SAYISAL İŞARET İŞLEME PROJE

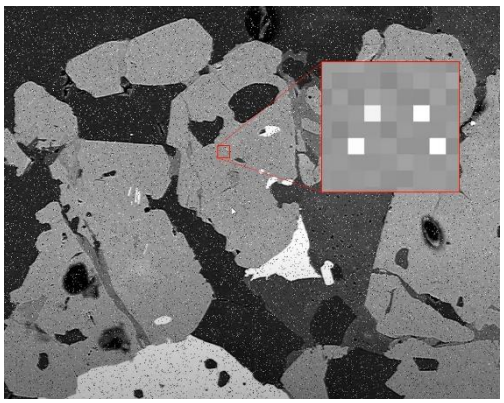
*Bu proje medyan filtresi (median filter) ve gauss filtresini (gaussian filter) konu alacaktır.

1-Medyan filtresi (Median Filter)

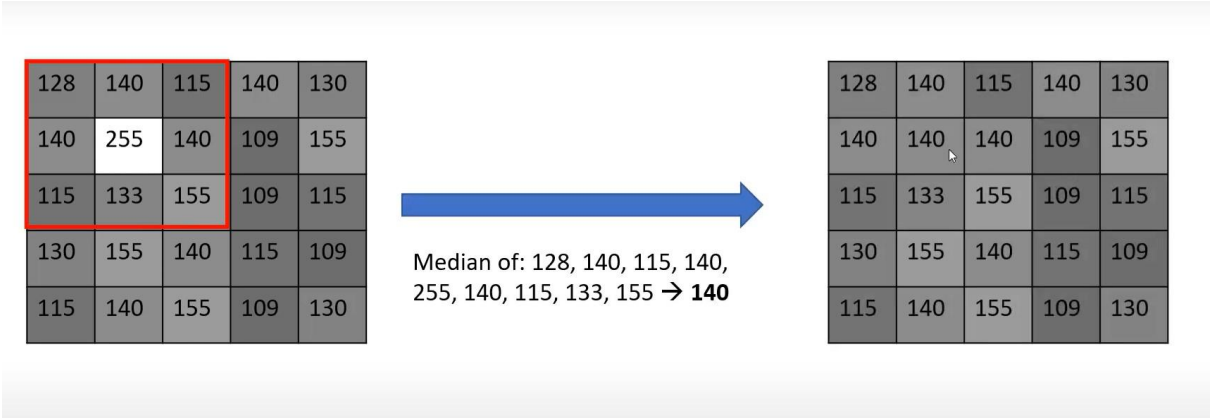
Medyan filtresi gürültü bastırma için kullanılan, lineer olmayan bir işleme tekniğidir. 1970'li yıllarda Tukey tarafından bulunan bu filtre resim işleme alanında kullanılmaktadır.

Medyan filtrelemenin çalışma şekli bir pencerenin, filtrelenecek resmin noktaları üzerinde hareket ederken pencere içindeki değerlerin ortanca değeri ile pencerenin merkezindeki değeri değiştirerek gerçekleştirilen bir filtreleme türüdür. Bu işlem uygulandığı zaman ortaya çıkan resim genelde daha düzgündür.

Klasik düzleştirme prosedürlerinde lineer bir düşük geçiren filtre kullanılır. Her ne kadar bu en uygun yöntem olsa da bazı durumlarda medyan filtreleme daha çok iş görür. Örneğin düşük geçiren filtre kenarları bulanıklaştırırken medyan filtreleme keskin kenarları korumaktadır.



*Filtrenin uygulanışını bir resim üzerinde gösterecek olursak giriş sinyali seçtiğimiz bir nokta olacaktır.



*Daha sonra giriş sinyalindeki koordinatları yan yana yazarak bu değerlerin medyanı alınır.

*Daha sonra bu medyan değer koordinatların ortasındaki değer ile değiştirilerek bir sonraki koordinat bloğuna geçilir ve bu işlem bütün resme uygulanana kadar devam eder.

