Digital Image Processing (261453)

Computer Assignment 3

จงหาวิธีในการคำนวณหา ว่าส่วนรูปดอกไม้สีขาวของรูป Flower_Snack.pgm มีพื้นที่เท่าไร ถ้าพื้นที่ทั้งหมดคือ 2.60 cm^2 และทำเช่นเดียวกันกับรูป Crab.pgm ถ้าในกรณีนี้พื้นที่ทั้งหมดคือ 2.766 cm^2 ในการทำงานชิ้นนี้นักศึกษาสามารถใช้วิธีใดก็ได้ที่เรียนในชั้นเรียน หรือวิธีอื่นๆที่อ่านจากหนังสือได้ **หมายเหตุ** สำหรับการบ้านนี้ นศ. ห้ามใช้ Tool Box หรือ Library สำเร็จรูปใดๆทั้งสิ้น ยกเว้น FFT และ wavelet transform หรือถ้าไม่แน่ใจให้นักศึกษาถามอาจารย์

Solution

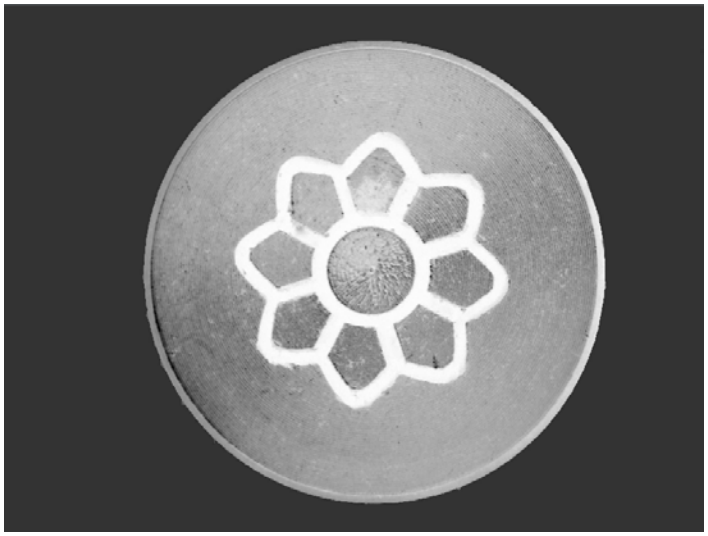
ภาพรวมของวิธีที่ใช้ในการหาค่า

1) นำภาพตั้งต้นไปทำการเบลอโดยใช้ Gaussian low pass filter เพื่อให้ภาพผิวของวัตถุต่อกัน รวมถึงจะได้ทำให้เราเห็นองค์ประกอบบางส่วนชัดเจนมากขึ้น
2) หลังจากนั้นจะทำตัดค่า Threshold เพื่อทำการแบ่งวัตถุออกจากพื้นหลัง เนื่องจากโจทย์ให้ค่าพื้นที่วงกลมมาแต่ค่านั้นรวมส่วนของดอกไม้ไปด้วย ดังนั้นเราต้องทำการแยกหาจำนวน Pixels ของดอกไม้ที่ไม่รวมวงกลม และจำนวน Pixels ของวงกลมที่รวมดอกไม้ หลังจากนั้นเราสามารถนำมาหาพื้นที่ของดอกไม้ได้จากอัตราส่วน

$$\text{พื้นที่ส่วนของดอกไม้ } (CM^2) = \frac{\text{พื้นที่ของทั้งหมดของวงกลมรวมดอกไม้ } (CM^2)}{\text{จำนวน } Pixels \text{ ของวงกลมทั้งหมดรวมดอกไม้ } (Pixels)} \times \text{ จำนวน } Pixels \text{ ของดอกไม้ } (Pixels)$$

