



# BCA 607 Hareket Analizi Sistemleri

## #2



## SERDAR ARITAN

[serdar.aritan@hacettepe.edu.tr](mailto:serdar.aritan@hacettepe.edu.tr)

Biyomekanik Araştırma Grubu  
[www.biomech.hacettepe.edu.tr](http://www.biomech.hacettepe.edu.tr)  
Spor Bilimleri Fakültesi  
[www.sbt.hacettepe.edu.tr](http://www.sbt.hacettepe.edu.tr)  
Hacettepe Üniversitesi, Ankara, Türkiye  
[www.hacettepe.edu.tr](http://www.hacettepe.edu.tr)

## Yansıtıcı İşaret Tabanlı Hareket Yakalama Sistemlerine Giriş



## Yansıtıcı İşaret Tabanlı Hareket Yakalama Sistemlerine Giriş



## Yansıtıcı İşaret Tabanlı Hareket Yakalama Sistemlerine Giriş





# http://winpython.github.io/



releases overview portable

## WinPython

The easiest way to run Python, Spyder with SciPy and friends out of the box on any Windows PC, without installing anything!

Project Home is on [Github](#), downloads pages are on [Sourceforge](#) and [Github](#), [md5-sha](#), [Discussion Group](#)

### Recent Releases

Release [2020-05](#) of December 28st, 2020

Highlights (\*): Spyder-4.2.1, VSCode-1.52.1, Pandas-1.1.5, scikit\_learn-0.24.0, SciPy-1.5.4, Numpy-1.19.4+mkl

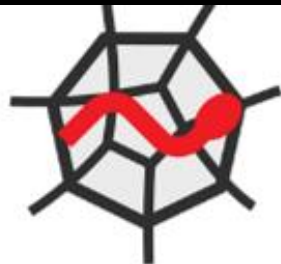
WinPython **3.8** Downloads (\*\*) via [SourceForge](#) and [Github](#)

- WinPython64-**3.8.7.0**dot = Python 3.8 64bit only : [Changelog](#), [Packages](#)
- WinPython32-**3.8.7.0**dot = Python 3.8 32bit only : [Changelog](#), [Packages](#)
- WinPython64-**3.8.7.0** = Python 3.8 64bit + PyQt5 + Spyder + Pytorch : [Changelog](#), [Packages](#)
- WinPython64-**3.8.7.0**cod = Python 3.8 64bit + PyQt5 + Spyder + VSCode : [Changelog](#), [Packages](#)

WinPython **3.9** Downloads (\*\*) via [SourceForge](#) and [Github](#)

- WinPython64-**3.9.1.0**dot = Python 3.9 64bit only : [Changelog](#), [Packages](#)
- WinPython32-**3.9.1.0**dot = Python 3.9 32bit only : [Changelog](#), [Packages](#)
- WinPython64-**3.9.1.0** = Python 3.9 64bit + PyQt5 + Spyder + Pytorch : [Changelog](#), [Packages](#)
- WinPython64-**3.9.1.0**cod = Python 3.9 64bit + PyQt5 + Spyder + VSCode : [Changelog](#), [Packages](#)

Release [2020-04](#) of October 31st, 2020



# SPYDER

The Scientific Python Development Environment

The screenshot displays the Spyder IDE interface with the following components:

- Left Panel (File Explorer):** Shows a project structure with files like `App.py`, `template.py`, `plot_example.py`, and `plugin.py`. The `plot_example.py` file is selected.
- Central Panel (Code Editor):** Displays the content of `plot_example.py`, which includes imports for `numpy`, `matplotlib.pyplot`, `matplotlib.cm`, and `mpl_toolkits.mplot3d`. It defines functions `generate_polar_plot()` and `generate_dem_plot()`.
- Right Panel (Variable Explorer):** Shows a table of variables in the current namespace.
 

Name	Type	Size	Value
data	Array of str128	(3, 3)	ndarray object of numpy module
df	DataFrame	(2, 2)	Column names: Col1, Col2
filename	str	1	/Users/Documents/spyder/spyder/tests/test_dont_use.py
i	Array of uint32	(10, 10)	[[0 0 ... 0 0] [0 0 ... 0 0]
li	list	5	['abcd', 745, 2.23, 'efgh', 70.2]
r	float	1	6.46567886443
t	tuple	5	('abcd', 745, 2.23, 'efgh', 70.2)
tinylist	list	2	[123, 'efgh']
- Bottom Panel (Plots):** Displays two plots: a 3D surface plot of a terrain model and a polar plot showing data distribution.
- Status Bar:** Shows the current state: "LSP Python: restarting...", "conda: base (Python 3.7.4)", "Line 1, Col 1", "ASCII", "LF", "RW", and "Mem 50K".



