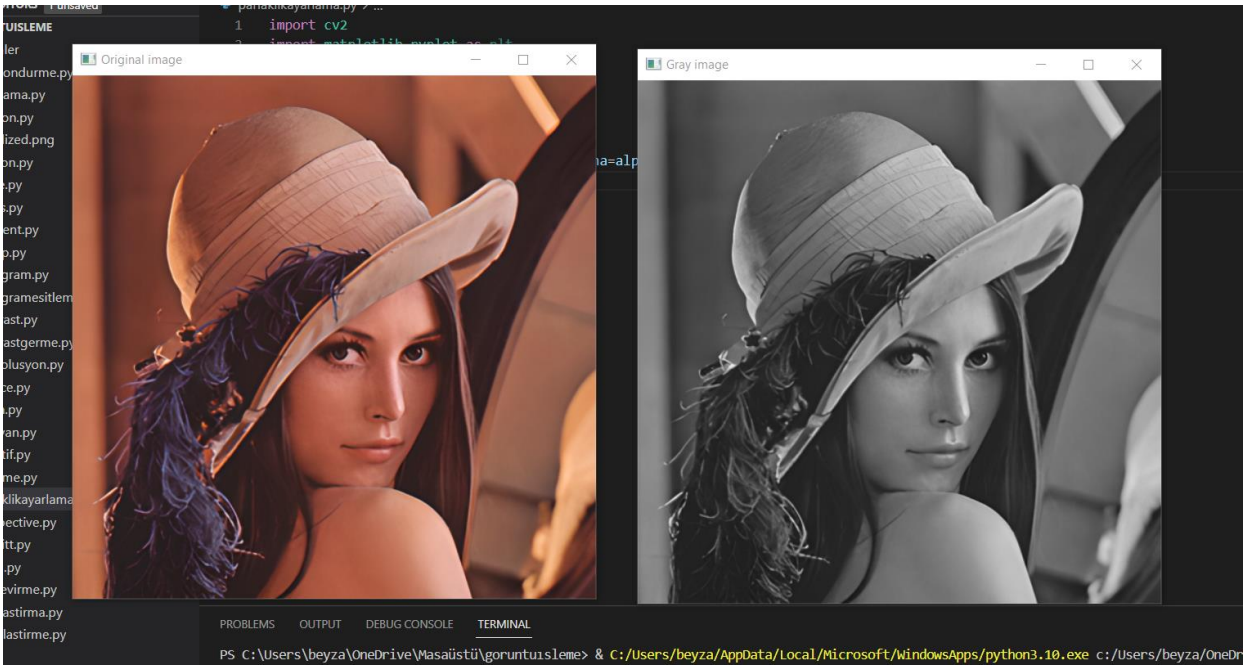
BEYZANUR EKİZ

OPENCV KÜTÜPHANESİ KULLANARAK PYTHON'DA GÖRÜNTÜ İŞLEMLERİ

İçindekiler Tablosu

1. RENKLİ GÖRÜNTÜLERİN GRİ TONLU HALE DÖNÜŞTÜRÜLMESİ

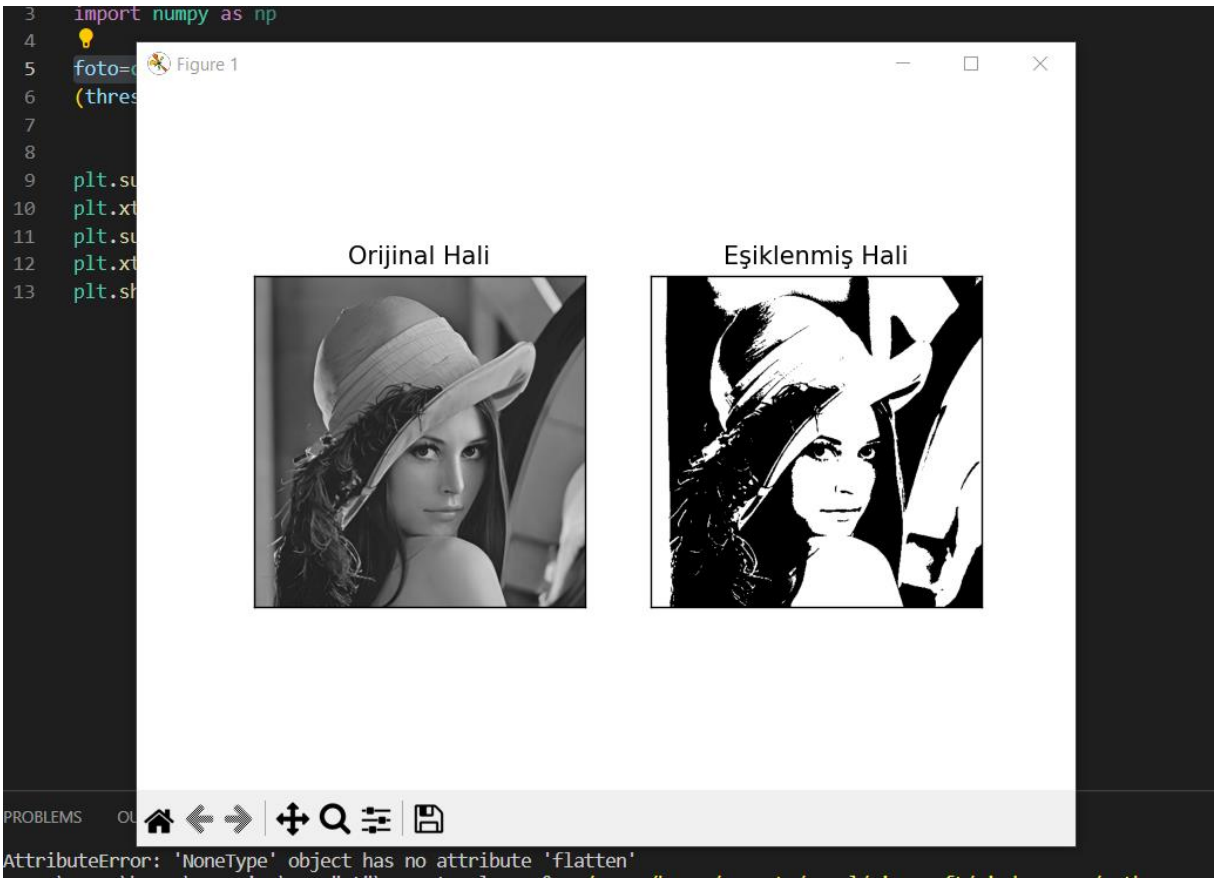
Renkli fotoğrafları gri tona dönüştürmemizi sağlayan filtredir.




```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

image = cv2.imread('resimler/lena.png')
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

cv2.imshow('Original image',image)
cv2.imshow('Gray image', gray)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

2. EŞİKLEME

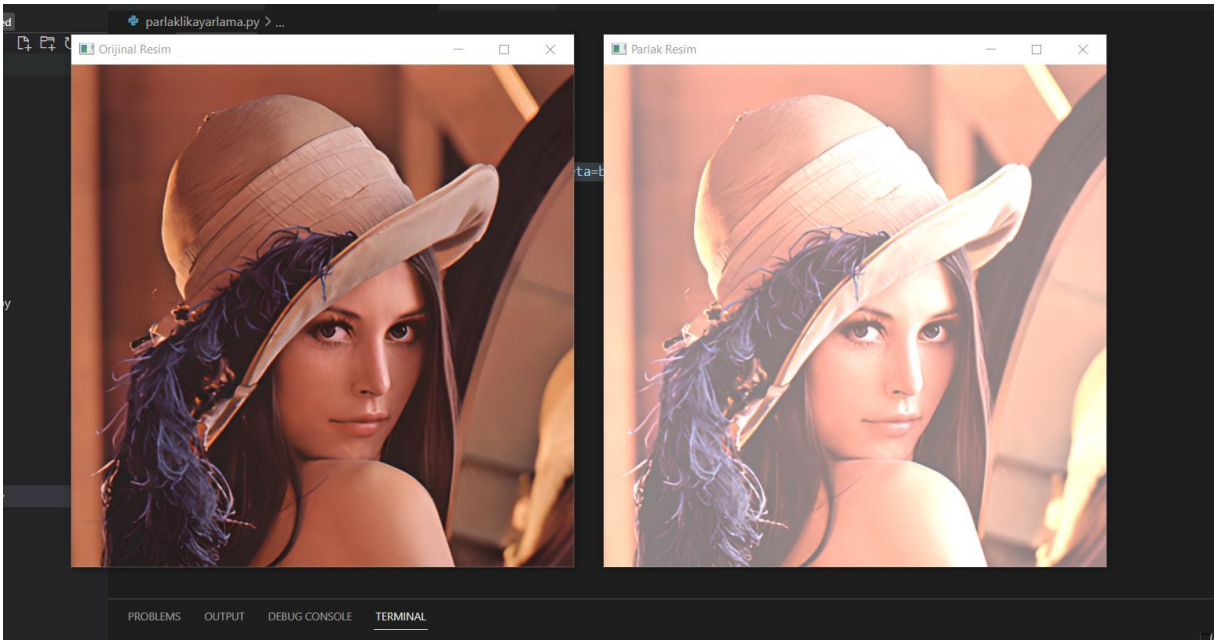
Belirli bir eşik değeri altında olan kısımları 0; üstünde olan kısımları 1 yapmak suretiyle ikili(**binary**) bir görüntü oluşturmaktır.




