



## GÖRÜNTÜ İŞLEME YARDIMCI NOTLARI -2023-



### **Aritmetik ve Mantıksal işlemler**

#### **İki resim ve maske arasında mantıksal işlemler**

İki resim seçelim ve bir de siyah-beyaz maske oluşturalım.

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
x=cv2.imread('balonlar2.jpg')
y=cv2.imread('anahtar.jpg')

print("x boyutu:"+str(x.shape))
print("y boyutu:"+str(y.shape))
#İki resmi aynı boyutlara getirmeliyiz.
y=cv2.resize(y,(226,223))
print("y boyutu:"+str(y.shape))
```

#### **#Bir maske oluşturalım.**

```
maske=np.zeros((223,226,3), np.uint8)
maske[50:150,50:150]=255
```

#### **#MANTIKSAL İŞLEMLER**

```
xANDmaske=cv2.bitwise_and(x,maske)
xORmaske=cv2.bitwise_or(x,maske)
xANDy=cv2.bitwise_and(x,y)
xORy=cv2.bitwise_or(x,y)
x_NOT=cv2.bitwise_not(x)
y_NOT=cv2.bitwise_not(y)
maske_NOT=cv2.bitwise_not(maske)
```

```
plt.figure(1)
plt.subplot(331)
plt.imshow(cv2.cvtColor(x, cv2.COLOR_BGR2RGB)),plt.title('x')
plt.subplot(332)
plt.imshow(cv2.cvtColor(y, cv2.COLOR_BGR2RGB)),plt.title('y')
