```
######################## SERKAN CAN
170220852##############################

import cv2
import numpy as np
import matplotlib.pyplot as plt
import math

def gauss_kernel_olustur(x,y,sigma,K):
    kernel = np.empty((x, y)) #oluşturmak istediğimiz filtrenin
boyutunda bos bir matris olusturuyoruz
    x_max = math.floor(x/2) # koordinat düzleminde max x noktasını
buluyoruz
    y_max = math.floor(y/2)# koordinat düzleminde max y noktasını
buluyoruz
    for satir in range(-x_max, x_max+1):
        for sutun in range(-y_max, y_max+1):
            r_kare = satir**2 + sutun**2
            kernel[x_max+satir, y_max+sutun] =
K*np.exp(-(r_kare/(2*sigma**2)))
    return kernel / kernel.sum()
def
filtre(I,kernel,isMedian=False,isSharpening=False,boyut=0,isShow=True):
# korelasyon ile kerneli imgelere uyguluyoruz
    x,y=I.shape # fotoğrafın x ve y boyutları
    cikti=np.zeros([x,y])
    if(isMedian==False):
        m,n=kernel.shape # kernelin x ve y boyutları
    else:
        m=n=boyut
    x_max=m//2 # xin yarısını alıp aşağıya yuvarlıyor, bu da fotoğrafın
ne kadar genişletileceğini belirliyor
    y_max=n//2
    genislet=np.pad(I ,((x_max,x_max),(y_max,y_max)),
constant_values=((0,0),(0,0))) #(giriş, (yukardan ne kadar
genişletileceği, aşağıdan,soldan, sağdan), genişletilen bölgeye
koyulacak değerler)
    geciciMatris=np.zeros_like(genislet)
    xM,yM=geciciMatris.shape[0],geciciMatris.shape[1]
    geciciMatris[x_max:xM-x_max,y_max:yM-y_max]=I
    for i in range(x):
        for j in range(y):
            if(isMedian):
```

```
                Temp= geciciMatris[i:i+m,j:j+m]
                cikti[i,j]=np.median(Temp[:])
            else:
                Temp= geciciMatris[i:i+m,j:j+m]*kernel
                cikti[i,j]=np.sum(Temp[:])
    if(isSharpening==False):
        cikti=cikti.astype(np.uint8)
    if(isShow):
        plt.subplot(1,1,1)
        plt.imshow(cikti,cmap="gray")
        plt.show()
    return cikti
def gammaCorrection(src, gamma):
    invGamma = 1 / gamma

    table = [((i / 255) ** invGamma) * 255 for i in range(256)]
    table = np.array(table, np.uint8)

    return cv2.LUT(src, table)
I=cv2.imread("circles1.png",0)
I=gammaCorrection(I,1.6) # daha iyi ayırabilmek için gamma filtresi
ekledim
kernel= gauss_kernel_olustur(19,19,sigma=3,K=1)
cikti=filtre(I,kernel,isShow=False)

sobelx=cv2.Sobel(cikti,cv2.CV_64F,1,0,ksize=11)
sobely=cv2.Sobel(cikti,cv2.CV_64F,0,1,ksize=11)
th=np.arctan2(sobelx,sobely)
kenarlar= cv2.Canny(cikti,10,10)
x,y= I.shape
acc= np.zeros([x,y])
r=25
for i in range(x):
    for j in range(y):
        if(kenarlar[i,j]!=0):
           xx= i + r*math.cos(th[i,j])
           yy = j +r* math.sin(th[i,j])
           xx2= i - r*math.cos(th[i,j])
           yy2 = j -r* math.sin(th[i,j])
           if(xx>x or xx2>x or yy>y or yy2>y):
               continue
           else:
               acc[int(xx),int(yy)]+=1
```

```
                acc[int(xx2),int(yy2)]+=1
P=cv2.imread("circles1.png")
b,g,r=cv2.split(P)
P=cv2.merge([r,g,b])
for i in range(x):
    for j in range(y):
        if(acc[i,j]>9):
            cv2.circle(P,(j,i),25,(0,0,255),2)


plt.subplot(1,2,1)
plt.imshow(P,cmap='gray')
plt.subplot(1,2,2)
plt.imshow(acc,cmap='gray')
plt.show()
```