# OpenCV

Açık Kaynak Bilgisayar Görüşü Kütüphanesi



OpenCV



Spyder (Python 3.9)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\SA-Lenovo\spyder-py3\temp.py

```
1 # -*- coding: utf-8 -*-
2 """
3 Spyder Editor
4 This is a temporary script file.
5 """
6
7
8
```

Nam	Type	Size	Value
i	int	1	10
s	str	11	hello world
t	Array of float64 (200,)		[0. 0.0315738 0.06314759 ... 6.22003772 6.25161151 6.28318531 ...

# Editor

# Bellek

Variable explorer Help Plots Files

Console 1/A

```
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.19.0 -- An enhanced Interactive Python.

In [1]: import numpy as np
In [2]: i = 10
In [3]: t = np.linspace(0, 2*np.pi, 200)
In [4]: s = 'hello world'
In [5]:
```

# Etkileşimli Python

Python console History

LSP Python: ready custom (Python 3.9.1) Line 1, Col 21 UTF-8 CRLF RW Mem 46%



# OpenCV çalışıyor mu?

The screenshot shows the Spyder Python IDE interface. The main editor displays a Python script named `ceaserChipper.py` with the following code:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Thu Sep 17 13:59:30 2015
4
5 @author: SA
6 """
7 import cv2 as cv
8 from matplotlib import pyplot as plt
9
10 RGB = cv.imread('topbirak_041.jpg')
11 imggray = cv.cvtColor(RGB, cv.COLOR_BGR2GRAY) # Convert to Gray
12
13 _, thresh1 = cv.threshold(imggray, 127, 255, cv.THRESH_BINARY)
14 _, thresh2 = cv.threshold(imggray, 127, 255, cv.THRESH_BINARY_INV)
15 _, thresh3 = cv.threshold(imggray, 127, 255, cv.THRESH_TRUNC)
16 _, thresh4 = cv.threshold(imggray, 127, 255, cv.THRESH_TOZERO)
17 _, thresh5 = cv.threshold(imggray, 127, 255, cv.THRESH_TOZERO_INV)