plt.subplot(333)
plt.imshow(cv2.cvtColor(xANDmaske, cv2.COLOR_BGR2RGB)),plt.title('xANDmaske')
plt.subplot(334)
plt.imshow(cv2.cvtColor(xORmaske, cv2.COLOR_BGR2RGB)),plt.title('xORmaske')
plt.subplot(335)
plt.imshow(cv2.cvtColor(xANDy, cv2.COLOR_BGR2RGB)),plt.title('xANDy')
plt.subplot(336)
```

Matplotlib kütüphanesi resimleri RGB formatında görürken **Opencv** kütüphanesi BGR formatında görür.

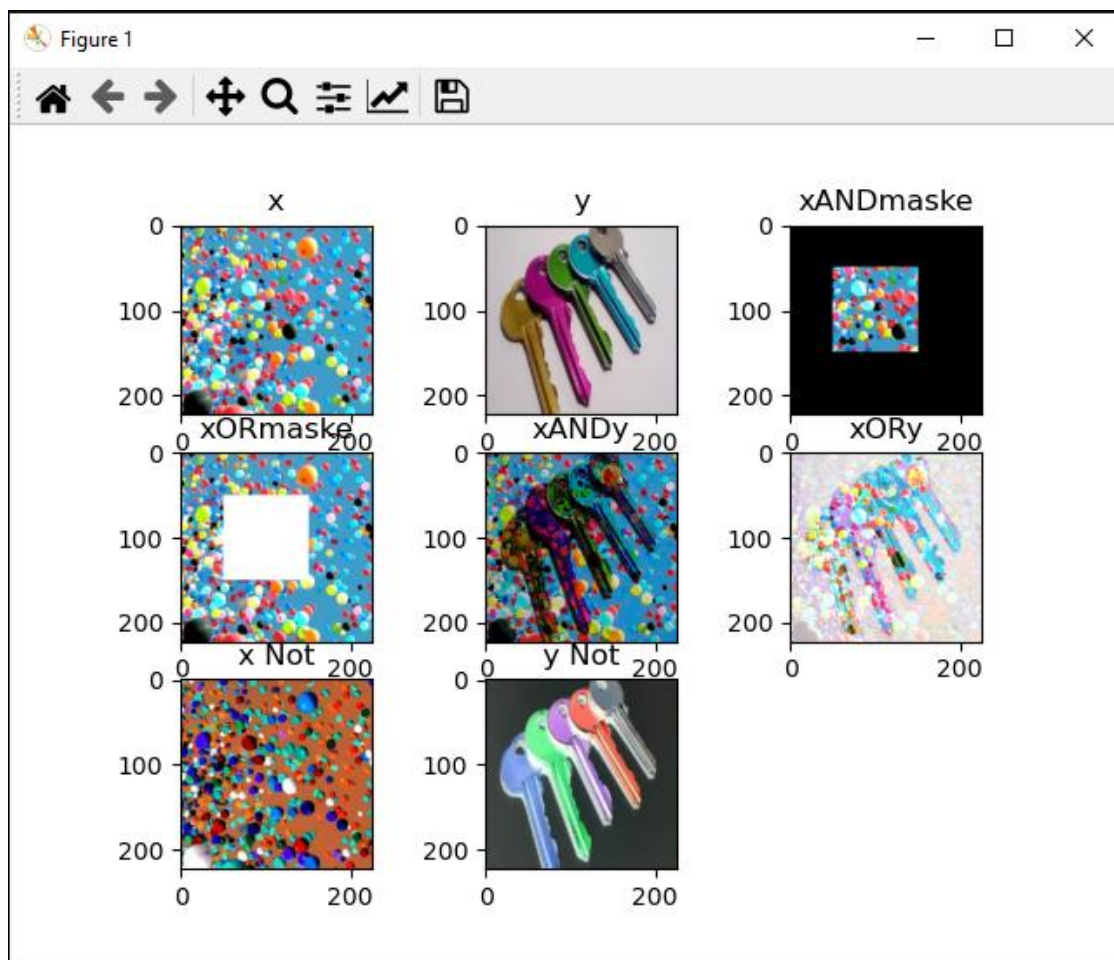
Resimleri cv2 yani Opencv ile okuduğumuz için **matplotlib** ile görüntüleyeceksek aşağıdaki fonksiyon parametrelerini kullanmalıyız.

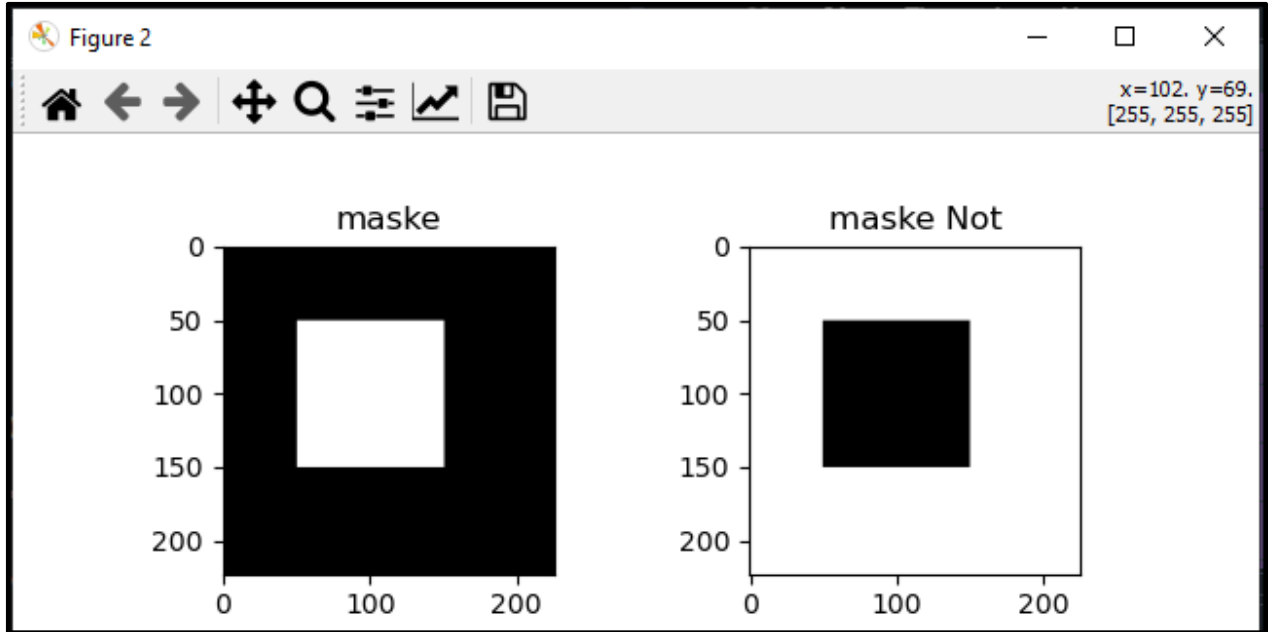
```
plt.imshow(cv2.cvtColor(x, cv2.COLOR_BGR2RGB))
```

```
plt.imshow(cv2.cvtColor(xORy, cv2.COLOR_BGR2RGB)),plt.title('xORy')
plt.subplot(337)
plt.imshow(cv2.cvtColor(x_NOT, cv2.COLOR_BGR2RGB)),plt.title('x Not')
plt.subplot(338)
plt.imshow(cv2.cvtColor(y_NOT, cv2.COLOR_BGR2RGB)),plt.title('y Not')
```

**#İkinci figür penceresini açıyoruz.**

