Aritmetik ve Mantıksal işlemler

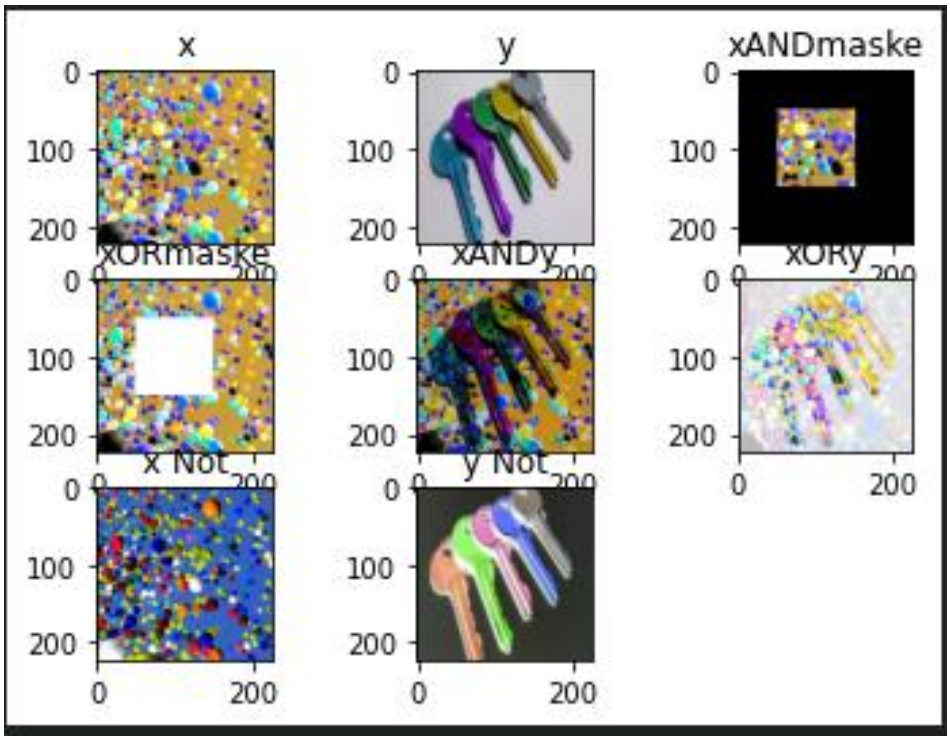
İki resim ve maske arasında mantıksal işlemler