```
import cv2
import matplotlib.pyplot as plt
import numpy as np

foto=cv2.imread("resimler/grilena.png")
(thresh, blackAndWhiteImage) = cv2.threshold(foto, 100, 255,
cv2.THRESH_BINARY)


plt.subplot(121),plt.imshow(foto),plt.title('Orijinal Hali')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(blackAndWhiteImage),plt.title('Eşiklenmiş Hali')
plt.xticks([]), plt.yticks([])
plt.show()
```

3. PARLAKLIK AYARI

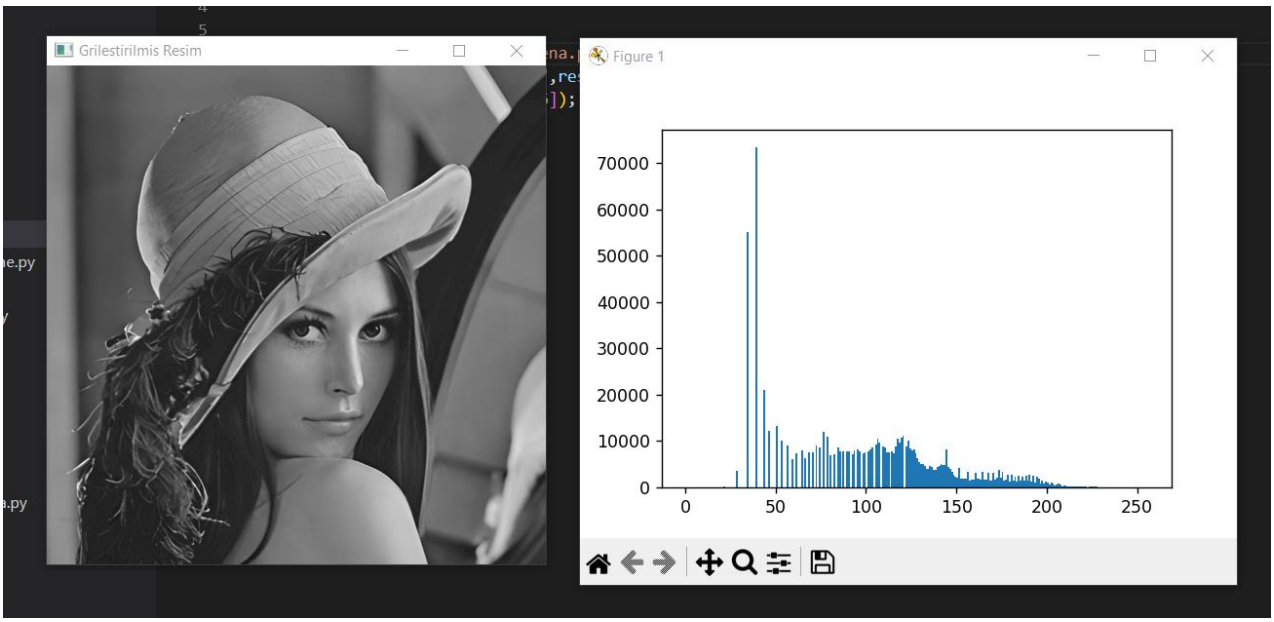
Resimlerin rgb değerlerini belli bir sayıyla toplayarak görüntünün renginin açılmasını sağlar.




```
import cv2
import matplotlib.pyplot as plt
import numpy as np

resim = cv2.imread('resimler/lena.png')
alpha=1
beta=100
adjusted = cv2.convertScaleAbs(resim, alpha=alpha, beta=beta)

cv2.imshow("Orijinal Resim",resim)
cv2.imshow('Parlak Resim', adjusted)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

4. GÖRÜNTÜ HİSTOGRAMININ ELDE EDİLMESİ

Görüntüdeki renk piksellerinin her birinden kaçar tane olduğunu gösterir.




```python
import cv2
import matplotlib.pyplot as plt
import numpy as np


resim= cv2.imread("resimler/grilena.png")
cv2.imshow("Grilestirilmis Resim",resim)
plt.hist(resim.ravel(),256,[0,256]); plt.show()
cv2.waitKey(0)
cv2.destroyAllWindows()
```

5. HİSTOGRAM EŞİTLEME

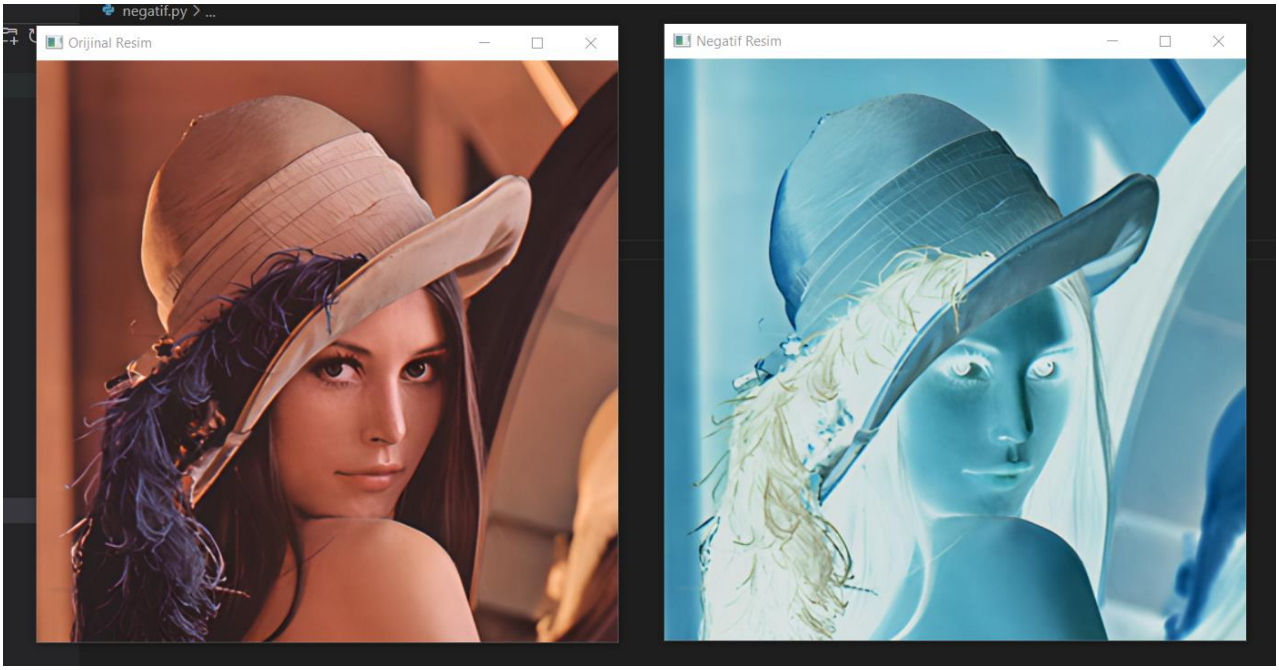
Histogram eşitleme, renk dağılımı bozukluğunu gidermek için kullanılan bir yöntemdir.

```
import cv2
import matplotlib.pyplot as plt
import numpy as np



img = cv2.imread('resimler/histogramesitleme.jpg',0)
row, col = img.shape[:2]


def df(img):
    values = [0]*256
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            values[img[i,j]]+=1
    return values


def cdf(hist):
    cdf = [0] * len(hist)
    cdf[0] = hist[0]
    for i in range(1, len(hist)):
        cdf[i]= cdf[i-1]+hist[i]

    cdf = [ele*255/cdf[-1] for ele in cdf]
    return cdf

def equalize_image(image):
    my_cdf = cdf(df(img))
    import numpy as np
```

```
    image_equalized = np.interp(image, range(0,256), my_cdf)
    return image_equalized

eq = equalize_image(img)
cv2.imshow('equalized', eq)
cv2.imshow('Orijinal görüntü',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

6. NEGATİF GÖRÜNTÜLEME

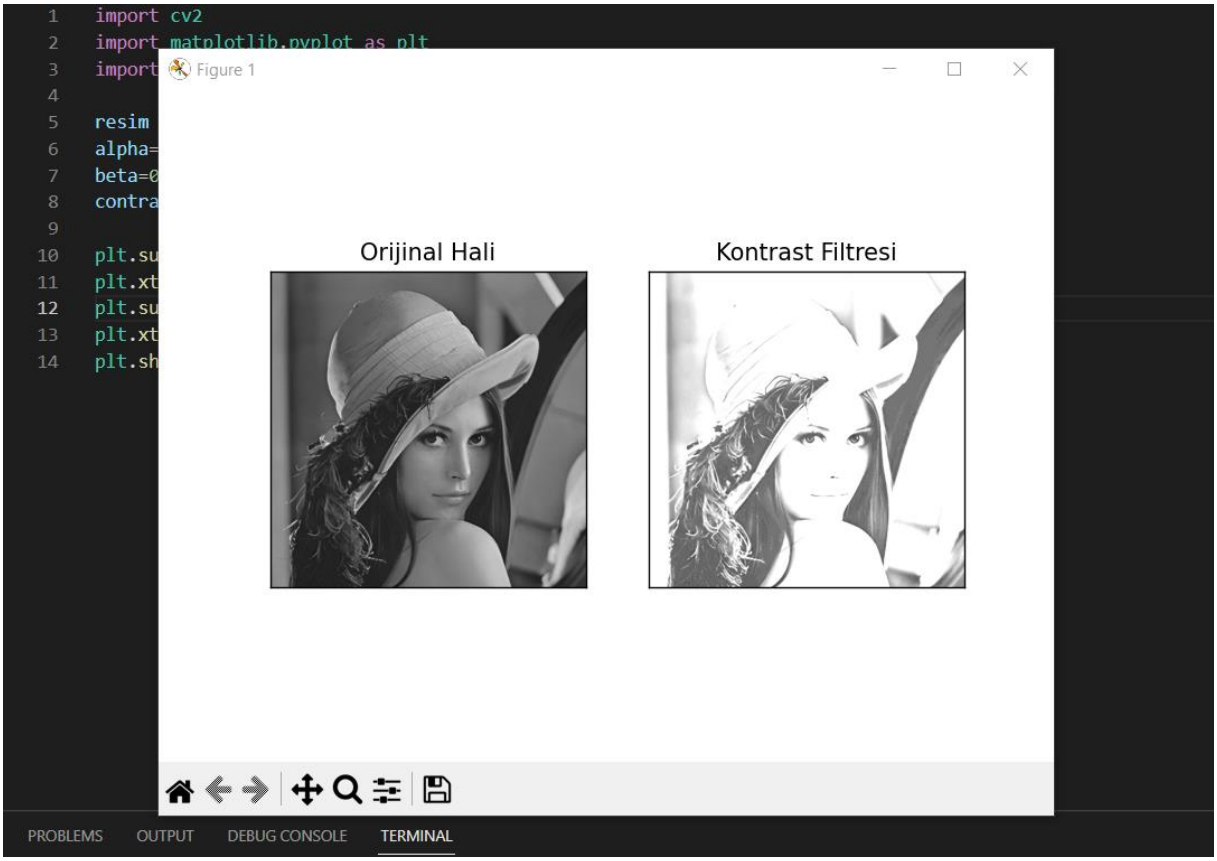
Her piksel renk değeri 255 sayısından çıkarıldığında geriye kalan değer, negatif rengi verir.