```
plt.figure(2)
plt.subplot(221)
plt.imshow(cv2.cvtColor(maske, cv2.COLOR_BGR2RGB)),plt.title('maske')
plt.subplot(222)
plt.imshow(cv2.cvtColor(maske_NOT, cv2.COLOR_BGR2RGB)),plt.title('maske Not')
```





İki resim arasında aritmetik işlemler

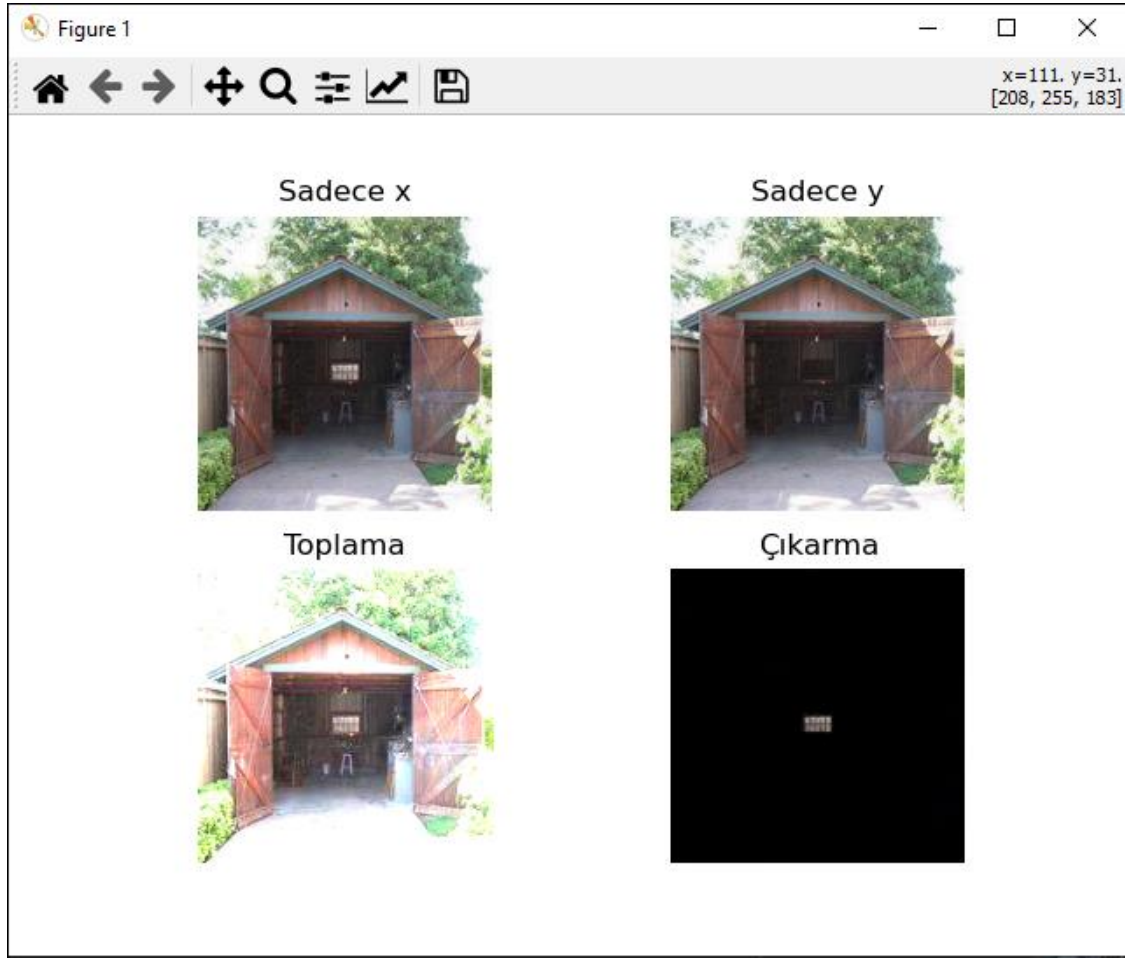
(Toplama ve Çıkartma)

```

import cv2
import numpy as np
from matplotlib import pyplot as plt
x=cv2.imread('headquarters.jpg')
y=cv2.imread('headquarters-Defect.jpg')
##Toplama-Çıkarma işlemi
xADDy=cv2.add(x,y)
xSUBTRACTy=cv2.subtract(x,y)
yanyana=cv2.hconcat([xADDy,xSUBTRACTy])

plt.figure(1)
plt.subplot(221)
plt.imshow(cv2.cvtColor(x, cv2.COLOR_BGR2RGB)),plt.title('Sadece x'), plt.axis('off')
plt.subplot(222)
plt.imshow(cv2.cvtColor(y, cv2.COLOR_BGR2RGB)),plt.title('Sadece y'), plt.axis('off')
plt.subplot(223)
plt.imshow(cv2.cvtColor(xADDy, cv2.COLOR_BGR2RGB)),plt.title('Toplama'), plt.axis('off')
plt.subplot(224)
plt.imshow(cv2.cvtColor(xSUBTRACTy, cv2.COLOR_BGR2RGB)),plt.title('Çıkarma'), plt.axis('off')

```



### Ağırlıklı toplama

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
x=cv2.imread('balon.jpg')
y=cv2.imread('anahtar.jpg')

print("x boyutu:"+str(x.shape))
print("y boyutu:"+str(y.shape))
#iki resmi aynı boyutlara getirmeliyiz.
y=cv2.resize(y,(225,225))
print("y boyutu:"+str(y.shape))

#Toplama işlemi
xADDy=cv2.add(x,y)
#Ağırlıklı toplama işlemi
xWADDy=cv2.addWeighted(x,0.2, y, 0.8,0)

plt.figure(1)
```

```
plt.subplot(141)
plt.imshow(cv2.cvtColor(x, cv2.COLOR_BGR2RGB)),plt.title('x'),plt.axis("off")
plt.subplot(142)
plt.imshow(cv2.cvtColor(y, cv2.COLOR_BGR2RGB)),plt.title('y'),plt.axis("off")
plt.subplot(143)
plt.imshow(cv2.cvtColor(xADDy, cv2.COLOR_BGR2RGB)),plt.title('xADDY'), plt.axis("off")
plt.subplot(144)
plt.imshow(cv2.cvtColor(xWADDy, cv2.COLOR_BGR2RGB)),plt.title('xWADDY'),plt.axis("off")
```