*Bir formül ile ifade edecek olursak:

$$I_{\text{med}}(x,y) = \text{median}\left\{I(x+i,y+j) \mid -\tfrac{n}{2} \leq i,j \leq \tfrac{n}{2}\right\}$$

Imed: Filtrelenmiş koordinatlar:

I(x+i,y+i): (x,y) noktamızın çevresindeki piksel değerlerini temsil ediyor.

N: komşuluk boyutu diyebiliriz mesela 3x3 ya da 8x8 gibi.

Bu ifade, bir pikselin değerini, belirli bir komşuluk boyutu içindeki piksellerin sıralanmış değerleri arasındaki ortanca değer ile değiştirir.

2-Gauss Filtresi (Gaussian Filter)

Gauss filtresi (Gaussian filter), görüntü işleme ve sinyal işleme alanlarında kullanılan bir düşük geçişli filtre türüdür.

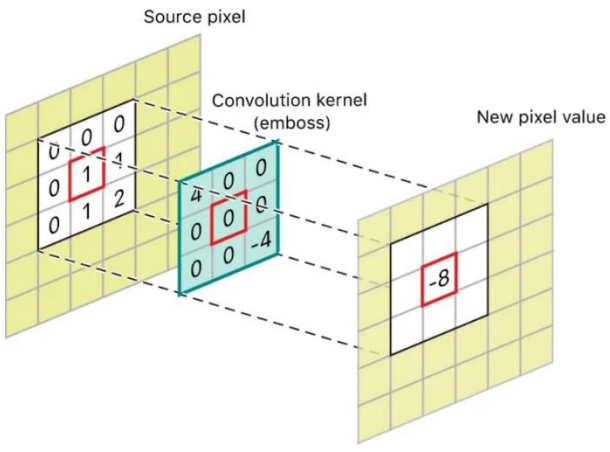
Gauss filtresi, bir görüntü veya sinyalin düzleştirilmesi (blurring) veya gürültü azaltma amacıyla kullanılır. Gauss filtresi, her bir pikselin etrafındaki komşu piksellerin ağırlıklı ortalamasını alarak çalışır. Bu ağırlıklar, Gauss fonksiyonu tarafından belirlenen bir ağırlık dağılımı kullanılarak hesaplanır. Bu dağılım, merkezi pikselin en fazla etkileyip, uzak pikselleri daha az etkileyerek bir tür yumuşatma (blurring) etkisi yaratır.

Gauss filtresi genellikle kenarları yumuşatmak, gürültüyü azaltmak veya görüntüyü düzleştirmek amacıyla kullanılır. Özellikle bilgisayarlı görü görüntü işleme uygulamalarında ve dijital fotoğrafçılıkta sıklıkla karşılaşılan bir tekniktir.

*Formül ile ifadesi şu şekildedir:

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-x^2/(2\sigma^2)}$$

σ: Standart sapmayı ifade eder.



*Çalışma mantığında ilk olarak “Gaussian Kernel” adında bir konvolüsyon matrisi belirlenir.

*Örneğin filtre edilecek şey bir resim olsun. Bu resimde bulunan her pixele seçtiğimiz matris ile konvolüsyon işlemi uygulanır ve yeni değer konvolüsyonu alınan değer ile değiştirilir.

*Her bir piksel, kendisi ve çevresindeki piksellerin değerleriyle çarpılıp ağırlıklandırılır, sonra bu değerler toplanarak yeni piksel değeri elde edilir. Bu işlem, her pikseli çevresindeki piksellerle ortalama birleştirerek görüntüyü yumuşatır.

Benzerlikler:

1-Gürültü giderme:

Hem Gaussian hem de Median filtreleri, dış görüntülerindeki gürültüyü azaltmak için kullanılan filtreleme teknikleridir.

2-Görüntü iyileştirme:

Her iki filtre de görüntü iyileştirmek ve netliği artırmak amacıyla kullanılır.

3-Pre-proccessing:

İki filtre de görüntü işleme aşamasında kullanılmıştır, bu aşama gürültüyü giderme ve görüntü kalitesini artırma amacıyla kullanılır.

Farklılıklar:

1-Çalışma prensipleri:

Gauss filtresi, bir pikselin etrafındaki piksellerin ağırlıklı ortalamasını alarak çalışır. Medyan filtresi ise bir pikselin etrafındaki piksellerin sıralanmış listesindeki ortanca değeri kullanır.