```
import cv2
import matplotlib.pyplot as plt
import numpy as np

resim=cv2.imread("resimler/lena.png")
renkli_negatif = abs(255-resim)
cv2.imshow("Orijinal Resim",resim)
cv2.imshow("Negatif Resim",renkli_negatif)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

7. KONTRAST AYARLAMA

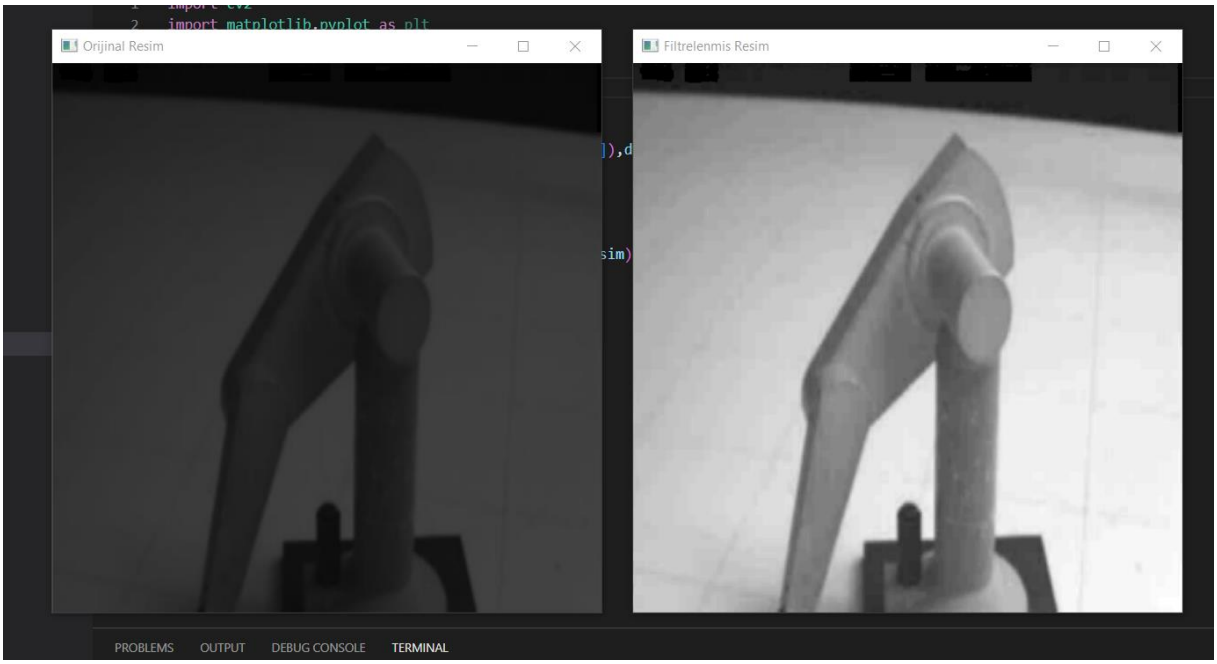
Kontrast, görüntüdeki en parlak ve en karanlık bölümler arasındaki farktır.




```
import cv2
import matplotlib.pyplot as plt
import numpy as np

resim = cv2.imread('resimler/grilena.png')
alpha=3
beta=0
contrast = cv2.convertScaleAbs(resim, alpha=alpha, beta=beta)

plt.subplot(121),plt.imshow(resim),plt.title('Orijinal Hali')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(contrast),plt.title('Kontrast Filtresi')
plt.xticks([]), plt.yticks([])
plt.show()
```

8. KONTRASTI GERME İŞLEMİ

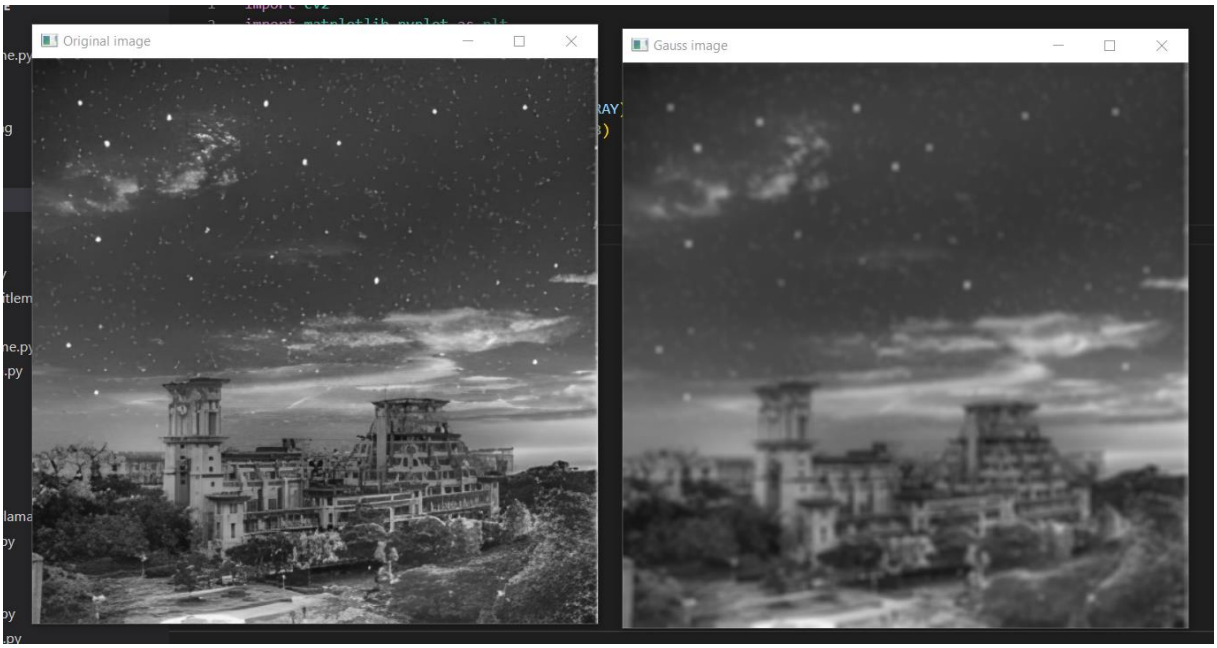
Kontrast germe, görüntüdeki gri- seviyelerin dinamik aralığını artırmayı amaçlar.




```
import cv2
import matplotlib.pyplot as plt
import numpy as np

resim = cv2.imread('resimler/kontrast germe.png',0)


minmax_img = np.zeros((resim.shape[0],resim.shape[1]),dtype = 'uint8')


for i in range(resim.shape[0]):
    for j in range(resim.shape[1]):
        minmax_img[i,j] = 255*(resim[i,j]-np.min(resim))/(np.max(resim)-
np.min(resim))


cv2.imshow("Orijinal Resim",resim)
cv2.imshow("Filtrelenmis Resim",minmax_img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

9. GAUSS

Gauss yumuşatma operatörü, görüntüleri 'bulanıklaştırmak', ayrıntı ve gürültüyü ortadan kaldırmak için kullanılır.




```
import cv2
import matplotlib.pyplot as plt
import numpy as np

foto=cv2.imread("resimler/gauss.png")
foto = cv2.cvtColor(foto, cv2.COLOR_BGR2GRAY)
output2 = cv2.GaussianBlur(foto, (7, 7), 3)
cv2.imshow('Original image',foto)
cv2.imshow('Gauss image', output2)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