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))
#iki 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(x)
plt.subplot(332)
plt.imshow(y)
plt.subplot(333)
plt.imshow(xANDmaske)
plt.subplot(334)
plt.imshow(xORmaske)
plt.subplot(335)
```

plt.imshow(xANDy)
plt.subplot(336)
plt.imshow(xORy)
plt.subplot(337)
plt.imshow(x_NOT),plt.title('x Not')
plt.subplot(338)
plt.imshow(y_NOT),plt.title('y Not')

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

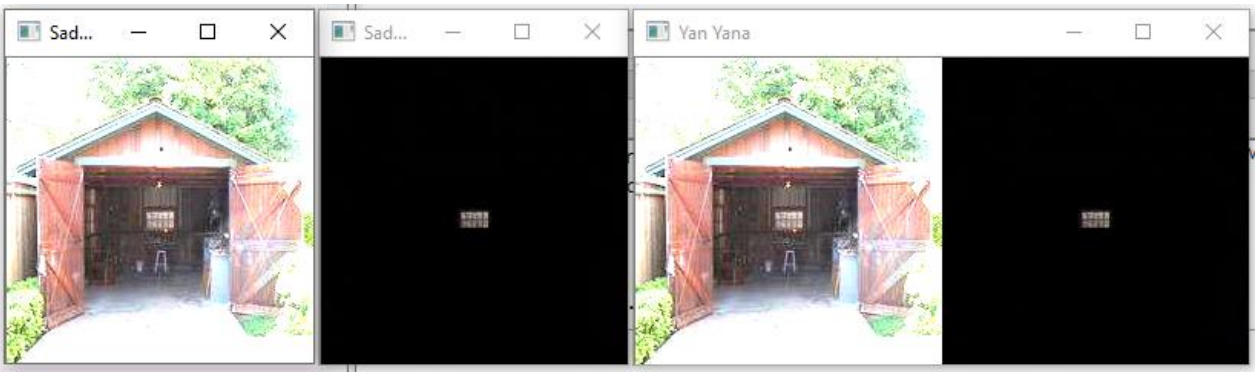


İki resim arasında aritmetik işlemler

(Toplama ve Çıkartma)

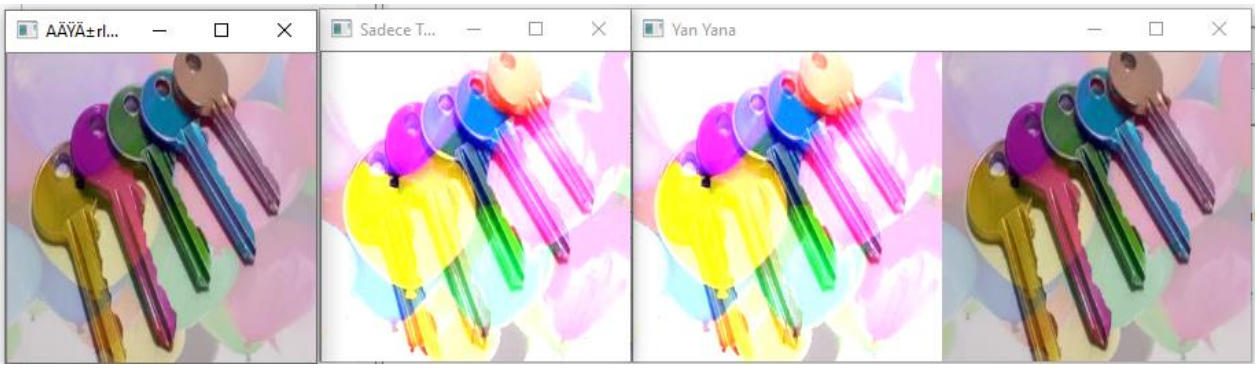
```
import cv2
import numpy as np
from matplotlib import pyplot as plt
x=cv2.imread('headquarters.jpg')
y=cv2.imread('headquarters-Defect.jpg')
##Toplama-Çıkarma işlemi
```

```
xADDy=cv2.add(x,y)
xSUBTRACTy=cv2.subtract(x,y)
toplam=cv2.hconcat([xADDy,xSUBTRACTy])
cv2.imshow('Sadece Toplama',xADDy)
cv2.imshow('Sadece Çıkarma',xSUBTRACTy)
cv2.imshow('Yan Yana',toplam)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



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)
xWADDy=cv2.addWeighted(x,0.2, y, 0.8,0)
toplam=cv2.hconcat([xADDy,xWADDy])


cv2.imshow('Sadece Toplama',xADDy)
cv2.imshow('Ağırlıklı Toplama',xWADDy)
cv2.imshow('Yan Yana',toplam)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



İkilik görüntüye çevirme, Eşikleme

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

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

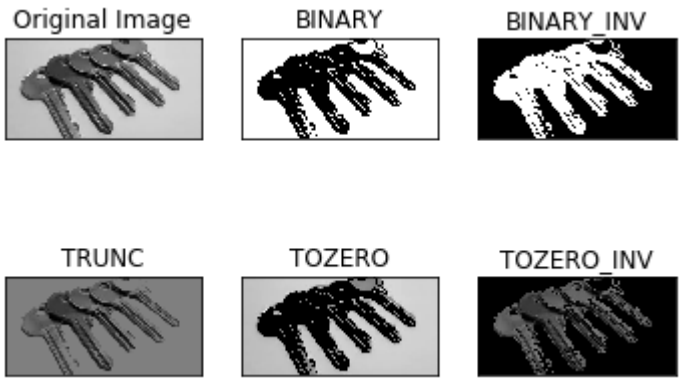
Basit Eşikleme

Burada mesele basittir. 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, eşiği 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

