kernel[2][2]

temp += pgmData[i][j-1]\*kernel[1][0]

temp += pgmData[i][j+1]\*kernel[1][2]

temp += pgmData[i+1][j-1]\*kernel[2][0]

temp += pgmData[i+1][j]\*kernel[2][1]

temp += pgmData[i+1][j+1]\*kernel[2][2]

pgmDataCon[i][j] = temp

pgmCon = np.array(pgmDataCon)

#extend grid

for i in range(15,int(pgmSize[1]),16):

pgmDataCon[255][i] = 160

pgmDataCon[i][255] = 160

ImageLib.buildPGMFile(self,str(inputFileName)+"Con",pgmSize[0],pgmSize[1],pgmGreyscale,pgmDataCon)

return pgmDataCon,pgmCon

def readJsonPixelPosition(self,fileName):

json\_data=open(fileName)

data = json.load(json\_data)

json\_data.close()

#print data[0]["y"+str(4)]

return data

def findPixelPosition(self,convoluteArr,width,height):

normalGridPosition = []

count = 0

for i in range(height):

for j in range(width):

if(convoluteArr[i][j]==160):

dict = {'u': count,'x1': j-15,'y1': i-15,'x2':j,'y2': i-15,'x3': j-15,'y3': i,'x4': j,'y4': i}

normalGridPosition.append(dict)

count +=1

return normalGridPosition

def findWeight(self,goodGrid,badGrid):

xWeight = []

yWeight = []

for i in range(256):

a = np.array([[ goodGrid[i]['x1'],goodGrid[i]['y1'],goodGrid[i]['x1']\*goodGrid[i]['y1'],1 ],[ goodGrid[i]['x2'],goodGrid[i]['y2'],goodGrid[i]['x2']\*goodGrid[i]['y2'],1 ],[ goodGrid[i]['x3'],goodGrid[i]['y3'],goodGrid[i]['x3']\*goodGrid[i]['y3'],1 ],[ goodGrid[i]['x4'],goodGrid[i]['y4'],goodGrid[i]['x4']\*goodGrid[i]['y4'],1 ]])

b = np.array([badGrid[i]['x1'],badGrid[i]['x2'],badGrid[i]['x3'],badGrid[i]['x4']])

x = np.linalg.solve(a, b)

xWeight.append(x)

a = np.array([[ goodGrid[i]['x1'],goodGrid[i]['y1'],goodGrid[i]['x1']\*goodGrid[i]['y1'],1 ],[ goodGrid[i]['x2'],goodGrid[i]['y2'],goodGrid[i]['x2']\*goodGrid[i]['y2'],1 ],[ goodGrid[i]['x3'],goodGrid[i]['y3'],goodGrid[i]['x3']\*goodGrid[i]['y3'],1 ],[ goodGrid[i]['x4'],goodGrid[i]['y4'],goodGrid[i]['x4']\*goodGrid[i]['y4'],1 ]])

b = np.array([badGrid[i]['y1'],badGrid[i]['y2'],badGrid[i]['y3'],badGrid[i]['y4']])

y = np.linalg.solve(a, b)

yWeight.append(y)

return xWeight,yWeight

def fixBadPicture(self,xWeight,yWeight,badPictureFileName,goodGrid):

pgmVer,pgmComment,pgmSize,pgmGreyscale,pgmData,htg = ImageLib.readPGMImage(self,str(badPictureFileName)+".pgm")

pgmFixedPicture = np.zeros((int(pgmSize[1]),int(pgmSize[0])), dtype=np.int32)

for k in goodGrid:

#print k

for i in range(k['y1'],k['y4']+1):

for j in range(k['x1'],k['x4']+1):

xAxis = round((xWeight[k['u']][0]\*j) + (xWeight[k['u']][1]\*i) + (xWeight[k['u']][2]\*i\*j) + (xWeight[k['u']][3]),1)

yAxis = round((yWeight[k['u']][0]\*j) + (yWeight[k['u']][1]\*i) + (yWeight[k['u']][2]\*i\*j) + (yWeight[k['u']][3]),1)

"""

if xAxis >=255 or yAxis >=255:

xAxis = 255

yAxis = 255

print "over"

elif xAxis <=0 or yAxis <=0:

xAxis = 0

yAxis = 0

print "less"

"""

pgmFixedPicture[i][j] = pgmData[yAxis][xAxis]

"""

print "i " + str(i)

print "j " + str(j)

print "x " + str(xAxis)

print "y " + str(yAxis)

print "xW " + str(xWeight[k['u']])

print "yW " + str(yWeight[k['u']])

print "u " + str(k['u'])

"""

ImageLib.buildPGMFile(self,str(badPictureFileName)+"fix",pgmSize[0],pgmSize[1],pgmGreyscale,pgmFixedPicture)

# under this line is for solving each problem

"""

#4

myLib = ImageLib()

#myLib.readJsonPixelPosition("disgrid.json")

kernel = np.array([[0,1,0],[1,1,1],[0,1,0]])

print kernel

#np.set\_printoptions(threshold=np.nan)

pgmDataCon,pgmCon = myLib.convolutionWithKernel("grid",kernel)

#print np.amax(pgmCon) #find max value

#print np.unique(pgmDataCon) #find number

#np.set\_printoptions(threshold=np.nan)

#print pgmDataCon

#print pgmCon

normalGridPosition = myLib.findPixelPosition(pgmDataCon,256,256)

unNormalGridPosition = myLib.readJsonPixelPosition("disgrid.json")

#print normalGridPosition

#print unNormalGridPosition[255]['x2']

#print normalGridPosition[255]['x2']

xWeight,yWeight = myLib.findWeight(normalGridPosition,unNormalGridPosition)

#print xWeight

#print yWeight

myLib.fixBadPicture(xWeight,yWeight,"distgrid",normalGridPosition)

"""

"""

#3

myLib = ImageLib()

myLib.geometricOperationsImage("SanFranPeak\_red","SanFranPeak\_green","SanFranPeak\_blue")

"""

"""

#2

myLib = ImageLib()

myLib.histogramEqualization("EqualCameraman","Cameraman")

myLib.histogramEqualization("EqualSEM256\_256","SEM256\_256")

"""

"""

#1.1

myLib = ImageLib()

pgmVer,pgmComment,pgmSize,pgmGreyscale,pgmData,htg = myLib.readPGMImage('scaled\_shapes.pgm')

#myLib.buildPGMFile("test",pgmSize[0],pgmSize[1],pgmGreyscale,pgmData)

print htg

#monent,pgmDataMoment = myLib.pqMoment(1,1,pgmData,pgmSize,pgmGreyscale,255)

print "object : "+ str(myLib.countingObject(htg,1000))

myLib.plotHistogramFromArray(htg)

"""

"""

#1.2

myLib = ImageLib()

pgmVer,pgmComment,pgmSize,pgmGreyscale,pgmData,htg = myLib.readPGMImage('scaled\_shapes.pgm')

print myLib.pqMoment(0,2,pgmData,pgmSize,pgmGreyscale,200) #return pq-moment

print myLib.centralMoment(0,2,pgmData,pgmSize,pgmGreyscale,200) #return central moment

print myLib.scaleInvariantMoment(2,0,pgmData,pgmSize,pgmGreyscale,200)+myLib.scaleInvariantMoment(0,2,pgmData,pgmSize,pgmGreyscale,200)

"""