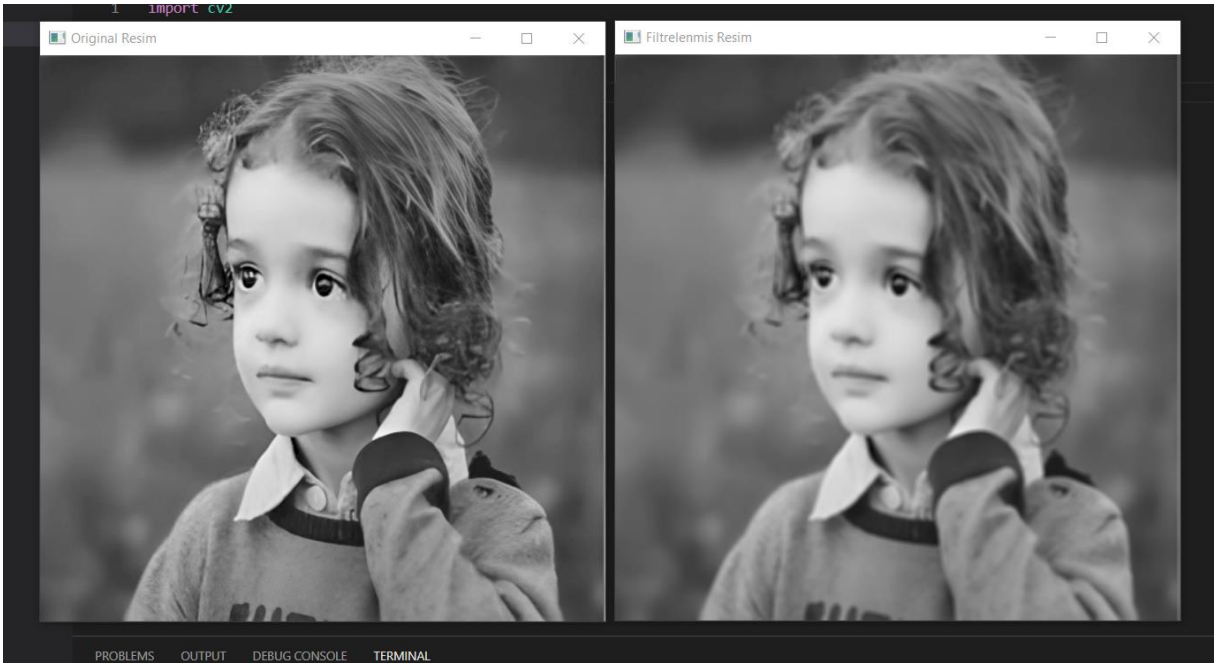
10. MEDYAN FİLTRESİ

Medyan filtresi, normal olarak mean filtresi gibi bir resimdeki gürültüyü azaltmak için kullanılır. Ancak resim üzerindeki detayların kaybolmaması noktasında mean filtresinden çok daha iyi sonuç verir.


```python
import cv2
import numpy as np
from matplotlib import pyplot as plt



img = cv2.imread('resimler/medyan.png',0)

filteredImg = cv2.medianBlur(img, ksize=3)

cv2.imshow('Original image', img)
cv2.imshow('Filtered image', filteredImg)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

11.MEAN FİLTRESİ

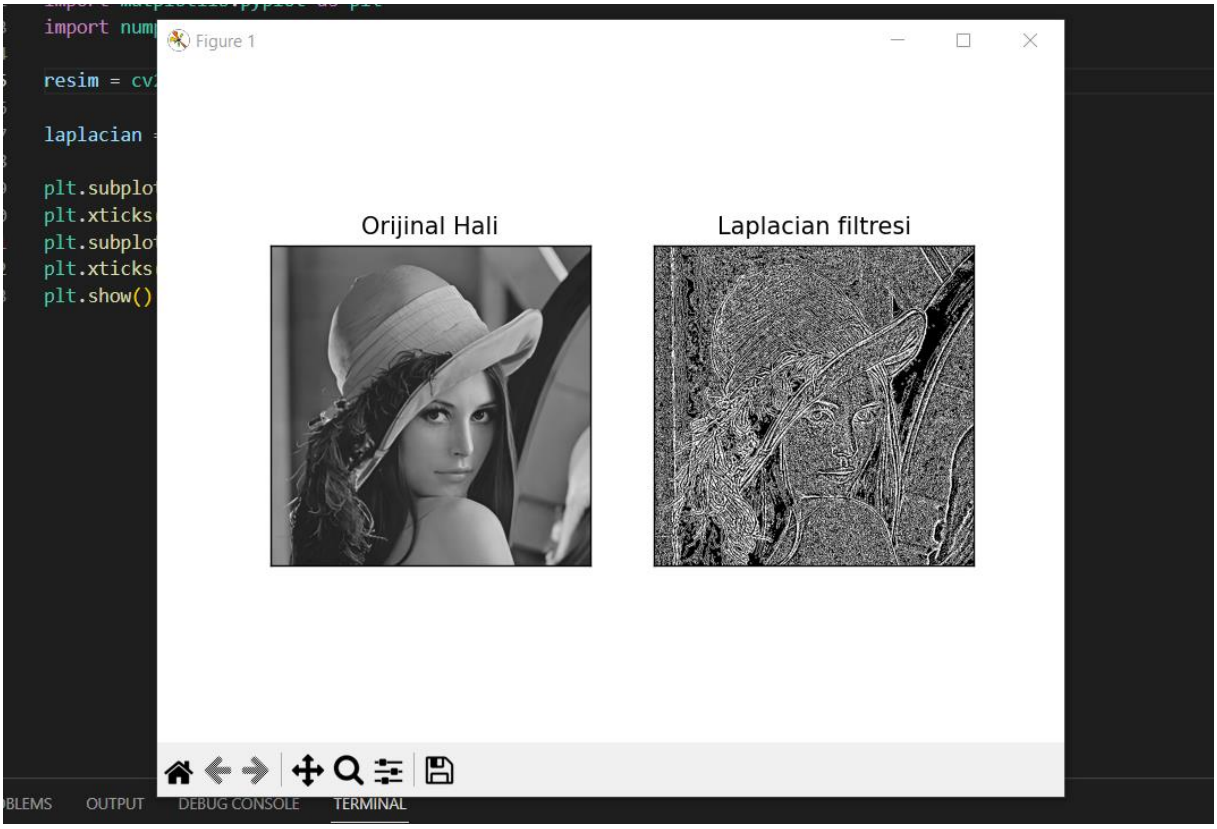
Medyan filtresi, genellikle bir görüntüden gürültüyü gidermek için kullanılır.




```
import cv2
import numpy as np
from matplotlib import pyplot as plt

resim=cv2.imread("resimler/meangri.png")
kernel = np.ones((5,5),np.float32)/25
dst = cv2.filter2D(resim,-1,kernel)

cv2.imshow('Original Resim', resim)
cv2.imshow('Filtrelenmis Resim', dst)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

12.LAPLACİAN FİLTRESİ

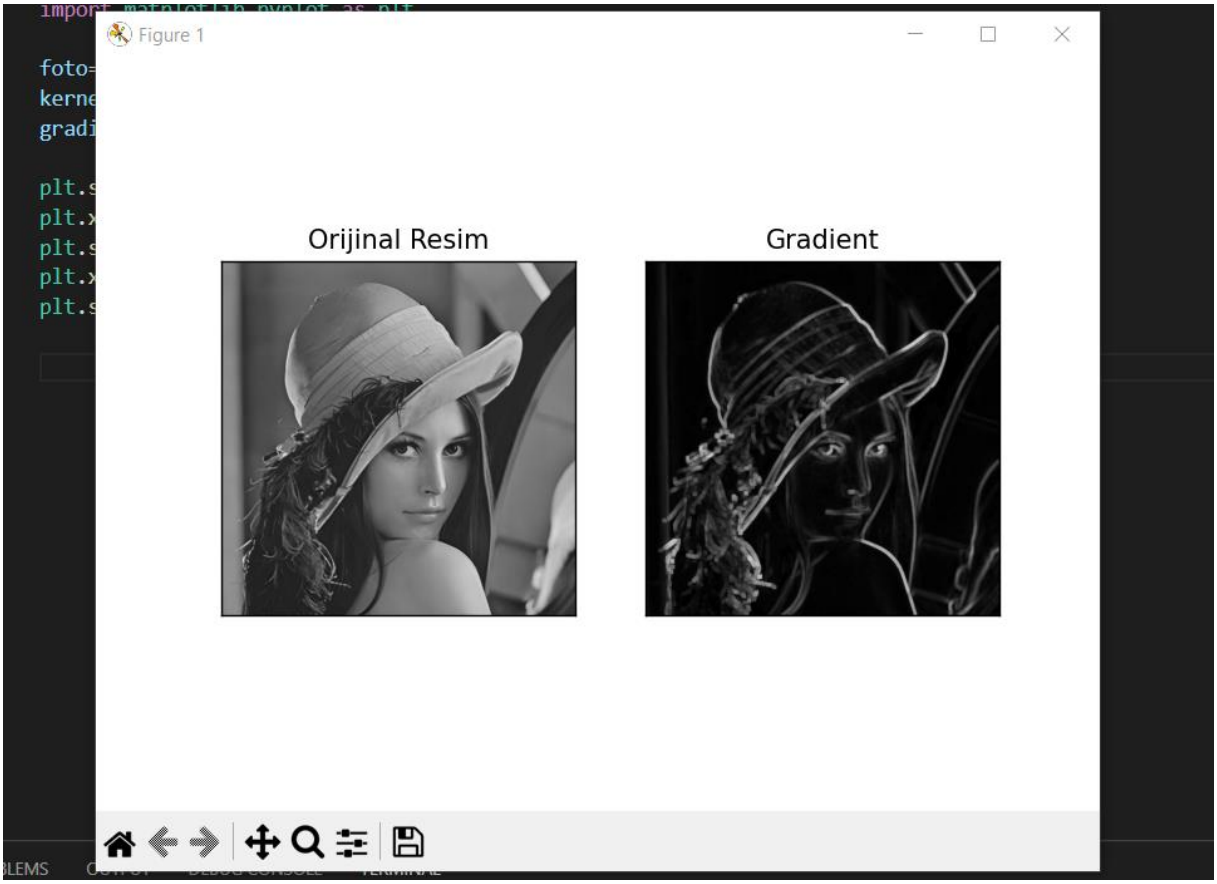
Laplas filitresi bastiçe bir resimdeki kenar hatlarını belirlemek için kullanılır.




```
import cv2
import matplotlib.pyplot as plt
import numpy as np

resim = cv2.imread('resimler/grilena.png')

laplacian = cv2.Laplacian(resim, cv2.CV_64F)

plt.subplot(121),plt.imshow(resim),plt.title('Orijinal Hali')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(laplacian),plt.title('Laplacian filtresi')
plt.xticks([]), plt.yticks([])
plt.show()
```