2-Görüntü iyileştirme:

Gauss filtresi, daha genel anlamda bazı spesifik durumlar haricinde medyan filtresine göre daha iyi sonuçlar vermektedir.

3-Kenar keskinliği

Medyan filtresi, kenarları daha iyi korur çünkü ortanca değer, piksellerin sıralanmış listesindeki gerçek değeri temsil eder. Bu, medyan filtresini kenar koruma konusunda daha etkili kılar.

4-Hesaplama zorluğu

Gauss filtresi, hesaplama açısından daha hızlıdır çünkü ağırlıklı ortalama, medyan filtresindeki sıralama işlemine kıyasla daha hızlı kalmaktadır.

Üstün ve zayıf yönler:

Birbirlerine göre üstün yanları:

Gaussian	**Median**
*Hızlı hesaplama süresi.	*Kenar koruma açısından daha etkili.
*Daha yumuşak, sürekli bir görüntü sonucu.	*Tuz ve biber gürültüsü gibi aykırı değerlere karşı daha dayanıklı.
*Genel kullanım için daha uygun.	*Kenar detaylarının önemli olduğu durumlarda daha iyi sonuçlar veren bir seçenektir.

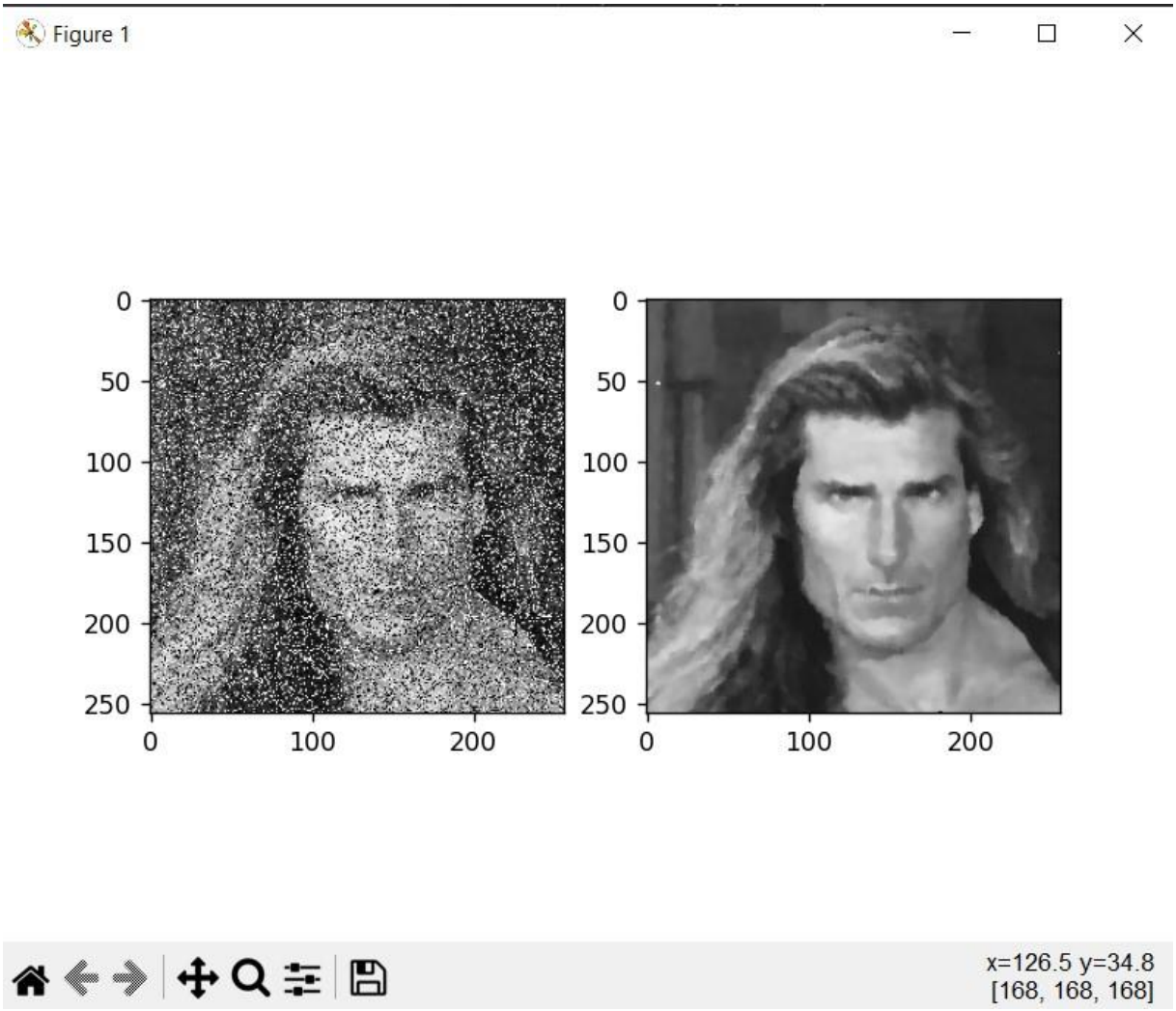
Birbirlerine göre zayıf yanları:

Gaussian	**Median**
*Medyana göre kenarları biraz daha bulanıklaştırabilir	*Gaussa göre daha yavaş bir hesaplama süresine sahiptir.
*Medyana kıyasla aykırı değerlere karşı daha dirençsizdir	*Daha iyi kenar koruma sağladığından daha pürüzsüz sonuçlar vermeyebilir.
	*Gauss kadar genel kullanıma uygun değildir.

MEDYAN FİLTRESİ İÇİN PYTHON KODU:

```
import cv2 #opencv kütüphanesi
import os
import matplotlib.pyplot as plt #matplotlib kütüphanesi
import numpy as np #sayısal işlemler için numPy kütüphanesi

def medianFiltering():
root = os.getcwd() #root değişkenine mevcut durumu atma
imgPath = os.path.join(root, 'resimDosyasi', 'tesla.jpg')
#resim dosyamızı implement etme

img = cv2.imread(imgPath)  #resmi okuyup img değişkenine atma
imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

#renk formatını BGR den RGB ye çevirme
noisyImg = imgRGB.copy()
noiseProb = 0.05 #resme gürültü ekleme
noise = np.random.rand(noisyImg.shape[0], noisyImg.shape[1])
#her piksel için [0,1] aralığında rastgele değerler oluşturma

noisyImg[noise < noiseProb / 2] = 0
 noisyImg[noise > 1 - noiseProb / 2] = 255

#Gürültü değeri noiseProb / 2'den küçük olan pikseller 0'a,
noiseProb / 2'den büyük olan pikseller ise 255'e ayarlanır.
İmgFilter = cv2.medianBlur(noisyImg, 5)
#Bu satır, OpenCV'nin `medianBlur` fonksiyonunu kullanarak 5x5 boyutlu bir çekirdek ile
resme ortanca filtre uygular.
```

```
plt.figure()
plt.subplot(121)
plt.imshow(noisyImg)
plt.subplot(122)
plt.imshow(imgFilter)
plt.show()
#resmi göstermek için yazılan kodlar
if __name__ == '__main__':
 medianFiltering()
```