18
19 titles = ['Original Image', 'BINARY', 'BINARY_INV', 'TRUNC', 'TOZERO', 'TOZERO_INV']
20 images = [imggray, thresh1, thresh2, thresh3, thresh4, thresh5]
21
22 for i in range(6):
23     plt.subplot(2, 3, i+1), plt.imshow(images[i], 'gray')
24     plt.title(titles[i])
25     plt.xticks([], plt.yticks([]))
26
27 plt.show()
28
29 cv.imshow('Original image', RGB)
30 cv.imshow('Gray image', imggray)
31 cv.imshow('Threshold', thresh1)
32 # median filter salt&pepper
33 filtered = cv.medianBlur(thresh1, 3)
34 cv.imshow('Filtered', filtered)
35
36 M = cv.moments(thresh1)
37 # calculate x,y coordinate of center
38 cX = int(M["m10"] / M["m00"])
39 cY = int(M["m01"] / M["m00"])
40
41 # put text and highlight the center
42 cv.circle(RGB, (cX, cY), 2, (0, 0, 255), -1)
43 cv.putText(RGB, "centroid", (cX - 25, cY - 25), cv.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2)
44 cv.imshow('Original image', RGB)
```

The right sidebar shows the Variable Explorer, Plots, and Files panels. The Console panel at the bottom shows the IPython 7.32.0 prompt and the execution of the first two lines of the script:

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun 6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.32.0 -- An enhanced Interactive Python.

In [1]: import cv2 as cv
In [2]:
```



```
import cv2 as cv
```

```
Console 1/A x
Python 3.10.5 (tags/v3.10.5:f377153, Jun 6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.32.0 -- An enhanced Interactive Python.

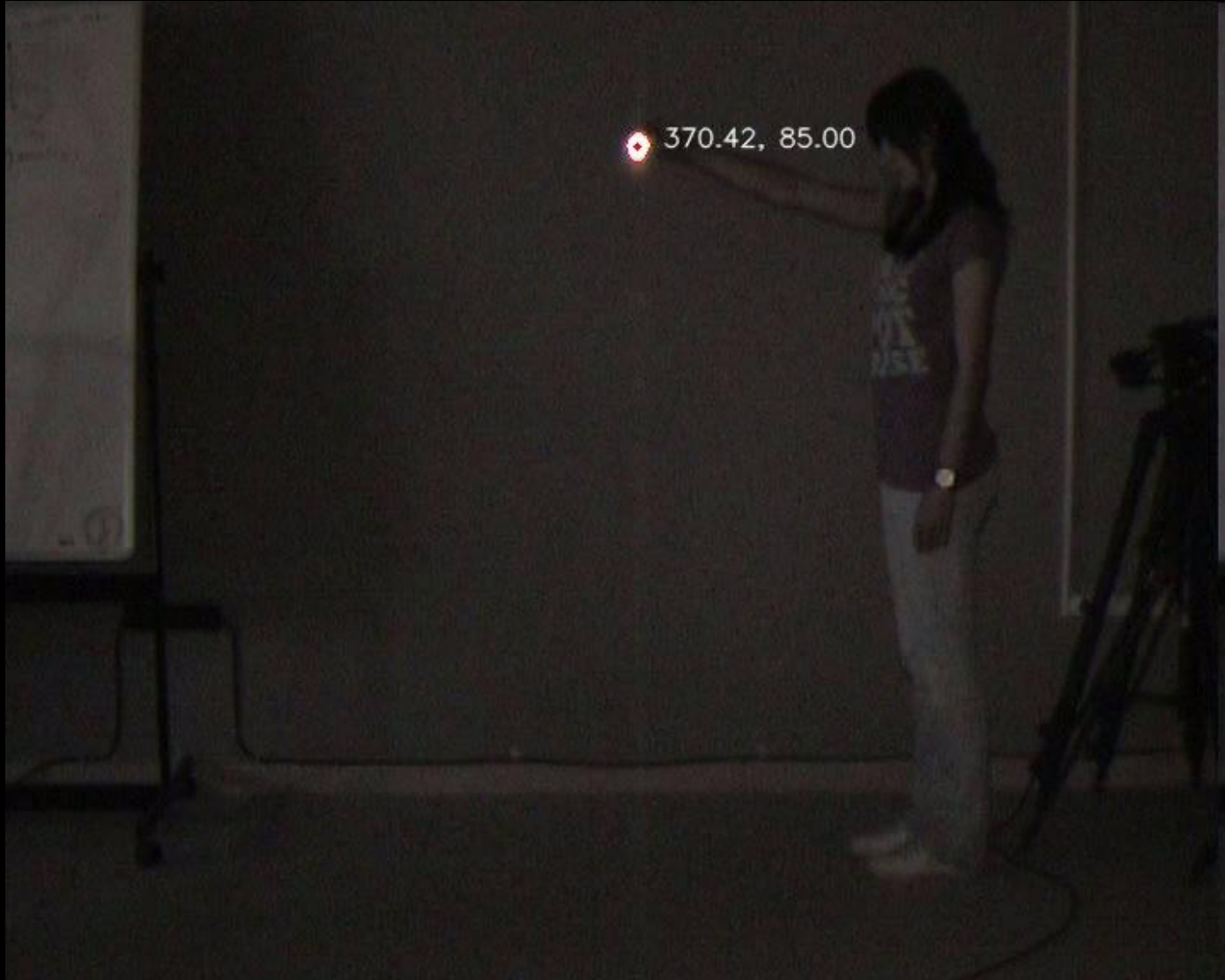
In [1]: import cv2 as cv

In [2]: |
```