13.GRADYENT METODUYLA KENAR KESKİNLEŞTİRME

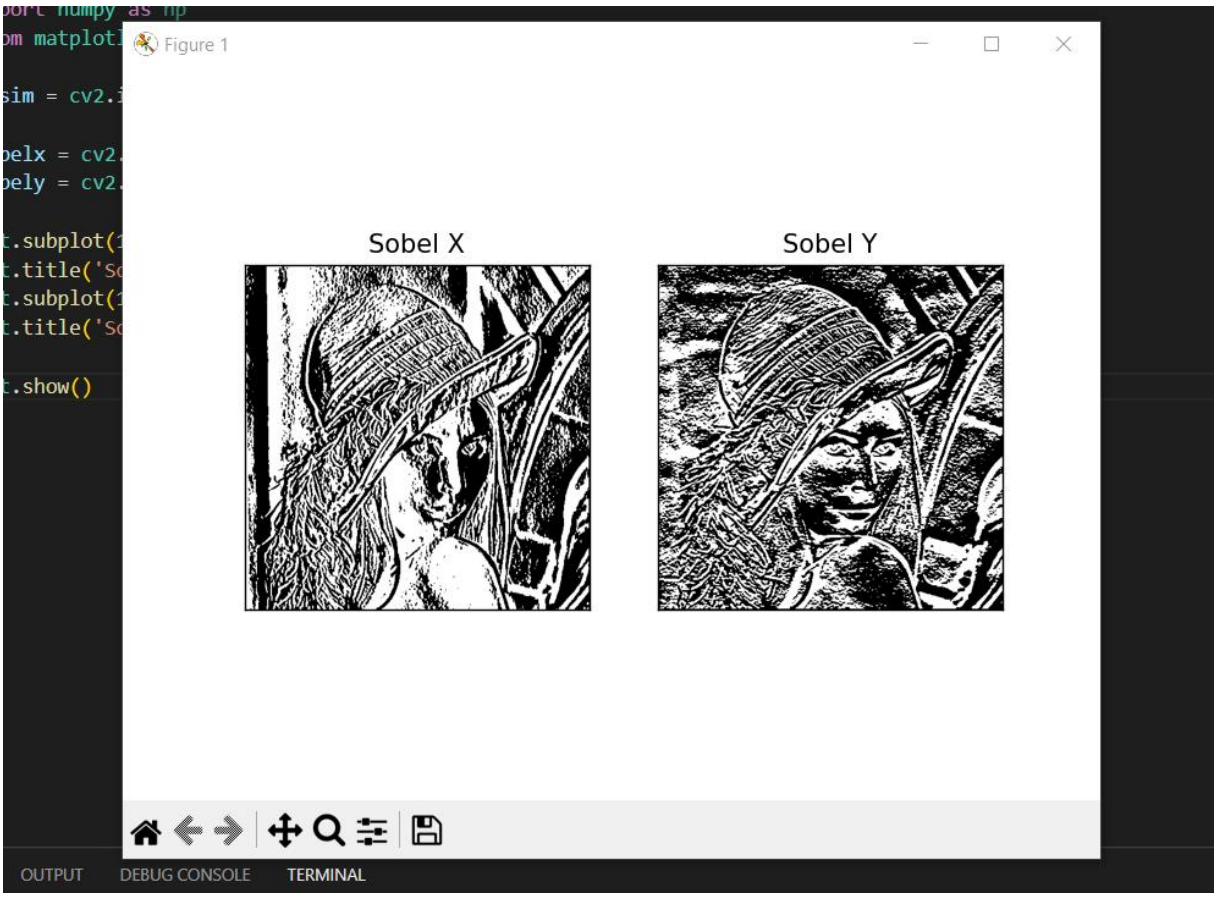
Gradyent, görüntünün x ve y doğrultusunda birinci dereceden türev alma işlemidir.




```
import cv2
import numpy as np
import matplotlib.pyplot as plt

foto=cv2.imread("resimler/grilena.png")
kernel = np.ones((5,5),np.uint8)
gradient= cv2.morphologyEx(foto, cv2.MORPH_GRADIENT, kernel)

plt.subplot(121), plt.imshow(foto), plt.title("Orijinal Resim")
plt.xticks([]), plt.yticks([])
plt.subplot(122), plt.imshow(gradient), plt.title("Gradient")
plt.xticks([]), plt.yticks([])
plt.show()
```

14. SOBEL FİLTRESİ

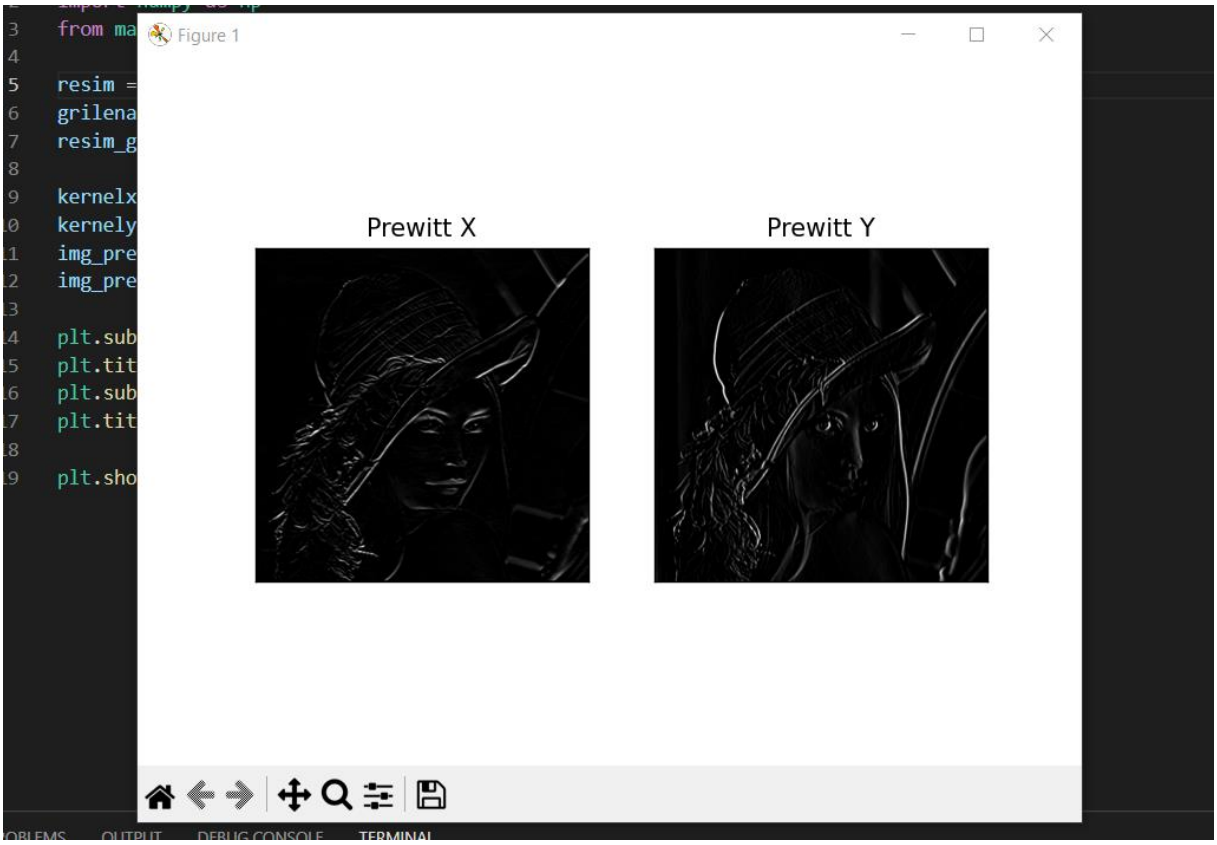
Bu filtre, bir resimdeki farklı renkler arasındaki sınırları bularak resimde yer alan nesnelerin dış hatlarını belirlememizi sağlar.




```python
import cv2
import numpy as np
from matplotlib import pyplot as plt

resim = cv2.imread('resimler/grilena.png')

sobelx = cv2.Sobel(resim,cv2.CV_64F,1,0,ksize=5)  # x
sobely = cv2.Sobel(resim,cv2.CV_64F,0,1,ksize=5)  # y

plt.subplot(121),plt.imshow(sobelx,cmap = 'gray')
plt.title('Sobel X'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(sobely,cmap = 'gray')
plt.title('Sobel Y'), plt.xticks([]), plt.yticks([])

plt.show()
```

15.PREWİTT FİLTRESİ

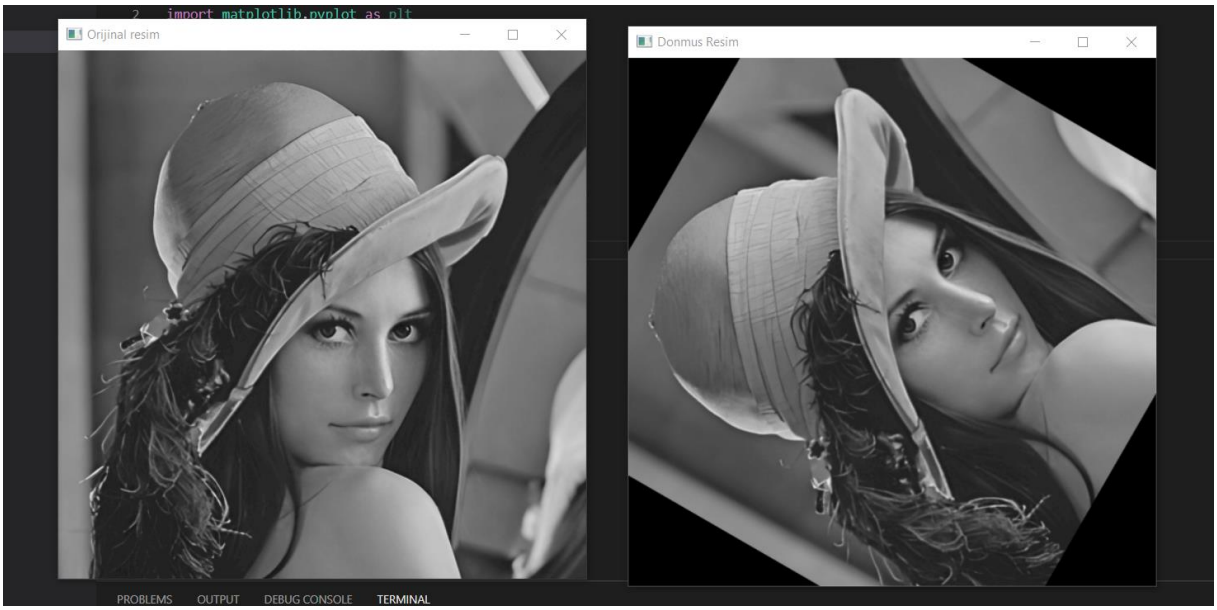
Resimdeki farklı renkler arasındaki sınırları bularak resimde yer alan nesnelerin dış hatlarını belirlememizi sağlar.




```
import cv2
import numpy as np
from matplotlib import pyplot as plt

resim = cv2.imread('resimler/grilena.png')
grilena = cv2.cvtColor(resim, cv2.COLOR_BGR2GRAY)
resim_gaussian = cv2.GaussianBlur(grilena,(3,3),0)

kernelx = np.array([[1,1,1],[0,0,0],[-1,-1,-1]])
kernely = np.array([[-1,0,1],[-1,0,1],[-1,0,1]])
img_prewittx = cv2.filter2D(resim_gaussian, -1, kernelx)
img_prewitty = cv2.filter2D(resim_gaussian, -1, kernely)

plt.subplot(121),plt.imshow(img_prewittx,cmap = 'gray')
plt.title('Prewitt X'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(img_prewitty,cmap = 'gray')
plt.title('Prewitt Y'), plt.xticks([]), plt.yticks([])

plt.show()
```