## Resimleri ikilik görüntüye (Siyah/Beyaz) çevirme, Eşikleme

Bir görüntünün ikilik hali aslında siyah/beyaz görüntüsüdür. Görüntünün piksellerinin hangisinin siyah, hangisinin beyaz olacağına eşik değeri ile karar verilir.

OpenCV'nin bir özelliği olarak: Eşik değerden büyük ya da küçük olanlar istenen gri seviye değerinde de bırakılabilir.

### Basit Eşikleme

Her piksel için aynı eşik değeri uygulanır. Piksel değeri eşikten küçükse 0 olarak ayarlanır, aksi takdirde maksimum bir değere ayarlanır. Eşiklemeyi uygulamak için `cv.threshold()` işlevi kullanılır. İlk bağımsız değişken, gri tonlamalı bir görüntü olması gereken kaynak görüntüdür. İkinci bağımsız değişken, piksel değerlerini sınıflandırmak için kullanılan eşik değeridir. Üçüncü bağımsız değişken, eşik değeri aşan piksel değerlerine atanan maksimum değerdir. OpenCV, fonksiyonun dördüncü parametresi tarafından verilen farklı eşikleme türleri sağlar. Yukarıda açıklandığı gibi temel eşikleme `cv.THRESH_BINARY` türü kullanılarak yapılır. Tüm basit eşikleme türleri şunlardır:

- `cv.THRESH_BINARY`
- `cv.THRESH_BINARY_INV`
- `cv.THRESH_TRUNC`
- `cv.THRESH_TOZERO`
- `cv.THRESH_TOZERO_INV`

```
img = cv.imread('anahtar.jpg',0)
ret,thresh1 = cv.threshold(img,127,255,cv.THRESH_BINARY)
```

Okunan resim

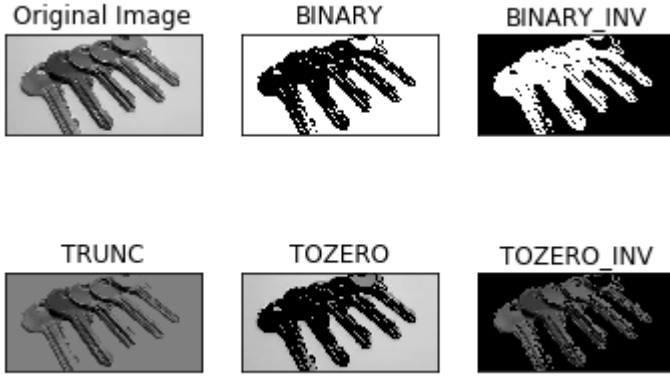
Eşik Değer

ED'den büyük olanlar

Enumerator	
THRESH_BINARY Python: cv.THRESH_BINARY	$dst(x, y) = \begin{cases} \text{maxval} & \text{if } src(x, y) > \text{thresh} \\ 0 & \text{otherwise} \end{cases}$
THRESH_BINARY_INV Python: cv.THRESH_BINARY_INV	$dst(x, y) = \begin{cases} 0 & \text{if } src(x, y) > \text{thresh} \\ \text{maxval} & \text{otherwise} \end{cases}$
THRESH_TRUNC Python: cv.THRESH_TRUNC	$dst(x, y) = \begin{cases} \text{threshold} & \text{if } src(x, y) > \text{thresh} \\ src(x, y) & \text{otherwise} \end{cases}$
THRESH_TOZERO Python: cv.THRESH_TOZERO	$dst(x, y) = \begin{cases} src(x, y) & \text{if } src(x, y) > \text{thresh} \\ 0 & \text{otherwise} \end{cases}$
THRESH_TOZERO_INV Python: cv.THRESH_TOZERO_INV	$dst(x, y) = \begin{cases} 0 & \text{if } src(x, y) > \text{thresh} \\ src(x, y) & \text{otherwise} \end{cases}$
THRESH_MASK Python: cv.THRESH_MASK	
THRESH_OTSU Python: cv.THRESH_OTSU	flag, use Otsu algorithm to choose the optimal threshold value
THRESH_TRIANGLE Python: cv.THRESH_TRIANGLE	flag, use Triangle algorithm to choose the optimal threshold value