Hata mesajı yoksa  
OpenCV kütüphanesi yüklenmiş

```
IPython Console History
```

## Yansıtıcı işaret yakalama



```
BGR = cv.imread('topbirak_041.jpg');
```

`imread()`  
decodes the image  
into a matrix with  
the color channels  
stored in the order  
of **Blue**, **Green** and  
**Red** respectively.

`imread` ile resim dosyasının SPYDER a yüklenmesi

**KYM** formatındaki renkli bir resim, **Kırmızı**, **Yeşil** ve **Mavi**  
ana renklerinin farklı miktardaki oranlarının karışımıyla  
oluşmaktadır.  
OpenCV **KYM** formatındaki bir resmi 3 boyutlu bir  
matris olarak saklamaktadır.

Nam	Type	Size	Value
BGR	Array of uint8	(576, 720, 3)	[[[43 44 48] [42 43 47]

imread ile resim dosyasının SPYDER a yüklenmesi

**KYM** formatındaki renkli bir resim, **Kırmızı**, **Yeşil** ve **Mavi**  
ana renklerinin farklı miktardaki oranlarının karışımıyla  
oluşmaktadır.  
OpenCV **KYM** formatındaki bir resmi 3 boyutlu bir  
matris olarak saklamaktadır.

Help Variable Explorer Plots Files

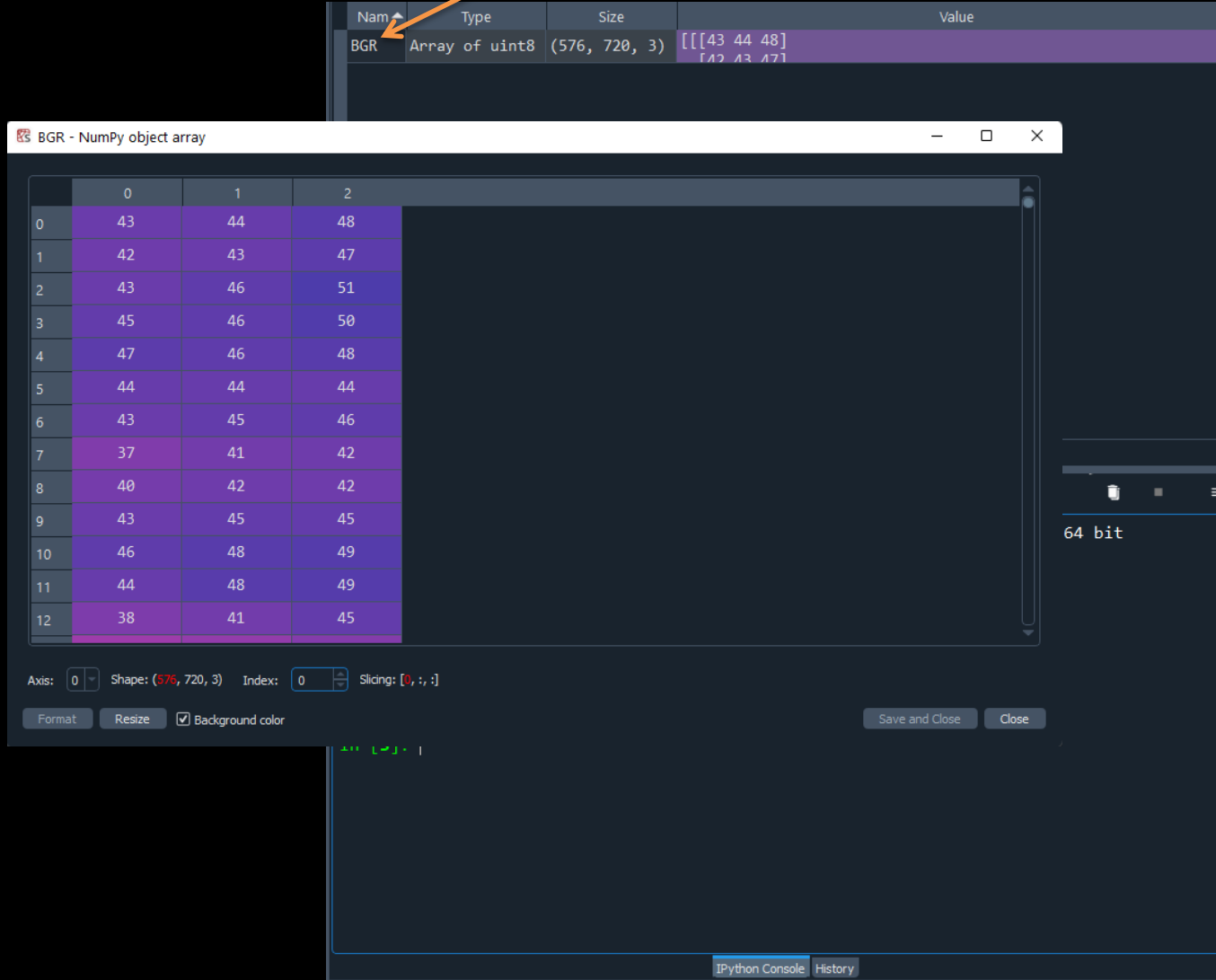
Console 1/A x

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun 6 2022, 16:14:13) [MSC v.1929 64 bit  
(AMD64)]  
Type "copyright", "credits" or "license" for more information.  
  
IPython 7.32.0 -- An enhanced Interactive Python.  
  
In [1]: import cv2 as cv  
  
In [2]: BGR = cv.imread('topbirak_041.jpg')  
  