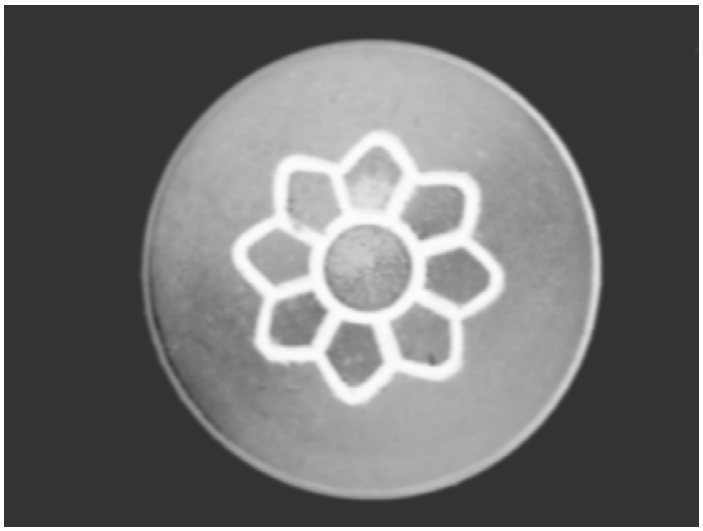
นาย ศุภากร ยุคลธง รหัสนักศึกษา 550610548

การคำนวณหา ส่วนรูปดอกไม้สีขาวจากรูป Flower_Snack.pgm



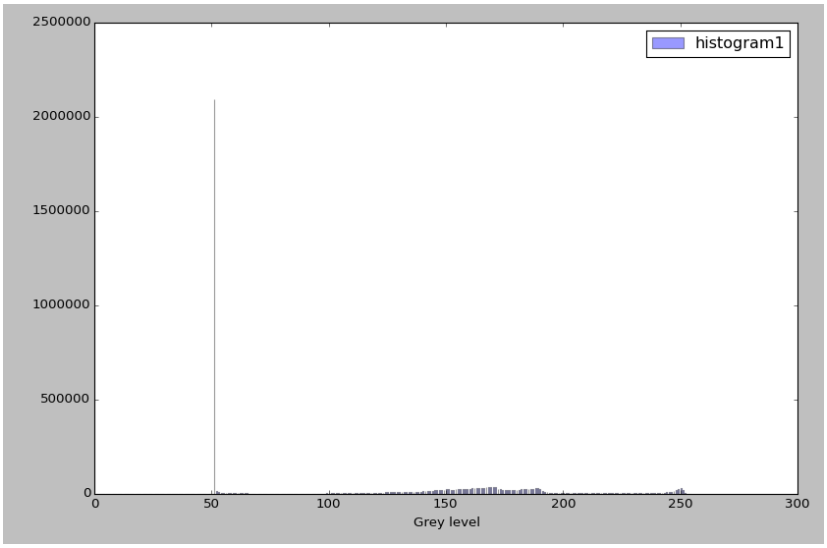
รูป Input (Flower_Snack.pgm)

เอาไปผ่าน Gaussian Low Pass Filter Cutoff 50



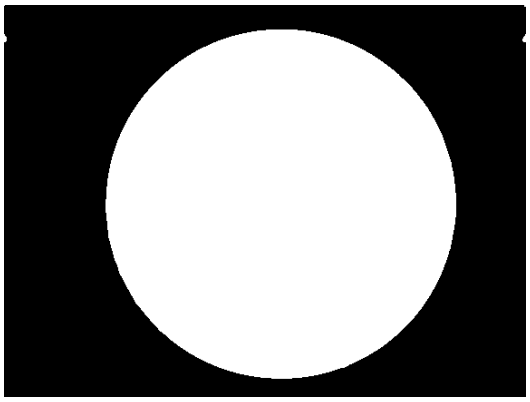
รูป (Flower_SnackGaussianLowPassFilter50.pgm) เป็นภาพที่ได้จากการเบลอด้วย Gaussian Low pass Filter ด้วย Cutoff 50

