```
####################### SERKAN CAN 170220852
############################
import cv2
import numpy as np
import math
import matplotlib.pyplot as plt
def gausNoise(img):
    gauss = np.random.normal(0,1,img.size)
    gauss =
gauss.reshape(img.shape[0],img.shape[1],img.shape[2]).astype('uint8')
    img_gauss = cv2.add(img,gauss)
    cv2.imshow('gausgurultuorijinal',img_gauss)
def ortalama_kernel_olustur(x,y,deger):
    kernel= np.full((x,y),deger) # istediğimiz filtre boyutunda içine
istediğimiz değeri atayabileceğimiz fonksiyon
    return kernel/kernel.sum()

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
I=cv2.imread("yugo.jpg",0)
cv2.imshow("orijinal",I)

laplace_kernel=np.array([[0,-1,0],
           [-1,4,-1],
           [0,-1,0]])
######### LAPLACE ############################
# filtre(I,laplace_kernel,isSharpening=True)#
##############################################

########## ORTALAMA ######################
#kernel= ortalama_kernel_olustur(5,5,1)####
#filtre(I,kernel)                        ###
##########################################


########## GAUS ###################################
#kernel= gauss_kernel_olustur(25,25,sigma=4,K=1)##
#filtre(I,kernel)                                ##
#################################################
######## MEDIAN ##################################
```

```
# filtre(I,kernel=None,isMedian=True,boyut=10)#
##############################################

############## SOBEL ###################################
#sobelx=np.array([[-1,-2,-1],                           #
#                 [0,0,0],                              #
#                 [1,2,1]])                             #
#sobely=sobelx.T                                        #
#a=filtre(I,sobelx,isSharpening=True,isShow=False)      #
#b=filtre(I,sobely,isSharpening=True,isShow=False)      #
#sonuc=np.sqrt(a**2+b**2)                               #
                                                        #
                                                        #
########################################################
```

UZAMSAL SÜZME ÖDEV

Orijinal İmge

```
gausNoise(I)
```

%Gauss gürültüsü eklenmiş

```
kernel= ortalama_kernel_olustur(5,5,1)
filtre(I,kernel)
#ORTALAMA
```

```
kernel=
gauss_kernel_olustur(25,25,sigma=4,K=1)
filtre(I,kernel)  #GAUS
```

```
filtre(I,kernel=None,isMedian=True,boyut=10) #Median
```

```
kernel=np.array([[0,-1,0],
                 [-1,4,-1],
                 [0,-1,0]])
filtre(I,kernel,isSharpening=True) #LAPLAC
```

```
sobelx=np.array([[-1,-2,-1],
                 [0,0,0],
                 [1,2,1]])
sobely=sobelx.T #TRANSPOSE
a=filtre(I,sobelx,isSharpening=True,isShow=False)
b=filtre(I,sobely,isSharpening=True,isShow=False)
sonuc=np.sqrt(a**2+b**2)
#SOBEL
```

```
prewitx=np.array([[-1,0,1],
                  [-1,0,1],
                  [-1,0,1]])
prewity=sobelx.T #TRANSPOSE
a=filtre(I,prewitx,isSharpening=True,isSho
=False)
b=filtre(I,prewity,isSharpening=True,isSho
=False)
sonuc=np.sqrt(a**2+b**2)#PREWIT
```