## Örnek Eşiklemeler

```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
img = cv.imread('anahtar.jpg',0)
ret,thresh1 = cv.threshold(img,127,255,cv.THRESH_BINARY)
ret,thresh2 = cv.threshold(img,127,255,cv.THRESH_BINARY_INV)
ret,thresh3 = cv.threshold(img,127,255,cv.THRESH_TRUNC)
ret,thresh4 = cv.threshold(img,127,255,cv.THRESH_TOZERO)
ret,thresh5 = cv.threshold(img,127,255,cv.THRESH_TOZERO_INV)
titles = ['Original Image','BINARY','BINARY_INV','TRUNC','TOZERO','TOZERO_INV']
images = [img, thresh1, thresh2, thresh3, thresh4, thresh5]
for i in range(6):
    plt.subplot(2,3,i+1),plt.imshow(images[i])
    plt.title(titles[i])
    plt.xticks([],plt.yticks([]))
plt.show()
```



## Adaptif Eşikleme

Bu teknik, görüntüdeki farklı bölgelerdeki ışık değişimlerini dikkate alarak yerel olarak eşikleme yapmayı mümkün kılar. Bu yöntem, farklı aydınlatma koşullarında veya kontrastın düşük olduğu durumlarda kullanışlıdır.

Python'da, OpenCV kütüphanesi, görüntü işleme uygulamaları için sıkça kullanılır. Adaptif eşikleme işlemi, bu kütüphane içinde bulunan bir fonksiyon olan `cv2.adaptiveThreshold` ile gerçekleştirilebilir. Bu fonksiyon, belirli bir pencere boyutunda (blok boyutu) ve her piksel için hesaplanan eşik değerini (C değeri) kullanarak adaptif eşikleme işlemi gerçekleştirir.

Adaptif eşikleme işlemi şu adımlarla gerçekleştirilir:

- Görüntü gri seviyeye dönüştürülür.
- Her pikselin eşik değeri, pikselin bulunduğu bölgeye göre (blok boyutu) hesaplanır.
- Eşik değeri, pikselin bulunduğu bölgeye (blok) bağlı olarak ayarlanır.



- Belirlenen eşik değerine göre piksel değerleri siyah veya beyaz olarak değiştirilir.

Bu yöntem, görüntülerdeki detayları koruyarak ve farklı ışık koşullarında daha tutarlı bir eşikleme sağlayarak gürültüyü azaltmaya yardımcı olur. Adaptif eşikleme, özellikle tarama veya belge tanıma gibi uygulamalarda kullanılır. Aşağıdaki kodlarla gerçekleştirilir.

**cv2.adaptiveThreshold(goruntu, maksDeger, adaptifMetot, esiklemeTipi, blokBoyutu, C)**

Burada:

- **goruntu**: Görüntünün kaynak matrisi.
- **maksDeger**: Maksimum eşik değeri. Bu değer, eşikleme sonrasında beyaz renkteki piksellerin değerini belirler.
- **adaptifMetot**: Kullanılacak adaptif eşikleme yöntemini belirler. İki seçenek vardır: cv2.ADAPTIVE\_THRESH\_MEAN\_C veya cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C. İlk seçenek, blok içindeki piksellerin ortalamasını alırken, ikincisi Gauss fonksiyonunu kullanır.
- **esiklemeTipi**: Eşikleme tipini belirler. cv2.THRESH\_BINARY veya cv2.THRESH\_BINARY\_INV değerlerinden biri olabilir.
- **blokBoyutu**: Eşik değerinin hesaplanacağı blok boyutunu belirler. Bu, bir sayı çifti olarak belirtilir ve genellikle tek sayı olmalıdır.
- **C**: Hesaplanan eşik değerine eklenen sabit bir değerdir.

*import cv2*

*import numpy as np*

*from matplotlib import pyplot as plt*

*x=cv2.imread("balon.jpg",0) # gri seviye olarak okundu.*

*ret,thresh1 = cv2.threshold(x,108,255,cv2.THRESH\_BINARY)*

*ret2,thresh2 = cv2.threshold(x,108,255,cv2.THRESH\_BINARY\_INV)#eşik değer olarak kendimiz değer atadık.*

*cv2.imshow('th1',thresh1)*

*print(ret)*

*ret3,thresh3 = cv2.threshold(x,108,255,cv2.THRESH\_BINARY+cv2.THRESH\_OTSU) #OTSU*

*yöntemine göre eşik değer kullanma*

*cv2.imshow('th3',thresh3)*

*print(ret3)*

*th2 = cv2.adaptiveThreshold(x,255,cv2.ADAPTIVE\_THRESH\_MEAN\_C, cv2.THRESH\_BINARY,11,2) #Adaptif eşikleme (mean) C=2, blok boyutu 11x11*