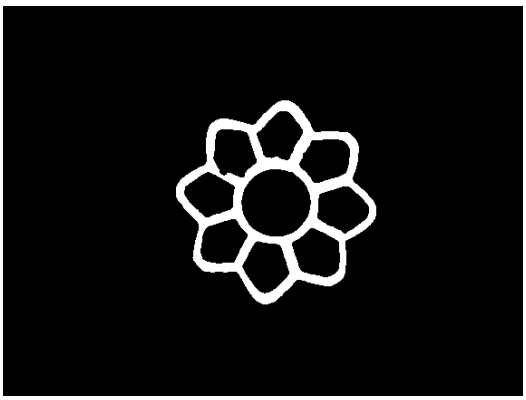
นำรูปไป plot Histogram



หลังจากวิเคราะห์จาก Histogram แล้วปรากฏว่า ค่า Grey Scale ของวงกลมและดอกไม้ มีค่าอยู่ในช่วง 55 ถึง 255 หลังจากนั้นจึงนำไปทำการ Segmentation จะได้ผลดังรูปด้านล่าง พร้อมกับทำการนับ pixels ก็จะได้ค่าประมาณ 1,864,794 pixels

หลังจากวิเคราะห์จาก Histogram แล้วปรากฏว่า ค่า Grey Scale ของดอกไม้ มีค่าอยู่ในช่วง 229 ถึง 255 หลังจากนั้นจึงนำไปทำการ Segmentation จะได้ผลดังรูปด้านล่าง พร้อมกับทำการนับ pixels ก็จะได้ค่าประมาณ 194,692 pixels





หลังจากนั้นนำไปพื้นที่รูปดอกไม้ตามสมการ

สัดส่วนที่ด้านบนจะได้พื้นที่ประมาณ

0.271450465842 ตารางเซนติเมตร

นาย ศุภากร ยุคลธง รหัสนักศึกษา 550610548

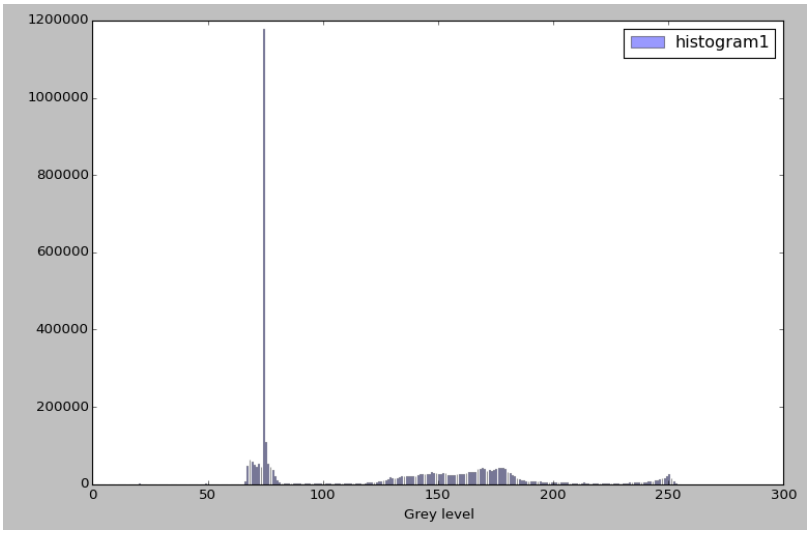
การคำนวณหา ส่วนรูปปูสีขาวของรูป Crab.pgm

รูป Input (Crab.pgm)

เอาไปผ่าน Gaussian Low Pass Filter Cutoff 50

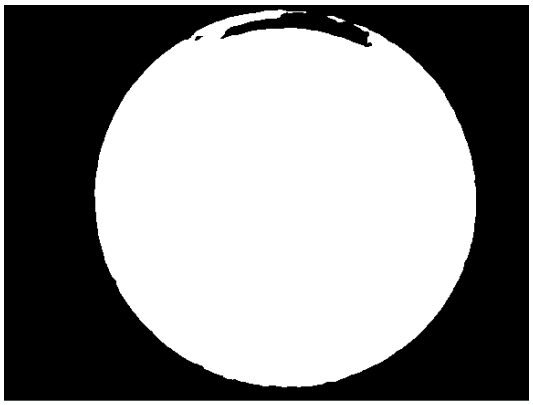
รูป (CrabGaussianLowPassFilter50.pgm) เป็นภาพที่ได้จากการเบลอด้วย Gaussian Low pass Filter ด้วย Cutoff 50