In [3]: |
```

IPython Console History

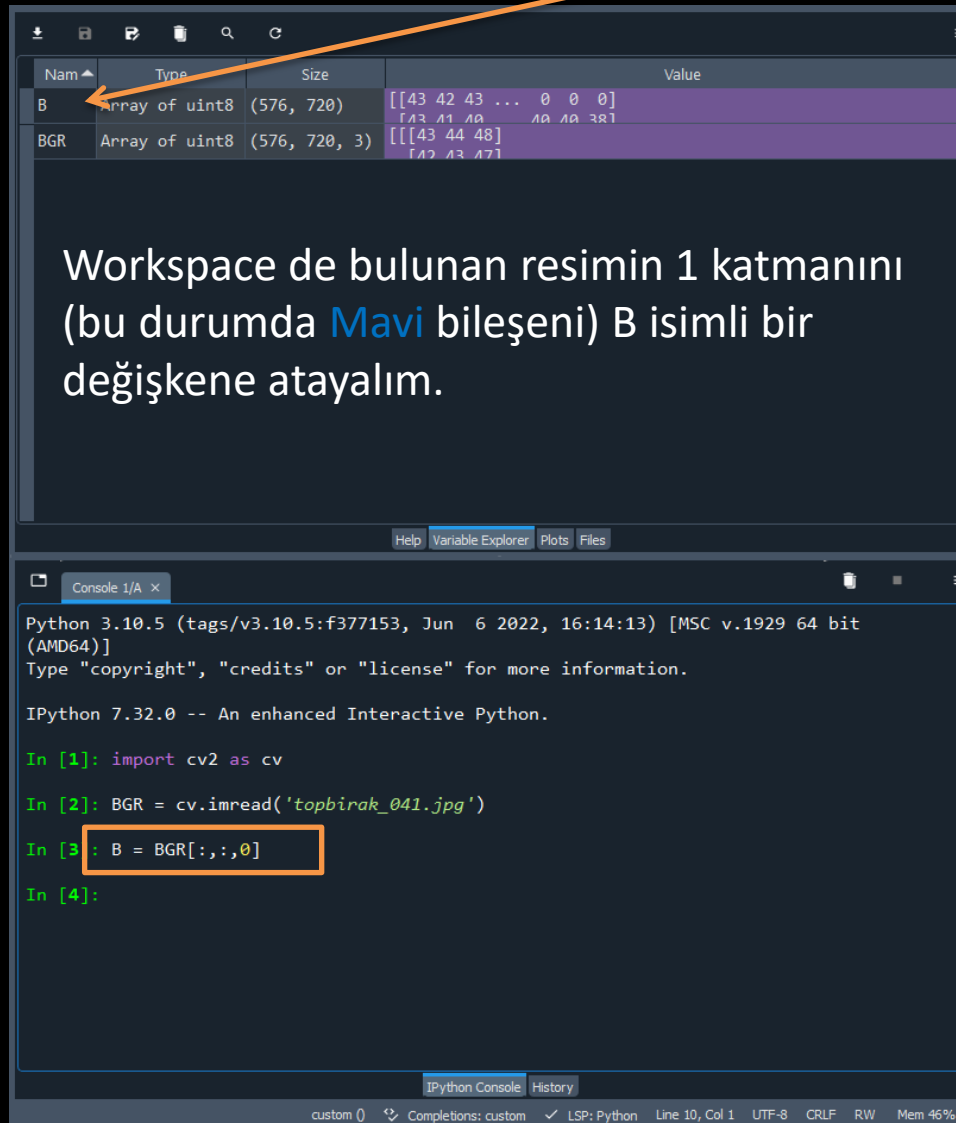


BGR üzerine çift tıklayın



$B = BGR[:, :, 1]$

B üzerine çift tıklayın



The screenshot shows a Jupyter Notebook interface. The top part displays the Variable Explorer with two variables: 'B' and 'BGR'. 'B' is an 'Array of uint8' with size '(576, 720)' and its value is shown as a 2D array of blue values. 'BGR' is an 'Array of uint8' with size '(576, 720, 3)' and its value is shown as a 3D array of blue, green, and red values. An orange arrow points from the text 'B üzerine çift tıklayın' to the 'B' variable in the Variable Explorer.

Workspace de bulunan resimin 1 katmanını (bu durumda **Mavi** bileşeni) B isimli bir değişkene atayalım.

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun 6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.32.0 -- An enhanced Interactive Python.

In [1]: import cv2 as cv
In [2]: BGR = cv.imread('topbirak_041.jpg')
In [3]: B = BGR[:, :, 0]
In [4]:
```



B - NumPy object array

	0	1	2	3	4	5	6	7	8
0	43	42	43	45	47	44	43	37	40
1	43	41	40	38	38	38	43	37	43
2	42	41	42	39	41	42	42	41	43
3	44	42	43	42	44	47	45	46	43
4	48	48	49	46	46	49	48	49	49
5	49	52	55	52	49	47	50	52	51
6	51	51	54	56	51	50	52	53	53
7	51	47	50	55	53	52	52	51	54
8	52	50	50	55	54	52	52	52	53
9	51	51	56	55	54	52	51	51	50
10	53	54	57	55	56	55	53	51	53
11	56	58	57	54	56	57	57	51	56
12	56	56	56	51	54	56	57	54	53
13	55	55	55	50	54	54	54	54	52

Format

Resize

☒ Background color

Save and Close

Close

## Resim göstermek için imshow

Nam	Type	Size	Value
B	Array of uint8	(576, 720)	[[43 42 43 ... 0 0 0] [43 41 40 ... 40 40 38]
BGR	Array of uint8	(576, 720, 3)	[[[43 44 48] [42 43 47]

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun 6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.32.0 -- An enhanced Interactive Python.