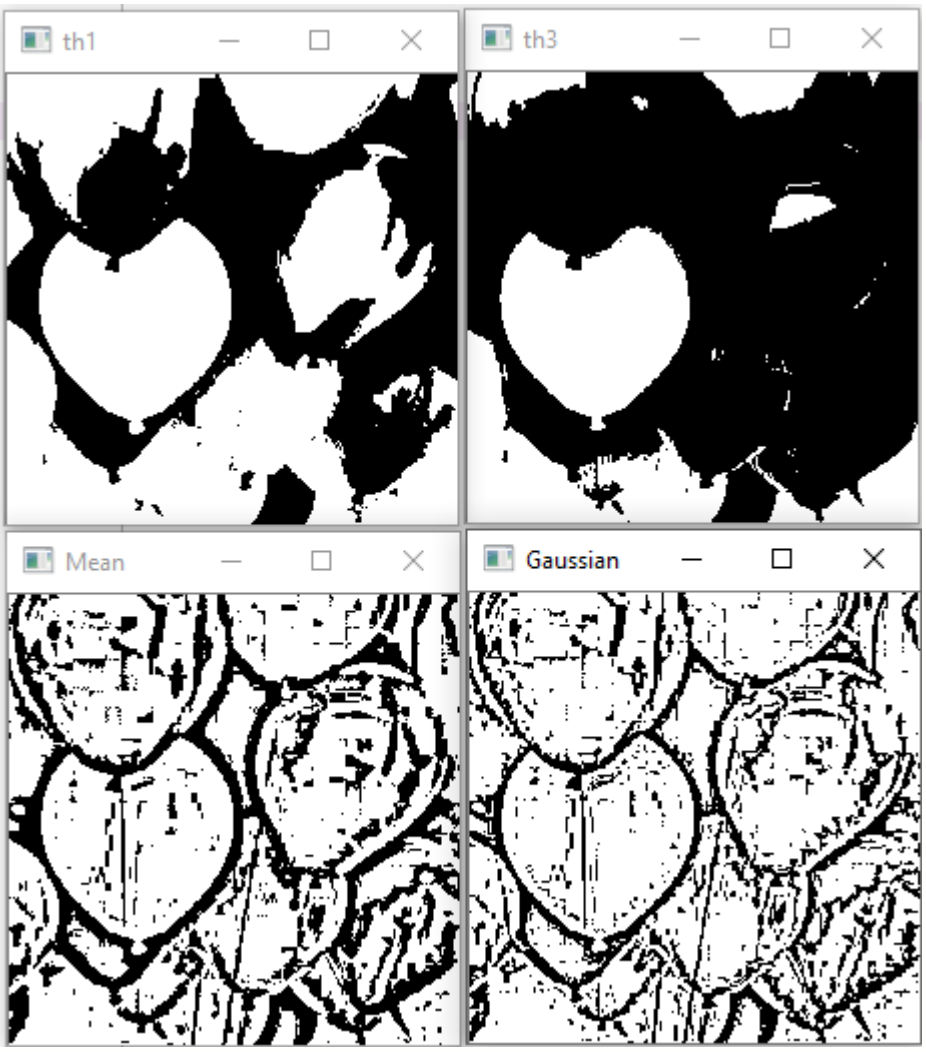
Enumerator	
THRESH_BINARY Python: cv.THRESH_BINARY	$\texttt{dst}(x,y) = \begin{cases} \texttt{maxval} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$
THRESH_BINARY_INV Python: cv.THRESH_BINARY_INV	$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{maxval} & \text{otherwise} \end{cases}$
THRESH_TRUNC Python: cv.THRESH_TRUNC	$\texttt{dst}(x,y) = \begin{cases} \texttt{threshold} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$
THRESH_TOZERO Python: cv.THRESH_TOZERO	$\texttt{dst}(x,y) = \begin{cases} \texttt{src}(x,y) & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$
THRESH_TOZERO_INV Python: cv.THRESH_TOZERO_INV	$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$
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

```
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],'gray',vmin=0,vmax=255)
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])
plt.show()
```