16. AÇILI DÖNDÜRME

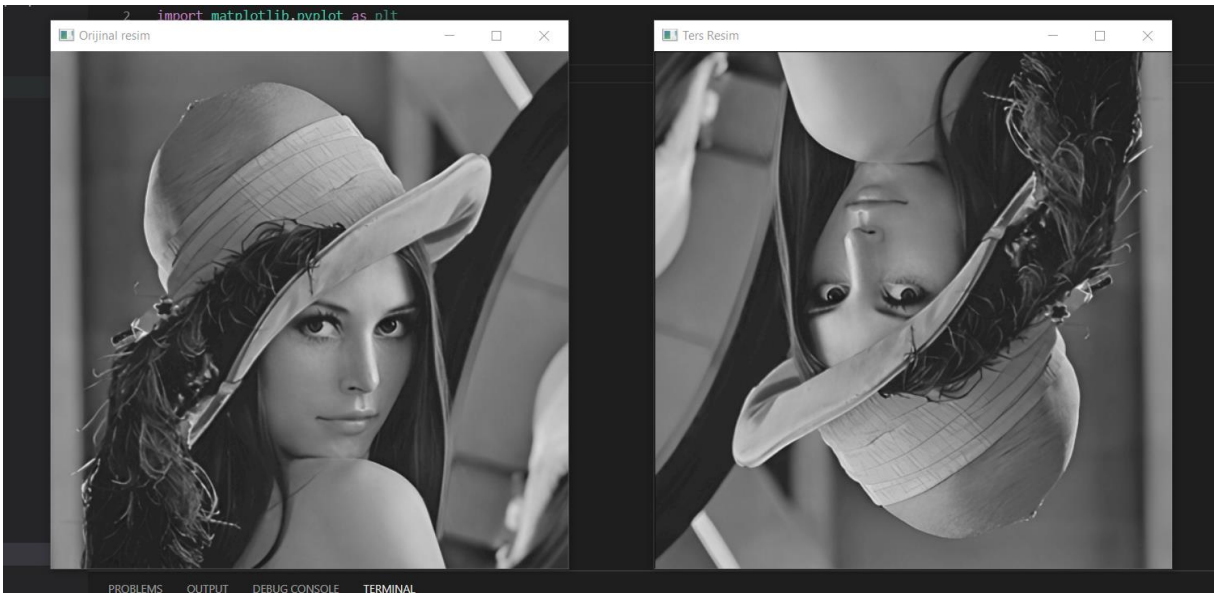
Bir görüntünün açılı döndürülmesinden kasıt verilen açıya göre saat yönünde ya da saat yönünün tersi yönünde görüntü düzleminin döndürülmesidir.




```python
import cv2
import matplotlib.pyplot as plt
import numpy as np

resim = cv2.imread('resimler/lena.png',0)
rows,cols = resim.shape

M = cv2.getRotationMatrix2D((cols/2,rows/2),60,1)
cikti = cv2.warpAffine(resim,M,(cols,rows))

cv2.imshow("Orijinal resim",resim)
cv2.imshow('Donmus Resim',cikti)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

17. GÖRÜNTÜNÜN TERS ÇEVRİLMESİ

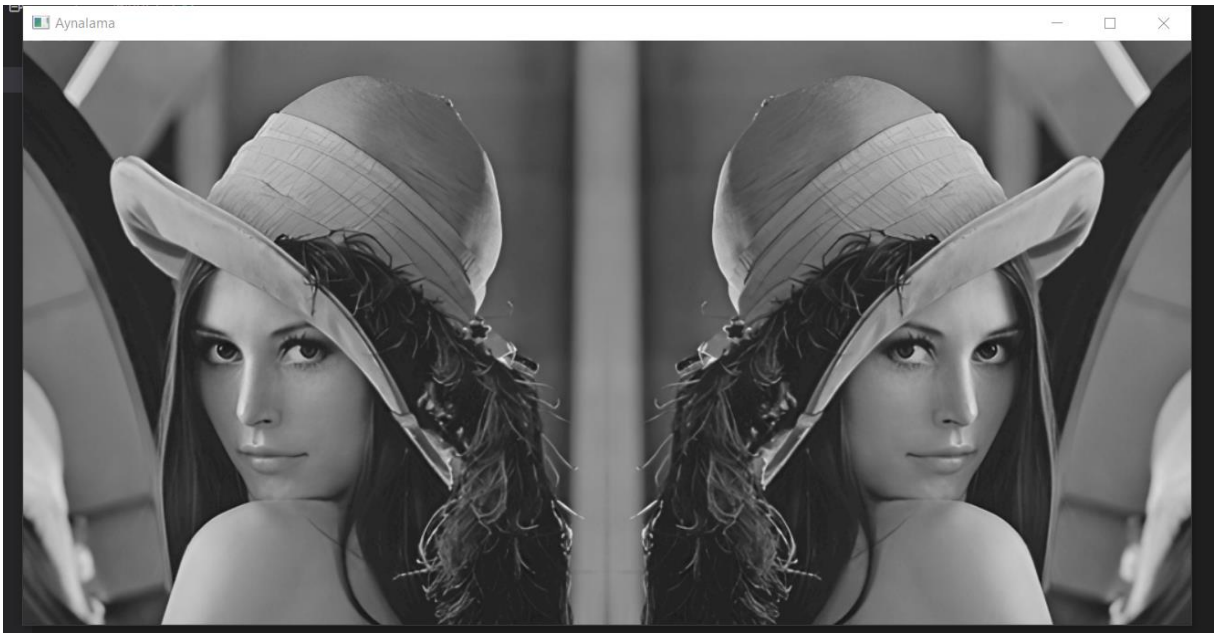
Görüntünün ters çevrilmesi için en üstteki satır en alta, en alttaki satır ise en üste gelecek şekilde görüntü matrisinin tüm elemanları karşılıklı olarak yer değiştirmesi ile sağlanır.




```python
import cv2
import matplotlib.pyplot as plt
import numpy as np

resim = cv2.imread('resimler/lena.png',0)
rows,cols = resim.shape

M = cv2.getRotationMatrix2D((cols/2,rows/2),180,1)
cıktı = cv2.warpAffine(resim,M,(cols,rows))

cv2.imshow("Orijinal resim",resim)
cv2.imshow('Ters Resim',cıktı)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