In [1]: import cv2 as cv

In [2]: BGR = cv.imread('topbirak_041.jpg')

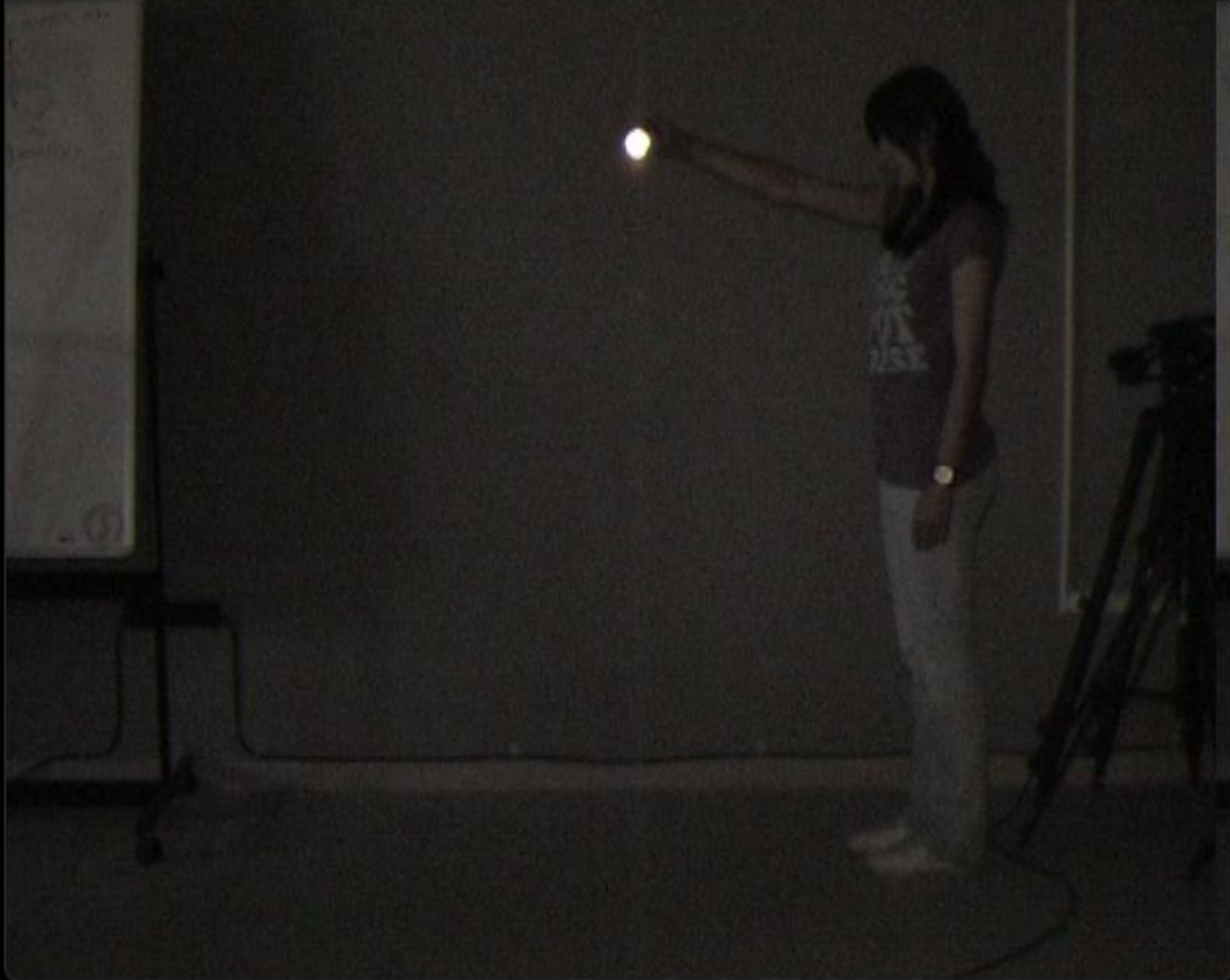
In [3]: B = BGR[:, :, 0]

In [4]: cv.imshow('Original image', BGR)