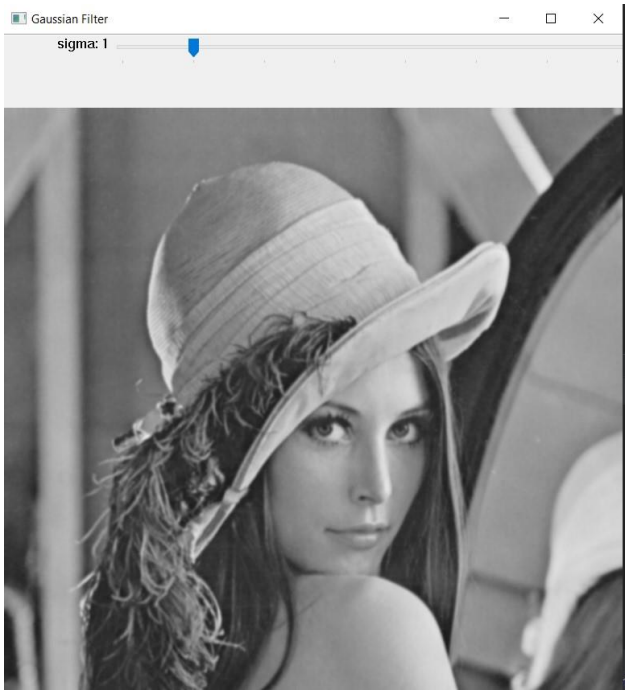


(Medyan filtresi kodumuz ile düzeltilmiş, salt&pepper gürültüsü yüksek bir resim örneği)

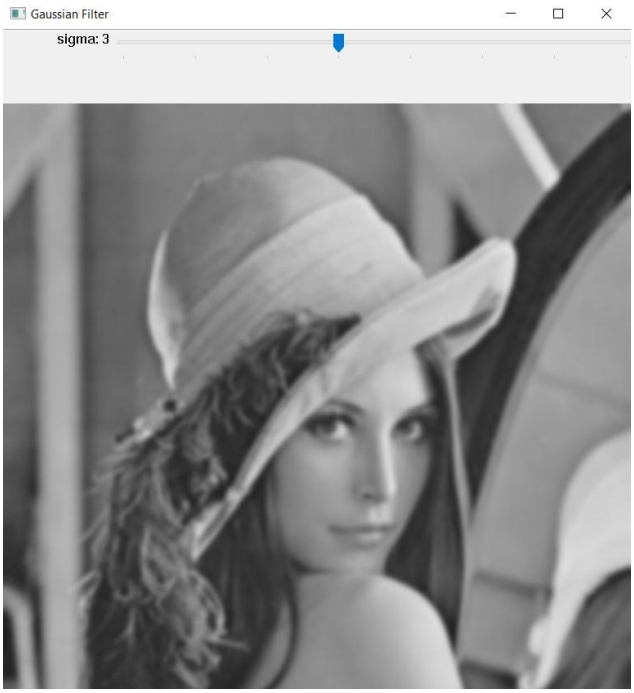
GAUSS FİLTRESİ İÇİN PYTHON KODU:

```
import cv2 as cv #OpenCV kütüphanesini 'cv' takma adıyla içeri aktarıyoruz.
import os #İşletim sistemi işlemleri için 'os' kütüphanesini içeri aktarıyoruz.
import numpy as np #NumPy kütüphanesini 'np' takma adıyla içeri aktarıyoruz.
import matplotlib.pyplot as plt #Matplotlib kütüphanesini 'plt' takma adıyla içeri aktarıyoruz.

def callback(input):
    pass
#Belirli bir boyutta ve standart sapmaya sahip bir Gauss çekirdeği oluşturan fonksiyon.
def gaussianKernel(size, sigma):
    kernel = cv.getGaussianKernel(size, sigma)
#OpenCV ile Gauss çekirdeği oluşturuyoruz.
    kernel = np.outer(kernel, kernel)
#Daha sonra oluşturduğumuz vektörü dış çarpım ile matrise çeviriyoruz.
    return kernel

def gaussianFiltering(): # Gauss filtreleme işlemini gerçekleştiren ana fonksiyon.
root = os.getcwd()
#Şu anki çalışma dizinini alıyoruz.
    imgPath = os.path.join(root, 'resimDosyasi2', 'tesla2.jpg')  # Resmin dosya yolunu
oluşturuyoruz.
    img = cv.imread(imgPath)  #OpenCV ile resmi okuyoruz.

    n = 51 #Gauss çekirdeği boyutu
    fig = plt.figure()  #Matplotlib ile bir figure oluşturuyoruz.
    plt.subplot(121) #İki alt grafikten ilki için konfigürasyon yapıyoruz.
    kernel = gaussianKernel(n, 8) #Belirli bir boyutta ve standart sapmaya sahip Gauss
çekirdeği oluşturuyoruz.
    plt.imshow(kernel)  #Çekirdeği görselleştiriyoruz.
```