18. AYNALAMA

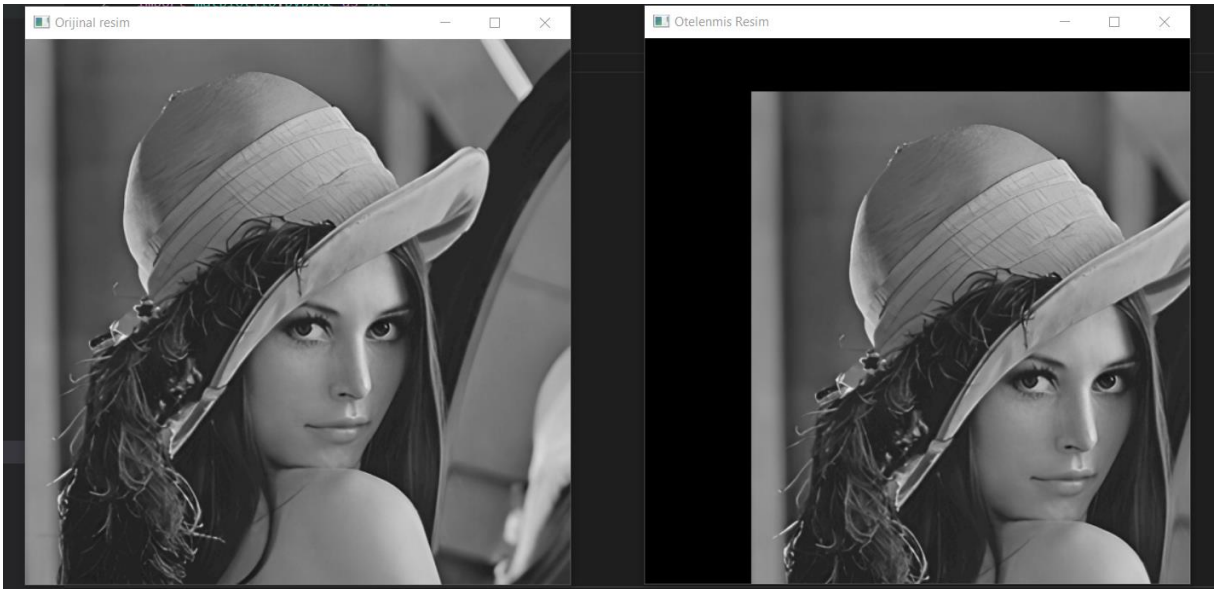
Aynalama işlemi, ters çevirme işleminin düşey eksende yapılmış halidir. Birinci sütun elemanları ile son sütun elemanları sırası ile yer değiştirir.




```
import cv2
import matplotlib.pyplot as plt
import numpy as np


foto=cv2.imread("resimler/grilena.png")

aynalananResim= cv2.copyMakeBorder(foto,0,0,512,0,cv2.BORDER_REFLECT)

cv2.imshow("Aynalama",aynalananResim)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

19. GÖRÜNTÜ ÖTELEME

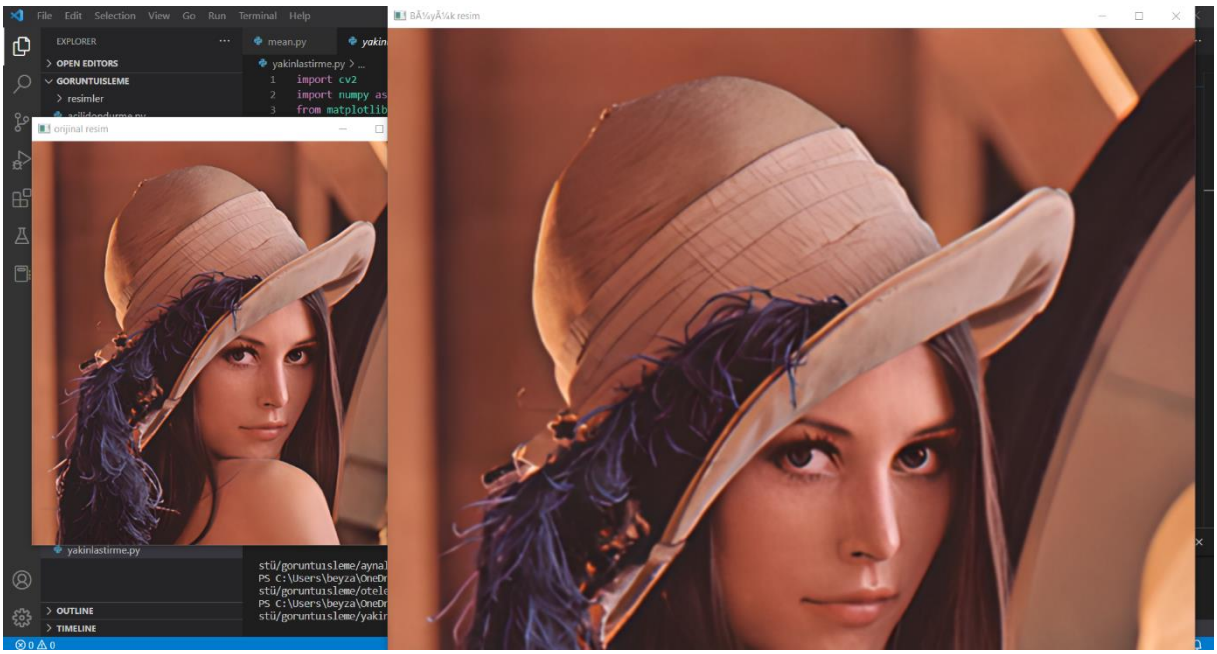
Öteleme işlemi yatay ve dikey eksende belirlenen piksel kadar görüntünün yatay ve düşey eksende kaydırılmasıdır.




```
import cv2
import matplotlib.pyplot as plt
import numpy as np

resim = cv2.imread('resimler/lena.png',0)
rows,cols = resim.shape

M = np.float32([[1,0,100],[0,1,50]])
cikti = cv2.warpAffine(resim,M,(cols,rows))

cv2.imshow("Orijinal resim",resim)
cv2.imshow('Otelenmis Resim',cikti)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

20. YAKINLAŞTIRMA

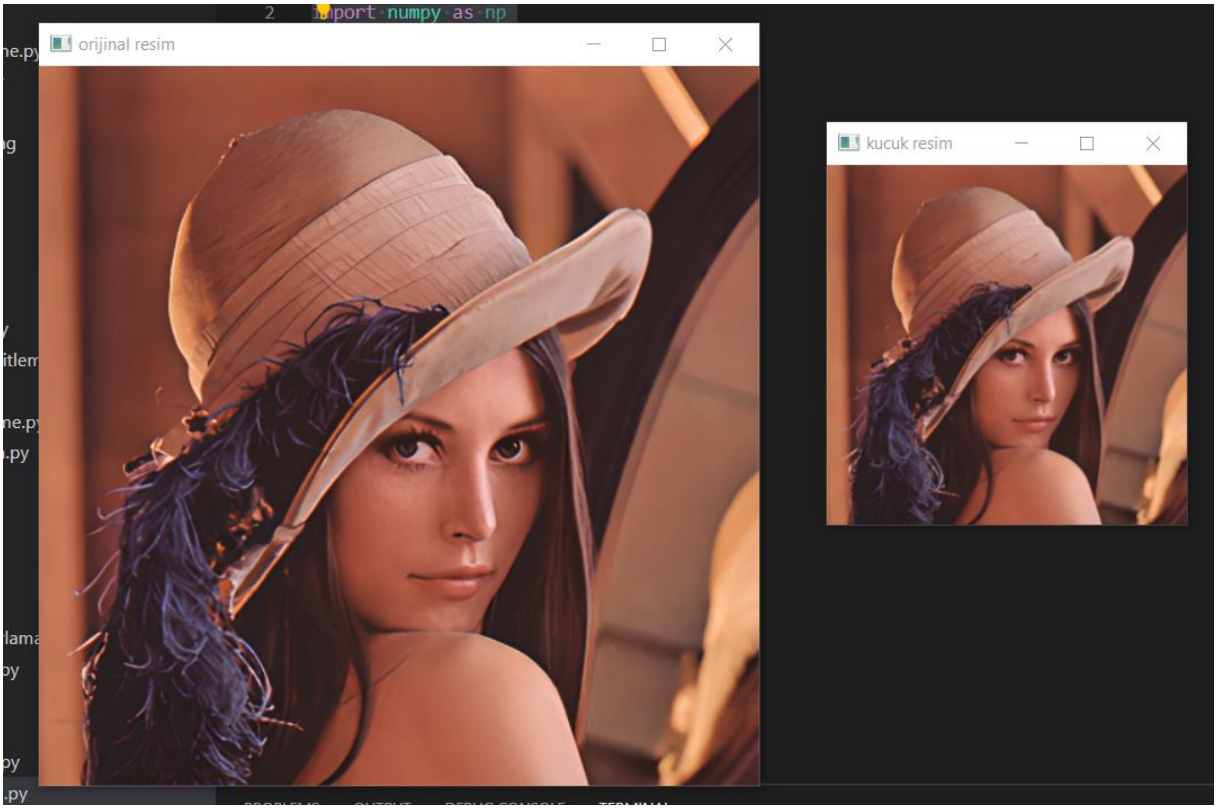
Yakınlaştırma, düşük piksel boyutlu bir imgenin piksel boyutunun yazılımsal olarak arttırılmasıdır.



```
import cv2
import numpy as np
from matplotlib import pyplot as plt

resim = cv2.imread('resimler/lena.png')
BuyukResim=cv2.pyrUp(resim)


cv2.imshow("orijinal resim",resim)
cv2.imshow("Büyük resim",BuyukResim)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

21. UZAKLAŞTIRMA

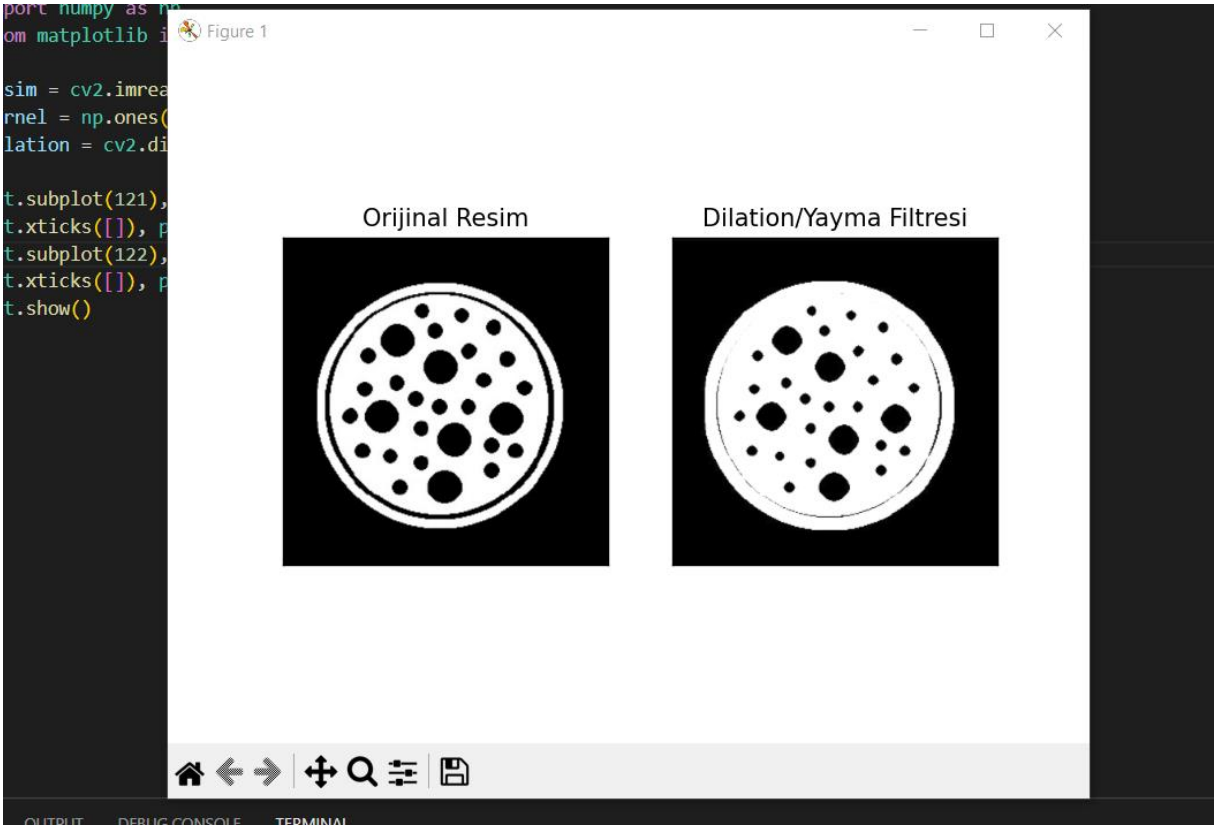
Birden fazla pikselin değeri çeşitli matematiksel işlemlerden geçirilerek bir piksele atanır.