In [5]: cv.waitKey(0)
```



Original image



## Image Thresholding

[https://docs.opencv.org/4.9.0/d7/d4d/tutorial\\_py\\_thresholding.html](https://docs.opencv.org/4.9.0/d7/d4d/tutorial_py_thresholding.html)

If the pixel value is smaller than the threshold, it is set to 0, otherwise it is set to a maximum value. The function `cv.threshold` is used to apply the thresholding.

`cv.threshold(src, thresh, maxval, type[, dst]) -> retval, dst`

The first argument is the source image, which should be a **grayscale image**.

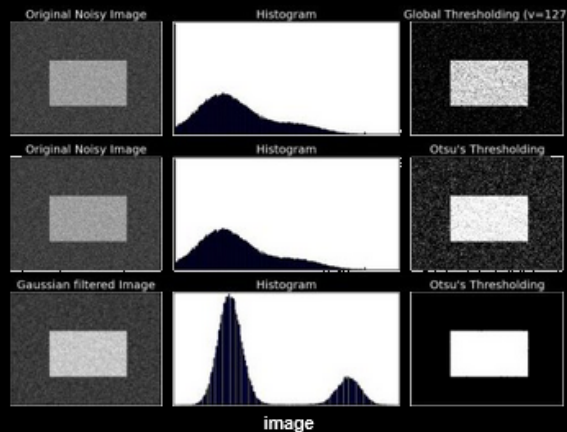
The second argument is the **threshold** value which is used to classify the pixel values.

The third argument is the maximum value which is assigned to pixel values exceeding the threshold.

OpenCV provides different types of thresholding which is given by the fourth parameter of the function.

## Image Thresholding

[https://docs.opencv.org/4.9.0/d7/d4d/tutorial\\_py\\_thresholding.html](https://docs.opencv.org/4.9.0/d7/d4d/tutorial_py_thresholding.html)



### How does Otsu's Binarization work?

This section demonstrates a Python implementation of Otsu's binarization to show how it actually works. If you are not interested, you can skip this.

Since we are working with bimodal images, Otsu's algorithm tries to find a threshold value ( $t$ ) which minimizes the **weighted within-class variance** given by the relation:

$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t)$$

where

$$q_1(t) = \sum_{i=1}^t P(i) \quad \& \quad q_2(t) = \sum_{i=t+1}^I P(i)$$

$$\mu_1(t) = \sum_{i=1}^t \frac{iP(i)}{q_1(t)} \quad \& \quad \mu_2(t) = \sum_{i=t+1}^I \frac{iP(i)}{q_2(t)}$$

$$\sigma_1^2(t) = \sum_{i=1}^t [i - \mu_1(t)]^2 \frac{P(i)}{q_1(t)} \quad \& \quad \sigma_2^2(t) = \sum_{i=t+1}^I [i - \mu_2(t)]^2 \frac{P(i)}{q_2(t)}$$

It actually finds a value of  $t$  which lies in between two peaks such that variances to both classes are minimal. It can be simply implemented in Python as follows:



## Image Thresholding

[https://docs.opencv.org/4.9.0/d7/d4d/tutorial\\_py\\_thresholding.html](https://docs.opencv.org/4.9.0/d7/d4d/tutorial_py_thresholding.html)

### A Threshold Selection Method from Gray-Level Histograms

NOBUYUKI OTSU

**Abstract**—A nonparametric and unsupervised method of automatic threshold selection for picture segmentation is presented. An optimal threshold is selected by the discriminant criterion, namely, so as to maximize the separability of the resultant classes in gray levels. The procedure is very simple, utilizing only the zeroth- and the first-order cumulative moments of the gray-level histogram. It is straightforward to extend the method to multithreshold problems. Several experimental results are also presented to support the validity of the method.

#### I. INTRODUCTION

It is important in picture processing to select an adequate threshold of gray level for extracting objects from their background. A variety of techniques have been proposed in this regard. In an ideal case, the histogram has a deep and sharp valley between two peaks representing objects and background, respectively, so that the threshold can be chosen at the bottom of this valley [1]. However, for most real pictures, it is often difficult to detect the valley bottom precisely, especially in such cases as when the valley is flat and broad, imbued with noise, or when the two peaks are extremely unequal in height, often producing no traceable valley. There have been some techniques proposed in order to overcome these difficulties. They are, for example, the valley sharpening technique [2], which restricts the histogram to the pixels with large absolute values of derivative (Laplacian or gradient), and the difference histogram method [3], which selects the threshold at the gray level with the maximal amount of difference. These utilize information concerning neighboring pixels (or edges) in the original picture to modify the histogram, so as to make it useful for

## Types of Thresholding

[https://docs.opencv.org/4.x/d7/d4d/tutorial\\_py\\_thresholding.html](https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html)

All simple thresholding types are:

`cv.THRESH_BINARY`

`cv.THRESH_BINARY_INV`

`cv.THRESH_TRUNC`

`cv.THRESH_TOZERO`

`cv.THRESH_TOZERO_INV`

The method returns two outputs.

The first is the threshold that was used and the second output is the thresholded image.

## Image Thresholding

[https://docs.opencv.org/4.x/d7/d4d/tutorial\\_py\\_thresholding.html](https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html)

