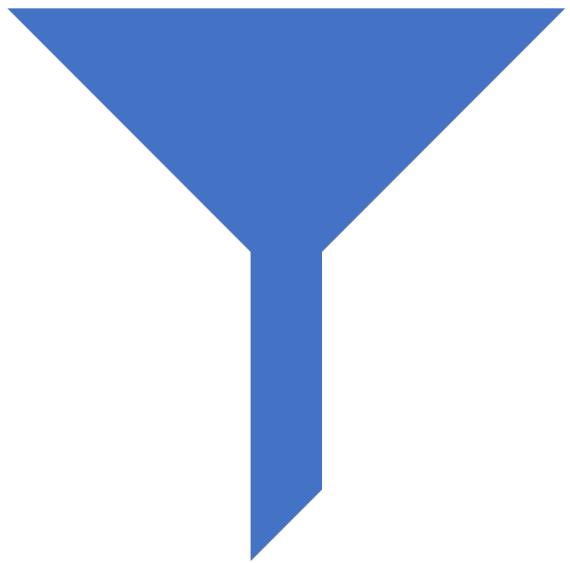




KIRIKKALE ÜNİVERSİTESİ BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ BİL1006-BİLGİSAYAR GRAFİĞİNE GİRİŞ

Doç. Dr. Serkan SAVAŞ



FİLTRELEME İŞLEMLERİ

- Görüntüleri Yumuşatma
 - 2D Evrişim (Görüntü Filtreleme)
 - Görüntü Bulanıklaştırma (Görüntü Düzeltme)
 - Ortalama
 - Gauss Bulanıklığı
 - Medyan Bulanıklık
 - İkili Filtreleme

Görüntüleri Yumuşatma

2D Evrişim (Görüntü Filtreleme)

Tek boyutlu sinyallerde olduğu gibi, görüntüler de çeşitli alçak geçiren filtreler (LPF), yüksek geçiren filtreler (HPF) vb. ile filtrelenebilir.

LPF, gürültülerin giderilmesine, görüntülerin bulanıklaştırılmasına vb. yardımcı olur.

HPF filtreleri, görüntülerde kenarların bulunmasına yardımcı olur.

2D Evrişim (Görüntü Filtreleme)

OpenCV'nin, bir çekirdeği bir görüntü ile birleştirmek için bir `cv2.filter2D()` fonksiyonu vardır. Örnek olarak, bir görüntü üzerinde bir 5x5 ortalama filtre çekirdeği aşağıdaki gibi görünecektir:

$$K = \frac{1}{25} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

2D Evrişim (Görüntü Filtreleme)

İşlem aşamaları: Bu çekirdeği bir pikselin üzerinde tut,
25 pikselin tamamını bu çekirdeğin altına ekle,
ortalamasını al,
merkezi pikseli yeni ortalama değerle değiştir.

Görüntüdeki tüm pikseller için bu işleme devam eder.

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
from google.colab.patches import cv2_imshow
img = cv2.imread('high_way.jpg')
```



```
kernel = np.ones((5,5),np.float32)/25
dst = cv2.filter2D(img,-1,kernel)
```



```
cv2_imshow(img)
cv2_imshow(dst)
```

2D Evrişim (Görüntü Filtreleme)

Görüntü Bulanıklaştırma (Görüntü Düzeltme)

Görüntü bulanıklaştırma, görüntüyü düşük geçişli bir filtre çekirdeği ile sararak elde edilir.

Gürültüleri gidermek için kullanışlıdır.

Aslında görüntüden yüksek frekanslı içeriği (örneğin: gürültü, kenarlar) kaldırır.

Yani bu işlemde kenarlar biraz bulanıklaşır.
(kenarları bulanıklaştırmayan bulanıklaştırma teknikleri de vardır.)

OpenCV, temel olarak **dört tür** bulanıklaştırma tekniği sağlar.

Ortalama

$$K = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

- + • Görüntüyü normalleştirilmiş bir kutu filtresiyle evirerek yapılır.
 - Çekirdek alanı altındaki tüm piksellerin ortalamasını alır ve merkezi elemanı değiştirir.
- Bu işlem, `cv2.blur()` veya `cv2.boxFilter()` fonksiyonuyla yapılır.
- Burada çekirdeğin genişliği ve yüksekliği belirtilmelidir.
- 3x3 normalleştirilmiş bir kutu滤resi yandaki gibidir.

Ortalama

```
blured_img = cv2.blur(img, (10,10))
```

◆ blur()

```
void cv::blur ( InputArray src,  
               OutputArray dst,  
               Size ksize,  
               Point anchor = Point(-1,-1) ,  
               int borderType = BORDER_DEFAULT  
 )
```

Python:

```
cv.blur( src, ksize[, dst[, anchor[, borderType]]] ) -> dst
```



◆ boxFilter()