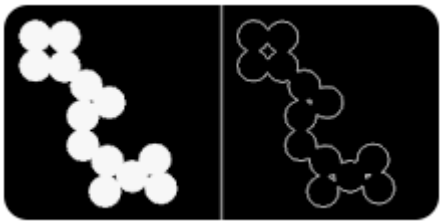
Adaptif Eşikleme

```
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,1
1,2) #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 öğesi 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 nesnelerin şeklini değiştirmek için kullanılır.
- Segmentasyon sonrası işlemlerde istenmeyen etkileri kaldırmak için kullanılabilir.
- Küçük nesneleri kaldırır (gürültü olduğu varsayılır)
- Büyük nesnelerin kenarlarını yumuşatır.

Morfolojik Yayma ve Aşındırma

En temel morfolojik işlemler yayma ve aşındırmadır. Yayma bir görüntüdeki nesnelerin 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).

$A \oplus B = \left\{ z \middle\| \left(\hat{B} \right)_z \cap A \neq \varnothing \right\}$	$A \Theta B = \left\{ z \middle\| \left(B \right)_z \subseteq A \right\}$
Yayma operatörü	Aşındırma operatörü

Yapı elemanları yayma işlemlerinin nasıl yapılacağını belirlemektedir.

Örnek:

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

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 İşlemi

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 İşlemi

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-openning.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:

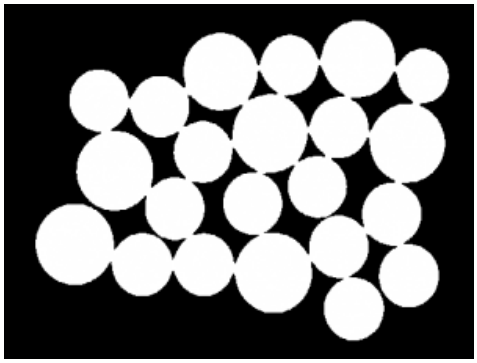
```
import cv2
import numpy as np
img = cv2.imread('j.png',0)
kernel = cv2.getStructuringElement(cv2.MORPH_RECT,(3,3))
gradyan = cv2.morphologyEx(img,cv2.MORPH_GRADIENT,kernel)
tum=cv2.hconcat([img,gradyan])
cv2.imshow('Gradyan',tum)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Çıktı:

Orjinal imge Sınırlar tespit

Örnek Soru:

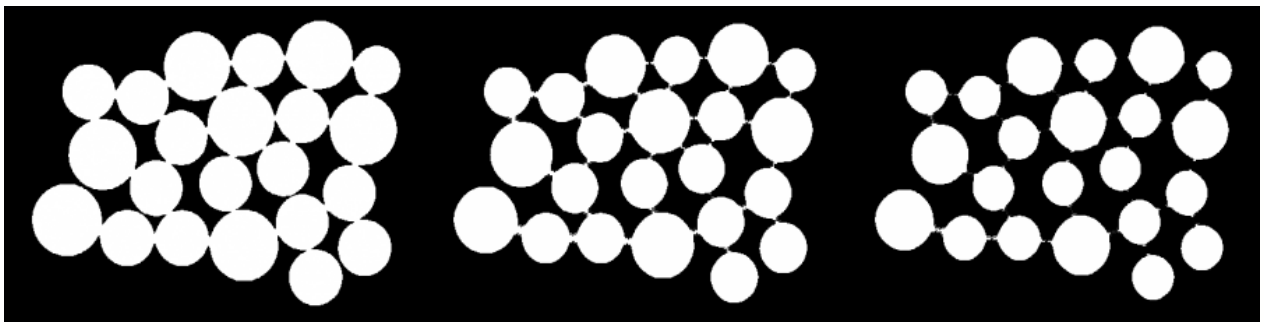
Bir banttan geçen yumurtaların tepe kamerası ile alınmış görüntüleri aşağıda verilmiştir. Bu yumurtaların görüntü işleme yoluyla sayılabilmesi için gereken işlemleri gerçekleştiriniz.



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