*th3 = cv2.adaptiveThreshold(x,255,cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C,cv2.THRESH\_BINARY,11,2) #Adaptif eşikleme (Gaussian)*

*cv2.imshow('Mean',th2)*

*cv2.imshow('Gaussian',th3)*

*cv2.waitKey(0)*

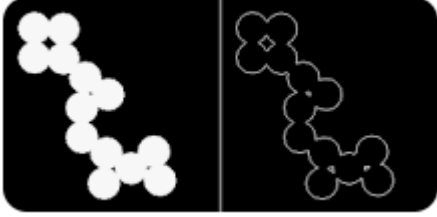
*cv2.destroyAllWindows()*





## Morfolojik İşlemler:

Biyolojinin canlıların şekil ve yapıları ile ilgilenen dalına morfoloji (biçim bilim) adı verilmektedir. Morfoloji, görüntüleri şekillere dayalı olarak işleyen geniş bir görüntü işleme işlemleri kümesidir. Morfolojik işlemler, bir giriş görüntüsüne bir yapılandırma ögesi uygulayarak aynı boyutta bir çıkış görüntüsü oluşturur.



Matematiksel morfoloji ise temel küme işlemlerine dayanan, imgedeki sınırlar (borders), iskelet (skeleton) gibi yapıların tanımlanması ve çıkartılması, gürültü giderimi, bölütleme gibi uygulamalar için gerekli bir araçtır. İmge işlemede genellikle, morfolojik filtreleme

- inceltme (thinning).
- budama (prunning) gibi ön/son işlem olarak da sıkça kullanılırlar.

Gri tonlu imgeler üzerinde de yapılabileceği gibi, genellikle ikilik (binary) imgeler üzerinde yapılan işlemlerdir.

- Yerel işlemler kullanarak bir görüntüdeki nesnelere şeklini değiştirmek için kullanılır.
- Segmentasyon sonrası işlemlerde istenmeyen etkileri kaldırmak için kullanılabilir.
- Küçük nesnelere kaldırır (gürültü olduğu varsayılır)
- Büyük nesnelere kenarlarını yumuşatır.

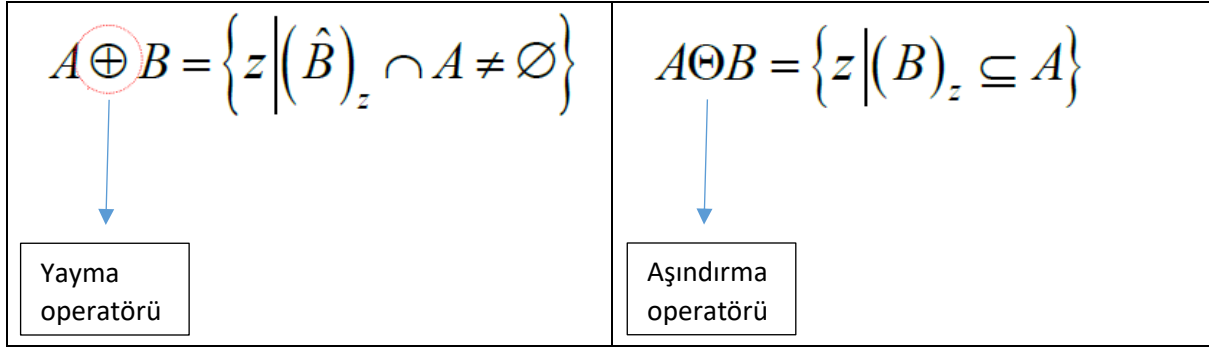
## Morfolojik Yayma ve Aşındırma

En temel morfolojik işlemler yayma ve aşındırma. Yayma bir görüntüdeki nesnelere sınırlarına piksel eklerken, erozyon nesne sınırlarındaki pikselleri kaldırır.

Bir görüntüdeki nesnelere eklenen veya çıkarılan piksel sayısı, görüntüyü işlemek için kullanılan yapılandırma elemanı boyutuna ve şekline bağlıdır.

Morfolojik operatörlerin iki girişi vardır:

1. Yayılacak ya da aşındırılacak (Kaynak) imge,
2. Yayma ya da aşındırma işleminin şeklini belirleyen yapılandırma elemanı (structure element).



Yapı elemanları yayma işlemlerinin nasıl yapılacağını belirlemektedir.