```
void cv::boxFilter ( InputArray  src,  
                    OutputArray dst,  
                    int          ddepth,  
                    Size         ksize,  
                    Point        anchor = Point(-1,-1) ,  
                    bool         normalize = true ,  
                    int          borderType = BORDER_DEFAULT  
    )
```

Python:

```
cv.boxFilter( src, ddepth, ksize[, dst[, anchor[, normalize[, borderType]]]] ) -> dst
```

Ortalama

```
box_img = cv2.boxFilter(img, -1, (10,10))
```

```
#çıktı görüntüsü derinliği (-1 = kaynakla aynı  
demek)
```



Gauss Bulanıklığı

Kutu filtre yerine gauss çekirdeği kullanılır. **cv2.GaussianBlur()** fonksiyonuyla yapılır.

Pozitif ve tek olması gereken çekirdeğin genişliği ve yüksekliği belirtilmelidir.

Standart sapma da sırasıyla X ve Y yönünde, sigmaX ve sigmaY olarak belirtilmelidir.

Yalnızca sigmaX belirtilirse, sigmaY, sigmaX ile aynı kabul edilir. Her ikisi de sıfır olarak verilirse, çekirdek boyutundan hesaplanır.

Gauss bulanıklaştırma, görüntünün gauss gürültüsünü gidermede oldukça etkilidir.

Gauss Bulanıklığı

```
gaussian_img =  
cv2.GaussianBlur(img,  
(7, 7), 0)
```



Gauss Bulanıklığı

◆ GaussianBlur()

```
void cv::GaussianBlur ( InputArray src,  
                      OutputArray dst,  
                      Size kszie,  
                      double sigmaX,  
                      double sigmaY = 0 ,  
                      int borderType = BORDER_DEFAULT  
)
```

Python:

```
cv.GaussianBlur( src, kszie, sigmaX[, dst[, sigmaY[, borderType]]] ) -> dst
```

Src giriş görüntüsü; görüntü, bağımsız olarak işlenen herhangi bir sayıda kanala sahip olabilir, ancak derinlik CV_8U, CV_16U, CV_16S, CV_32F veya CV_64F olmalıdır.

Src ile aynı boyut ve türde dst çıktı görüntüsü.

ksize Gauss çekirdek boyutu: kszie.width ve kszie.height farklı olabilir ancak her ikisi de pozitif ve tek olmalıdır. Veya sıfır olabilirler ve sonra sigmadan hesaplanırlar.

X yönünde sigmaX: Gauss çekirdeği standart sapması.

Y yönünde sigmaY: Gauss çekirdeği standart sapması;

sigmaY sıfırsa, sigmaX'e eşit olacak şekilde ayarlanır, her iki sigma da sıfırsa sırasıyla kszie.width ve kszie.height'tan hesaplanır.

borderType: Border türü. BORDER_WRAP desteklenmiyor.

Gauss Bulanıklığı

Gauss, bilgisayarla görme uygulamalarında belki de en sık kullanılan alçak geçiren filtredir.

Bir piksel değerini interpolasyon yapacak olursak, komşu piksellerinkine benzeme olasılığı daha fazla ve uzak piksellerde daha az olur.

Benzer şekilde, bir görüntüyü yumusatırken, yalnızca maskenin altındaki değerlerin ortalamasını almak yerine ağırlıklı ortalamayı almak daha mantıklıdır.

Gauss Bulanıklığı

Uzak piksellere kıyasla en yakın piksellere daha fazla ağırlık atayan bir dağılım/fonksiyon olarak Gauss dağılımını kullanmanın nedeni budur.

Çıktıyı almak için 2-D Gauss fonksiyonu görüntüyle birleştirilir. Ancak bunun için Gauss fonksiyonuna ayrık bir yaklaşım üretilmesi gereklidir.

Gauss fonksiyonu sonsuz desteği sahip olduğundan (her yerde sıfırdan farklı olduğu anlamına gelir), yaklaşıklık sonsuz büyük bir evrişim çekirdeği gerektirir.

Başka bir deyişle, her piksel hesaplaması için görüntünün tamamına ihtiyaç vardır. Bu nedenle, çekirdek boyutunu kısaltmamız veya sınırlamamız gereklidir.

Gauss Bulanıklığı

Yaklaşık çekirdek ağırlıkları tam olarak 1'e ulaşmaz, bu nedenle ağırlıkları genel çekirdek toplamına göre normalleştirilir.

Aksi takdirde bu, görüntünün kararmasına veya parlamasına neden olur.

Normalize edilmiş bir 3x3 Gaussfiltresi:

$$\frac{1}{16} \times$$

1	2	1
2	4	2
1	2	1

$$\frac{1}{16} \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix} = \frac{1}{16} \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 1 \end{pmatrix}$$

Gauss Bulanıklığı

İlk olarak, Gauss çekirdeği lineer olarak ayrılabilir. Bu, herhangi bir 2 boyutlu filtreyi iki 1 boyutlu filtreye bölebileceğimiz anlamına gelir. Bu nedenle, hesaplama karmaşıklığı $O(n^2)$ 'den $O(n)$ 'e düşürülür.

Gauss Bulanıklığı

Ardışık birden çok Gauss çekirdeği uygulamak, yarıçapı birden çok çekirdek yarıçapının karelerinin toplamının karekökü olan tek, daha büyük bir Gauss bulanıklığı uygulamaya eşdeğerdir.

Bu özelliği kullanarak, ayrılabilir birden fazla filtrenin bir kombinasyonu ile ayrılamaz bir filtreye yaklaşılabilir.

$$\begin{array}{ccccccccc} & & & & & 1 & & & \\ & & & & & 1 & 1 & 1 & \\ & & & & & 1 & 2 & 1 & \\ & & & & & 1 & 3 & 3 & 1 \\ & & & & & 1 & 4 & 6 & 4 & 1 \\ & & & & & 1 & 5 & 10 & 10 & 5 & 1 \\ & & & & & 1 & 6 & 15 & 20 & 15 & 6 & 1 \end{array}$$

Gauss Bulanıklığı

- gauss_kernel = cv2.getGaussianKernel(7,1)
- gaussian_img3 = cv2.sepFilter2D(img,-1,gauss_kernel, gauss_kernel)

Medyan Bulanıklık

cv2.medianBlur() fonksiyonu, çekirdek alanı altındaki tüm piksellerin ortalamasını alır ve merkezi öğe, bu ortalama değerle değiştirilir. Görüntülerdeki tuz-biber gürültüsüne karşı oldukça etkilidir.

Önceki filtrelerde merkezi öğe, görüntüdeki piksel değeri veya yeni bir değer olabilecek yeni hesaplanan bir değerdir. Ancak medyan bulanıklaşdırma, merkezi öğe her zaman görüntüdeki bazı piksel değerleriyle değiştirilir.

Gürültüyü etkili bir şekilde azaltır. Çekirdek boyutu pozitif bir tek sayı olmalıdır.

Medyan Bulanıklık

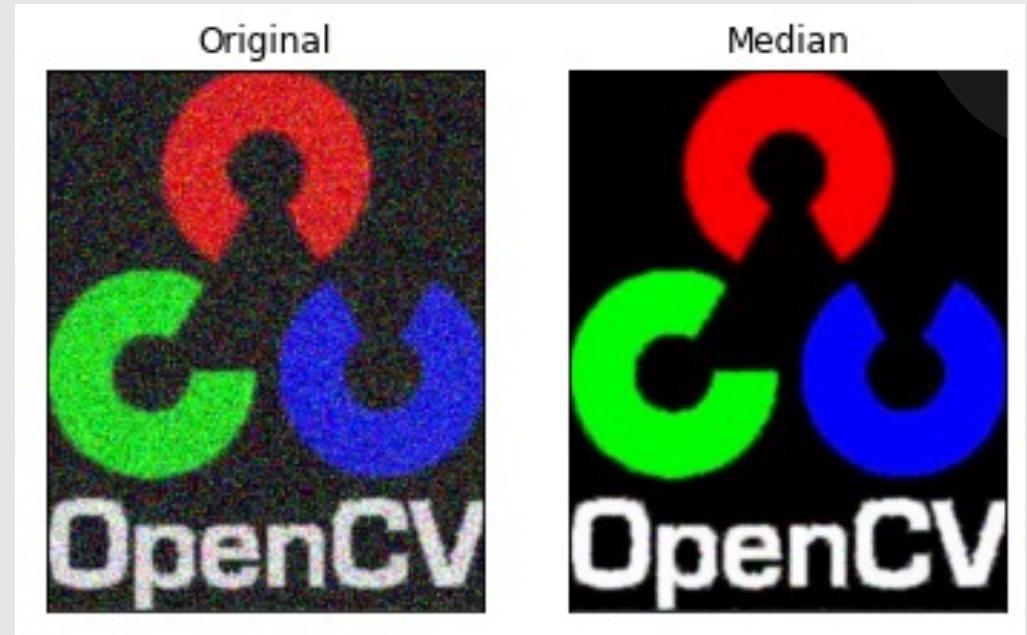
- noisy_img = cv2.imread('noisy_logo.png')
- median_img = cv2.medianBlur(noisy_img, 5)

♦ medianBlur()

```
void cv::medianBlur ( InputArray    src,  
                      OutputArray   dst,  
                      int          ksize  
                    )
```

Python:

```
cv.medianBlur( src, ksize[, dst] ) -> dst
```



İkili Filtreleme

[cv2.bilateralFilter\(\)](#), kenarları keskin tutarken gürültü gidermede oldukça etkilidir. Ancak işlem diğer filtrelere göre daha yavaştır.

Gauss filtresinin piksel etrafındaki bir komşuluğu alır ve gauss ağırlıklı ortalamasını bulur. Bu gauss filtresinde,filtreleme sırasında yakındaki pikseller dikkate alınır.

Piksellerin neredeyse aynı yoğunluğa sahip olup olmadığını dikkate almaz. Pikselin kenar piksel olup olmadığına bakmaz. Böylece, kenarları da bulanıklaştırır.

İkili Filtreleme

♦ bilateralFilter()

```
void cv::bilateralFilter ( InputArray src,  
                          OutputArray dst,  
                          int d,  
                          double sigmaColor,  
                          double sigmaSpace,  
                          int borderType = BORDER_DEFAULT  
)
```

Python:

```
cv.bilateralFilter( src, d, sigmaColor, sigmaSpace[, dst[, borderType]] ) -> dst
```

İkili filtre uzayda bir gauss filtresi alır, ancak piksel farkının bir fonksiyonu olan bir tane daha gauss filtresi alır.

Alanın Gauss işlevi, bulanıklaştırma için yalnızca yakındaki piksellerin dikkate alınmasını sağlarken, yoğunluk farkının gauss işlevi, bulanıklaştırma için yalnızca merkezi piksele benzer yoğunluğa sahip piksellerin dikkate alınmasını sağlar.

Bu nedenle, kenarlardaki pikseller büyük yoğunluk değişimine sahip olacağından kenarları korur.

İkili Filtreleme

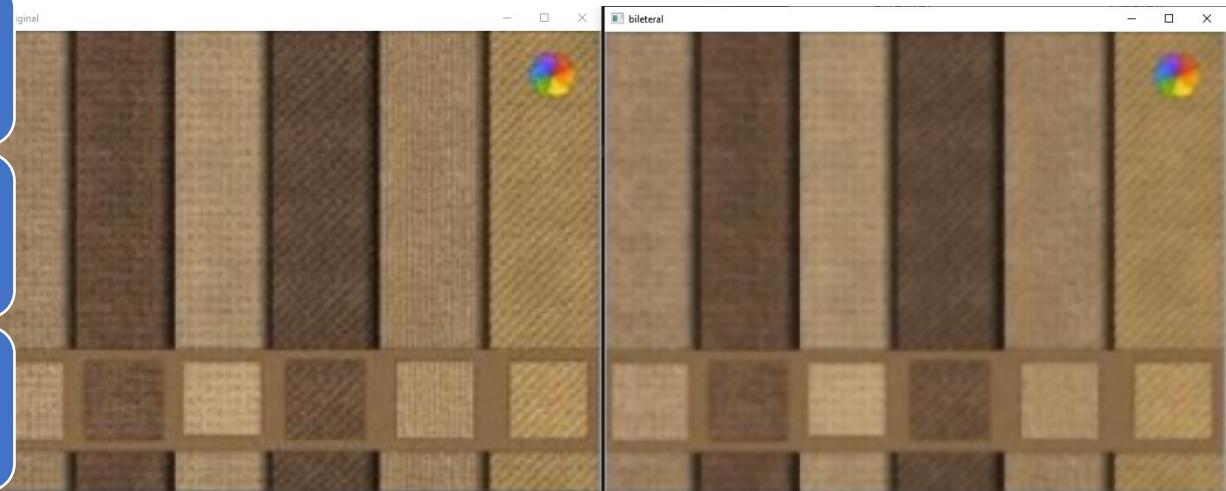
```
bi_img = cv2.imread('bileteral.png')
bi.blur_img = cv2.bilateralFilter(bi_img, 9, 75, 75)
```

d: Filtreleme sırasında kullanılan her piksel komşuluğunun çapı. Pozitif değilse, sigmaSpace'ten hesaplanır.

sigmaColor: Renk uzayında sigmayı filtersi. Parametrenin daha büyük bir değeri, piksel komşuluğu içindeki daha uzaktaki renklerin birbirine karışacağı ve yarı eşit renkli daha geniş alanlara yol açacağı anlamına gelir.

sigmaSpace: Sigma'yı koordinat uzayında filtreleme. Parametrenin daha büyük bir değeri, renkleri yeterince yakın olduğu sürece daha uzaktaki piksellerin birbirini etkileyeceği anlamına gelir.

d>0 olduğunda, sigmaSpace'ten bağımsız olarak komşuluk boyutunu belirtir. Aksi takdirde d, sigmaSpace ile orantılıdır.



KAYNAKLAR

- https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_filtering/py_filtering.html#filtering
- <https://theailearner.com/tag/cv2-getgaussiankernel/>
- https://docs.opencv.org/3.4/d4/d86/group__imgproc__filter.html#ga9d7064d478c95d60003cf839430737ed