```python
import cv2
import numpy as np
from matplotlib import pyplot as plt

resim = cv2.imread('resimler/lena.png')
KucukResim=cv2.pyrDown(resim)


cv2.imshow("orijinal resim",resim)
cv2.imshow("kucuk resim",KucukResim)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

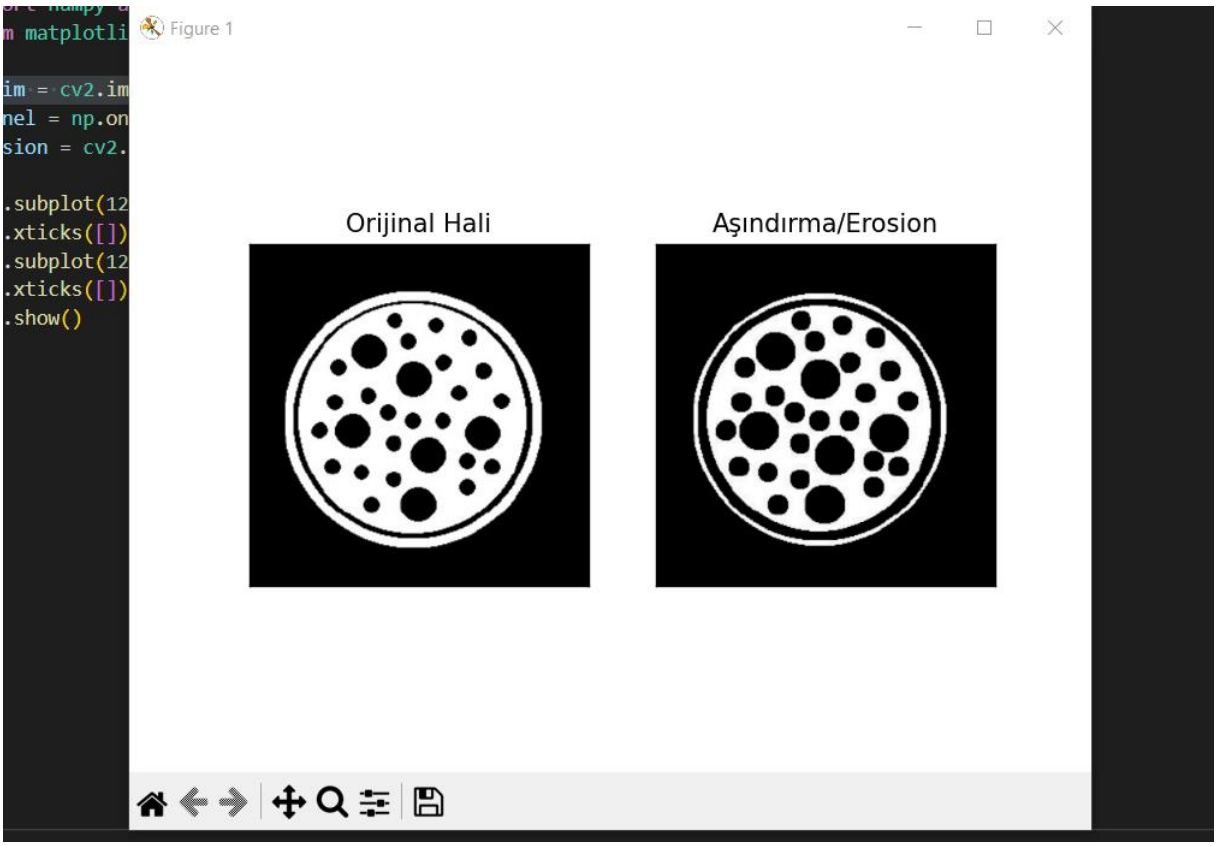
22. YAYMA

Yayma, ikili imgedeki nesneyi büyütmeye ya da kalınlaştırmaya yarayan morfolojik işlemdir.




```python
import cv2
import numpy as np
from matplotlib import pyplot as plt

resim = cv2.imread('resimler/morfoloji2.jpeg')
kernel = np.ones((5,5),np.float32)/25
dilation = cv2.dilate(resim,kernel,iterations = 1)

plt.subplot(121),plt.imshow(resim),plt.title('Orijinal Resim')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(dilation),plt.title('Dilation/Yayma Filtresi')
plt.xticks([]), plt.yticks([])
plt.show()
```

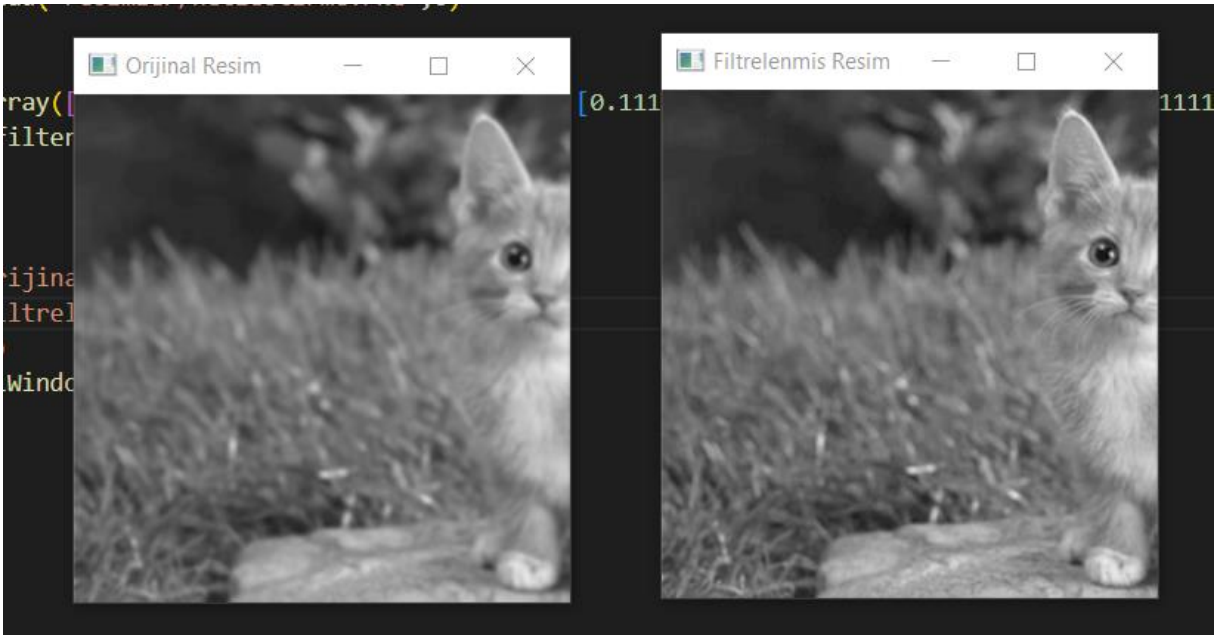
23. AŞINDIRMA




```
import cv2
import numpy as np
from matplotlib import pyplot as plt

resim = cv2.imread('resimler/morfoloji2.jpeg')
kernel = np.ones((5,5),np.float32)/25
erosion = cv2.erode(resim,kernel,iterations = 1)

plt.subplot(121),plt.imshow(resim),plt.title('Orijinal Hali')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(erosion),plt.title('Aşındırma/Erosion')
plt.xticks([]), plt.yticks([])
plt.show()
```

24. KONVOLÜSYON İLE NETLEŞTİRME

Bu algoritma orjinal görüntüden, görüntünün yumuşatılmış halini çıkararak belirgin kenarların görüntüsünü ortaya çıkarır. Daha sonra orjinal görüntü ile belirginleşmiş kenarların görüntüsünü birleştirerek, kenarları keskinleşmiş görüntüyü (netleşmiş görüntü) elde eder.




```
import cv2
import numpy as np

from matplotlib import pyplot as plt

resim=cv2.imread("resimler/netlestirme.PNG",0)


kernel = np.array([[0.1111111, 0.1111111, 0.1111111], [0.1111111, 0.1111111,
0.1111111], [0.1111111, 0.1111111, 0.1111111]])
resim2 = cv2.filter2D(resim,-1,kernel)



cv2.imshow('Orijinal Resim', resim2)
cv2.imshow('Filtrelenmis Resim', resim)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

25. PERSPEKTİF DÜZELTME

Perspektif düzeltmede amaç kişinin veya nesnenin konum değiştirmesi sonucu oluşacak etkiyi düzeltmektir. Bu işlem sayesinde görüntü oluştuktan sonra dahi belirli kısıtlar içerisinde resme baktığımız açıyı değiştirebiliriz.


```python
import cv2
import numpy as np

img = cv2.imread('resimler/perspektif.jpg')
rows,cols,ch = img.shape
pts1 = np.float32([[32,131],[505,95],[30,245],[463,353]])
pts2 = np.float32([[0,0],[500,0],[0,400],[500,400]])
M = cv2.getPerspectiveTransform(pts1,pts2)
dst = cv2.warpPerspective(img,M,(cols, rows))

cv2.imshow('Orjinal Resim', img)
cv2.imshow('Transformed Image', dst)
cv2.waitKey(0)
cv2.destroyAllWindows()
```