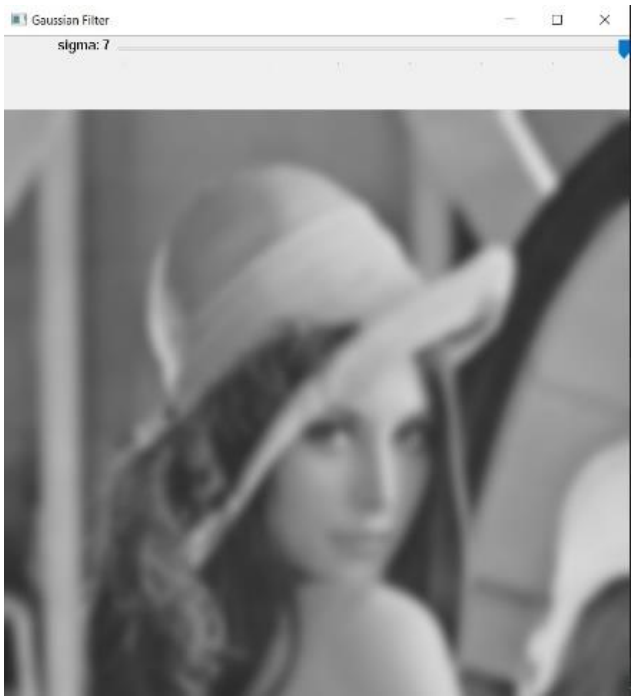
```
    ax = fig.add_subplot(122, projection='3d')  #İki alt grafikten ikincisi için konfigürasyon yapıyoruz.
    x = np.arange(0, n, 1) #X ekseni değerleri
    y = np.arange(0, n, 1) #Y ekseni değerleri
    X, Y = np.meshgrid(x, y)  #2D ızgara oluşturuyoruz.
    Z = kernel.flatten()  #Çekirdeği düzleştirerek Z ekseni değerleri elde ediyoruz.
    ax.plot_surface(X, Y, kernel, cmap='viridis')  #3D yüzey çizdirme işlemi.
    plt.show()  #Oluşturduğumuz görseli görüntülüyoruz.
    winName = 'Gaussian Filter' #Pencere adı
    cv.namedWindow(winName)  #OpenCV'de bir pencere oluşturuyoruz.
    cv.createTrackbar('sigma', winName, 1, 20, callback)  #Bir trackbar ekliyoruz.
    height, width, _ = img.shape  #Resmin yüksekliğini, genişliğini ve kanal sayısını alıyoruz.
    scale = 1 / 4  #Boyutu küçültmek için bir ölçek belirliyoruz.
    width = int(width * scale)  #Genişliği ölçeğe göre ayarlıyoruz.
    height = int(height * scale)  #Yüksekliği ölçeğe göre ayarlıyoruz.
    img = cv.resize(img, (width, height))  #Resmi yeni boyutlara göre yeniden boyutlandırıyoruz.


while True:
if cv.waitKey(1) == ord('q'):  #döngü için.
break
        sigma = cv.getTrackbarPos('sigma', winName)  #Trackbar'dan sigma değerini alıyoruz.
        imgFilter = cv.GaussianBlur(img, (n, n), sigma)  #OpenCV ile Gauss filtreleme işlemi gerçekleştiriliyor.
        cv.imshow(winName, imgFilter)  #Filtrelenmiş resmi gösteriyoruz.
        cv.destroyAllWindows() #Tüm pencereleri kapatıyoruz.
if __name__ == '__main__':
gaussianFiltering()
```



(1)



(2)



(3)

(Kodun bir resme uygulanışı yandaki gibidir.)

KAYNAKÇA:

Two-Dimensional Digital Signal Prcessing II pp 161–196

Median Filtering: A New Insight - Sebastián A. Villar, Sebastián Torcida &Gerardo G. Acosta

Comparison of Gaussian and Median Filters to Remove Noise in Dental Images (Available in SSRN eLibrary)

Understanding Digital Signal Processing with MATLAB® and Solutions.

Digital Signal Processing 4th Edition (John G.Proakis, Dimitris K. Manolakis)