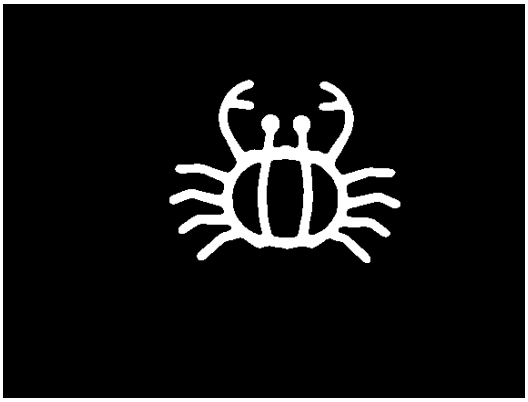
นำรูปไป plot Histogram



หลังจากวิเคราะห์จาก Histogram แล้วปรากฏว่า ค่า Grey Scale ของวงกลมและปู มีค่าอยู่ในช่วง 83 ถึง 255 และยังพบว่ามีเศษบางส่วนที่ขาดหายไปซึ่งอยู่ในช่วง 0 ถึง 65 หลังจากนั้นจึงนำไปทำการ Segmentation จะได้ผลดังรูปด้านล่าง พร้อมกับทำการนับ pixels ก็จะได้ทั้งคู่จะได้ค่าประมาณ 2,151,726 pixels

หลังจากวิเคราะห์จาก Histogram แล้วปรากฏว่า ค่า Grey Scale ของปู มีค่าอยู่ในช่วง 220 – 255 หลังจากนั้นจึงนำไปทำการ Segmentation จะได้ผลดังรูปด้านล่าง พร้อมกับทำการนับ pixels ก็จะได้ค่าประมาณ 225,636 pixels





หลังจากนั้นนำไปพื้นที่รูปปูตามสมการสัดส่วนด้ายบนจะได้พื้นที่ประมาณ 0.290050487841 ตารางเซนติเมตร