Örnek:

Yapı elemanı dizilimi	Yapı elemanı tipi	Aşağıdaki kodlar, yapılandırma elemanları oluşturmamızı sağlar
<pre>0 0 1 0 0 0 0 1 0 0 1 1 1 1 1 0 0 1 0 0 0 0 1 0 0</pre>	Cross	<pre>cv2.getStructuringElement(cv2.MORPH_CROSS,(5,5))</pre>
<pre>0 0 1 1 1 0 0 0 1 1 1 1 1 0 1 0 1 1 1 1 1 0 0 0 1 1 1 0 0</pre>	Elips	<pre>cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(7,7))</pre>
<pre>1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1</pre>	Rect	<pre>cv2.getStructuringElement(cv2.MORPH_RECT,(4,4))</pre>

Morfolojik Aşındırma örnek kodları:

```
import cv2
import numpy as np
img = cv2.imread('j.png',0)
#kernel = np.ones((5,5),np.uint8)
kernel=cv2.getStructuringElement(cv2.MORPH_RECT,(5,5))
erozyon = cv2.erode(img,kernel,iterations = 1)
tum=cv2.hconcat([img,erozyon])
cv2.imshow('erozyon',tum)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Yapılandırma elemanı oluşturma

Aşındırma işlemi

Çıktı:



Orijinal

Aşındırılmış

Morfolojik Yayma işlemi örnek kodları:

```
import cv2
import numpy as np
img = cv2.imread('j.png',0)
#kernel = np.ones((5,5),np.uint8)
kernel=cv2.getStructuringElement(cv2.MORPH_RECT,(5,5))
yayma = cv2.dilate(img,kernel,iterations = 1)
tum=cv2.hconcat([img,yayma])
cv2.imshow('erozyon',tum)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Yapılandırma  
elemanı oluşturma

Yayma işlemi

Çıktı:



Orijinal

Yayılmış

Morfolojik Açma ve Kapama İşlemleri

### Morfolojik Açma

Önce A'yı B yapısal elemanı ile **aşındırma** işlemine tabi tut. Sonra çıkan sonucu aynı yapısal eleman ile **yayma** işlemine tabi tut.

$$A \circ B = (A \ominus B) \oplus B.$$

### Morfolojik Kapama

Önce A'yı B yapısal elemanı ile **yayma** işlemine tabi tut. Sonra çıkan sonucu aynı yapısal eleman ile **aşındırma** işlemine tabi tut.

$$A \bullet B = (A \oplus B) \ominus B.$$

**Morfolojik Açma işlemi örnek kodları:**

```
import cv2
import numpy as np
img = cv2.imread('j-opening.png',0)
kernel = cv2.getStructuringElement(cv2.MORPH_RECT,(9,9))
acma = cv2.morphologyEx(img,cv2.MORPH_OPEN,kernel)
tum=cv2.hconcat([img,acma])
cv2.imshow('acma',tum)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Çıktı:



Gürültülü imge

Temizlenmiş İmge

**Morfolojik Kapama işlemi örnek kodları:**

```
import cv2
import numpy as np
img = cv2.imread('j-closing.png',0)
kernel = cv2.getStructuringElement(cv2.MORPH_RECT,(9,9))
acma = cv2.morphologyEx(img,cv2.MORPH_CLOSE,kernel)
tum=cv2.hconcat([img,acma])
cv2.imshow('acma',tum)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



Gürültülü imge

Temizlenmiş İmge

### Morfolojik Gardyan:

İmgelerin sınırlarını bulmak için kullanılabilir.

### Örnek Kod:

```
import cv2
import numpy as np
img = cv2.imread('j.png',0)
kernel = cv2.getStructuringElement(cv2.MORPH_RECT,(3,3))
gradyan = cv2.morphologyEx(img,cv2.MORPH_GRADIENT,kernel)
tum=cv2.hconcat([img,gradyan])
cv2.imshow('Gradyan',tum)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Çıktı:

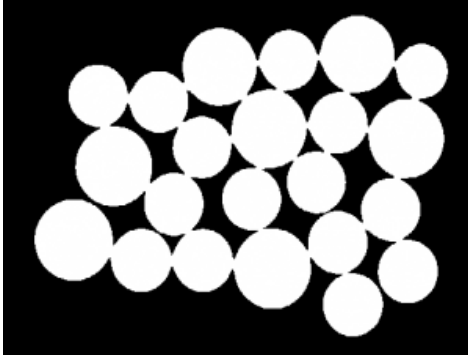


Orjinal imge

Sınırlar tespit

### Örnek Soru:

Bir banttın geçen yumurtaların tepe kamerası ile alınmış görüntüleri aşağıda verilmiştir. Bu yumurtaların görüntü işleme yoluyla sayılabilmesi için gereken işlemleri gerçekleştiriniz.

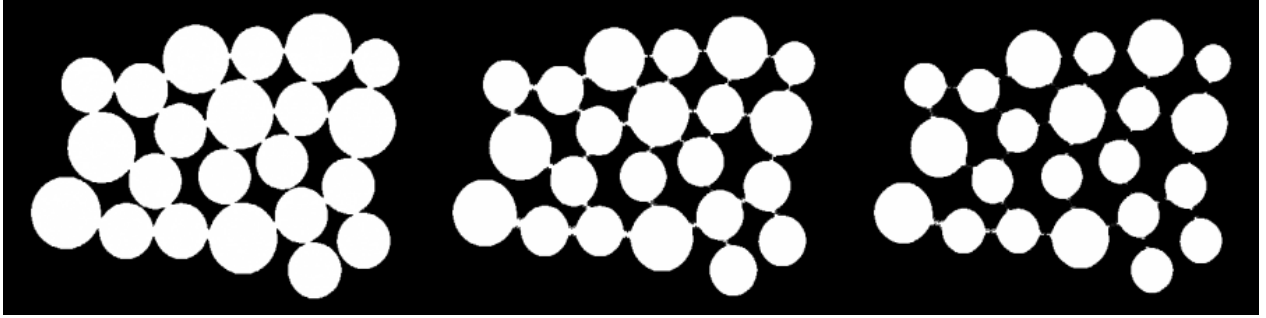


### Kodlar:

```
import cv2
import numpy as np
img = cv2.imread('morfolojiyumurta.png',0)
#kernel = np.ones((5,5),np.uint8)
kernel1=cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(5,5))
kernel2=cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(9,9))
erozyon5_5 = cv2.erode(img,kernel1,iterations = 1)
erozyon9_9 = cv2.erode(img,kernel2,iterations = 1)
tum=cv2.hconcat([img,erozyon5_5,erozyon9_9])
cv2.imshow('Yumurta Ayirma',tum)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



**Sonuç:**



Orijinal İmge

5x5 Elips Yapılandırma  
elemanı ile aşındırma

9x9 Elips Yapılandırma  
elemanı ile aşındırma

## Derste İşlenen Kodlar:

### İki resim arası farkı bulma kodları

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
x=cv2.imread('balonlar2.jpg',0) #Gri seviye olarak okuyoruz.
y=cv2.imread('balonlar3.jpg',0) #Gri seviye olarak okuyoruz.

fark=cv2.subtract(x, y)
ret,thresh1 = cv2.threshold(fark,50,255,cv2.THRESH_BINARY)

plt.imshow(cv2.cvtColor(fark, cv2.COLOR_BGR2RGB)),plt.title('Farklar'),plt.axis('off')
cv2.circle(thresh1,(147,25), 25, (80,150,210),2)
plt.figure(2)
plt.imshow(cv2.cvtColor(thresh1, cv2.COLOR_BGR2RGB)),plt.title('eşiklenmiş'),plt.axis('off')
```

### Aşındırma Örneği:

```
import cv2
from matplotlib import pyplot as plt
img = cv2.imread('j.png',0)
kare=cv2.getStructuringElement(cv2.MORPH_RECT,(5,5))
elips=cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(3,3))

kareyleAsindir = cv2.erode(img,kare,iterations = 1)
elipsleAsindir = cv2.erode(img,elips,iterations = 1)
tum=cv2.hconcat([img,kareyleAsindir,elipsleAsindir])
plt.imshow(cv2.cvtColor(tum, cv2.COLOR_BGR2RGB)),plt.title('Aşındırma'), plt.axis('off')
```

### Adaptif Eşikleme:

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
x=cv2.imread("balon.jpg",0) # gri seviye olarak okundu.
a,b=cv2.threshold(x, 127, 255, cv2.THRESH_BINARY)
adaptif=cv2.adaptiveThreshold(x, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 7, 4)
adaptifMean=cv2.adaptiveThreshold(x, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY, 7, 4)
plt.subplot(131)
plt.imshow(cv2.cvtColor(x, cv2.COLOR_BGR2RGB)),plt.title('Orjinal'),plt.axis('off')
plt.subplot(132)
plt.imshow(cv2.cvtColor(adaptifMean, cv2.COLOR_BGR2RGB)),plt.title('MEAN'),plt.axis('off')
plt.subplot(133)
plt.imshow(cv2.cvtColor(adaptif, cv2.COLOR_BGR2RGB)),plt.title('GAUSSIAN'),plt.axis('off')
```

### Morfolojik Açma:

```
import cv2

from matplotlib import pyplot as plt

img = cv2.imread('j-opening.png',0)

kare=cv2.getStructuringElement(cv2.MORPH_RECT,(7,7))

asindir=cv2.erode(img,kare,iterations=1)

yay=cv2.dilate(asindir,kare,iterations=1)

plt.subplot(131)

plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB)),plt.title('Orjinal'),plt.axis('off')

plt.subplot(132)

plt.imshow(cv2.cvtColor(asindir, cv2.COLOR_BGR2RGB)),plt.title('Aşındırılmış'),plt.axis('off')

plt.subplot(133)

plt.imshow(cv2.cvtColor(yay, cv2.COLOR_BGR2RGB)),plt.title('Morfolojik Açma'),plt.axis('off')

ret, son=cv2.threshold(yay, 200,255, cv2.THRESH_BINARY)

plt.figure(2)

plt.imshow(cv2.cvtColor(son, cv2.COLOR_BGR2RGB)),plt.title('Son Açma'),plt.axis('off')
```