Source Code ที่ใช้ในการทำงาน

Link GitHub → https://github.com/SupakornYu/ImageProcessingHW3

```
import numpy as np
import math
import cmath
import matplotlib.pyplot as plt
import threading

class ImageSegmentationEx:
    def readPGMImage(self,path):
        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

def buildPGMFile(self,fileName,width,height,greyLevel,pgmData):
    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 plotHistogramFromArray(self,histogram_arr):
    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 moveAxispgmDataBeforeFourier(self,pgmData,pgmSize):
        for j in range(int(pgmSize[0])):
            for i in range(int(pgmSize[1])):
                x = math.pow(-1.0,float(i+j))
                pgmData[i][j] = float(pgmData[i][j])*x
        return pgmData
        #function for moving axis to center before using FFT.


    def convertToFourier(self,pgmData):
        return np.fft.fft2(pgmData)
        #function for fast fourier transform converting.


    def GaussianLowPassFilter(self,filename,cutoff):
        pgmVer,pgmComment,pgmSize,pgmGreyscale,pgmData,htg =
self.readPGMImage(str(filename)+".pgm")
        #print pgmSize
        pgmData = self.moveAxispgmDataBeforeFourier(pgmData,pgmSize)
        pgmDataFourier = self.convertToFourier(pgmData)
        hFilter = np.zeros((int(pgmSize[1]),int(pgmSize[0])),dtype=np.float)
        hFilter.fill(0)
        m = float(pgmSize[0])
        n = float(pgmSize[1])
        for j in range(int(pgmSize[0])):
            for i in range(int(pgmSize[1])):
                dUV = float(np.sqrt(((float(j)-(m/2.0))**2.0)+((float(i)-(n/2.0))**2.0)))
                hFilter[i][j] = np.exp(-1.0*((dUV**2.0)/(2.0*(cutoff**2.0))))
        pgmResult = pgmDataFourier*hFilter
        pgmResult = np.abs(np.fft.ifft2(pgmResult))
        pgmResult = self.moveAxispgmDataBeforeFourier(pgmResult,pgmSize)
```

```
        pgmResult = np.abs(np.round(pgmResult,0).astype(int))

self.buildPGMFile(str(filename)+"GaussianLowPassFilter"+str(cutoff),pgmSize[0],pgmSize[1],pgmGreyscale,pgmResult)

        #function for taking gaussian filter to image.


    def segmentSetGreyScaleWithMultiThreshold(self,filename,outputfilename,greyscaleWasSet,ThresholdMin,ThresholdMax):

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

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

        pgmDataOutput.fill(0)

        countPixel = 0

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

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

                if pgmData[i][j] >= ThresholdMin and  pgmData[i][j] <= ThresholdMax:

                    pgmDataOutput[i][j] = greyscaleWasSet

                    countPixel+=1

                else:

                    pgmDataOutput[i][j] = 0


self.buildPGMFile(str(outputfilename)+"Threshold"+str(greyscaleWasSet),pgmSize[0],pgmSize[1],pgmGreyscale,pgmDataOutput)

        return countPixel

        #pixels in grey scale range(ThresholdMin,ThresholdMax) are assigned as greyscaleWasSet parameter.


    def calculateArea(self,countPixelObject,countPixelBackGroundCircle,backGroundCircleArea):

        return (float(backGroundCircleArea)/float(countPixelBackGroundCircle))*float(countPixelObject)

        #calculate from Ratio for getting area.


    def plotThreadHistogram(self,htgData):
```

```
        self.plotHistogramFromArray(htgData)


if __name__ == "__main__":



    myLib = ImageSegmentationEx()


    #myLib.GaussianLowPassFilter("Flower_Snack",30) #Function For Blur Picture By gaussian filter



    countObj =
myLib.segmentSetGreyScaleWithMultiThreshold("CrabGaussianLowPassFilter50","CrabGaussian_Cra
b",255,220,255)

    countCir_part =
myLib.segmentSetGreyScaleWithMultiThreshold("CrabGaussianLowPassFilter50","CrabGaussian_Cra
b_Circle_little_part",255,0,65)

    countCir =
myLib.segmentSetGreyScaleWithMultiThreshold("CrabGaussianLowPassFilter50","CrabGaussian_Cra
b_Circle",255,83,255)

    countCir = countCir + countCir_part

    print str("Crab Area : ")+str(myLib.calculateArea(countObj,countCir,2.766))+str(" obj :
")+str(countObj)+str(" cir : ")+str(countCir)




    pgmVer,pgmComment,pgmSize,pgmGreyscale,pgmData,htg =
myLib.readPGMImage("Flower_SnackGaussianLowPassFilter50.pgm")

    print pgmSize

    print pgmData.shape


    countObj =
myLib.segmentSetGreyScaleWithMultiThreshold("Flower_SnackGaussianLowPassFilter50","Flower_
Snack_Gaussian_Flower",255,229,255)
```

```
    countCir = myLib.segmentSetGreyScaleWithMultiThreshold("Flower_SnackGaussianLowPassFilter50","Flower_Snack_Gaussian_Flower_Circle",255,55,255)

    print str("Flower Area : ")+str(myLib.calculateArea(countObj,countCir,2.6))+str(" obj : ")+str(countObj)+str(" cir : ")+str(countCir)



    pgmVer,pgmComment,pgmSize,pgmGreyscale,pgmData,htgCrab = myLib.readPGMImage("CrabGaussianLowPassFilter50.pgm")

    pgmVer,pgmComment,pgmSize,pgmGreyscale,pgmData,htgFlower = myLib.readPGMImage("Flower_SnackGaussianLowPassFilter50.pgm")

    print str("pgmsize : ")+str(pgmSize)

    print str("data size : ")+str(pgmData.shape)


    """

    print "starting threading command"

    t = threading.Thread(target=myLib.plotThreadHistogram, args=(htgCrab,))

    t.start()

    t = threading.Thread(target=myLib.plotThreadHistogram, args=(htgFlower,))

    t.start()

    print "ending threading command"

    """


    myLib.plotHistogramFromArray(htgCrab)

    myLib.plotHistogramFromArray(htgFlower)
```