```
import cv2 as cv
from matplotlib import pyplot as plt

BGR = cv.imread('topbirak_041.jpg')
imgray = cv.cvtColor(BGR, cv.COLOR_BGR2GRAY) # Convert to Gray

_, thresh1 = cv.threshold(imgray, 127, 255, cv.THRESH_BINARY)
_, thresh2 = cv.threshold(imgray, 127, 255, cv.THRESH_BINARY_INV)
_, thresh3 = cv.threshold(imgray, 127, 255, cv.THRESH_TRUNC)
_, thresh4 = cv.threshold(imgray, 127, 255, cv.THRESH_TOZERO)
_, thresh5 = cv.threshold(imgray, 127, 255, cv.THRESH_TOZERO_INV)

titles = ['Original Image', 'BINARY', 'BINARY_INV', 'TRUNC', 'TOZERO', 'TOZERO_INV']
images = [imgray, thresh1, thresh2, thresh3, thresh4, thresh5]

for i in range(6):
    plt.subplot(2, 3, i+1), plt.imshow(images[i], 'gray')
    plt.title(titles[i])
    plt.xticks([], plt.yticks([]))

plt.show()
```

### Exercise

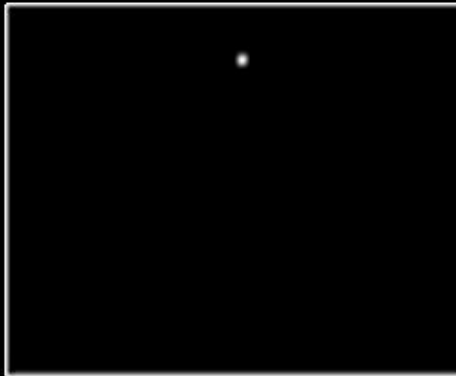
There are some optimizations available for Otsu's binarization. You can search and implement it.

## Image Thresholding

Original Image



BINARY



BINARY\_INV



TRUNC



TOZERO



TOZERO\_INV



```
cv.threshold(imgray, 127, 255, cv.THRESH_BINARY)
```

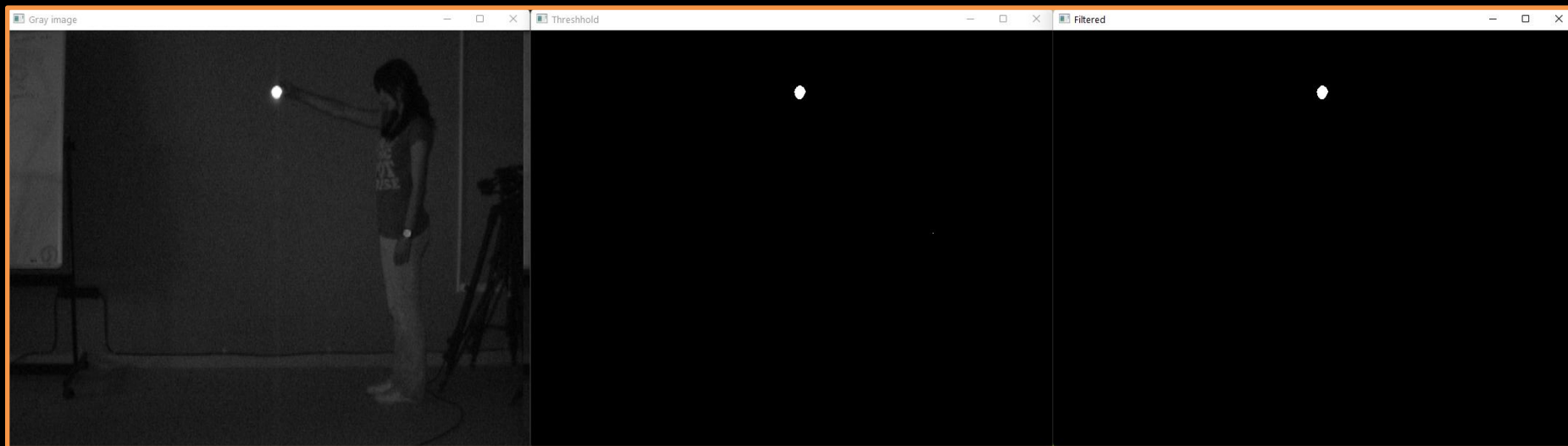
```
cv.imshow('Gray image', imgray)
```

```
cv.imshow('Threshold', thresh1)
```

```
# median filter salt&pepper
```

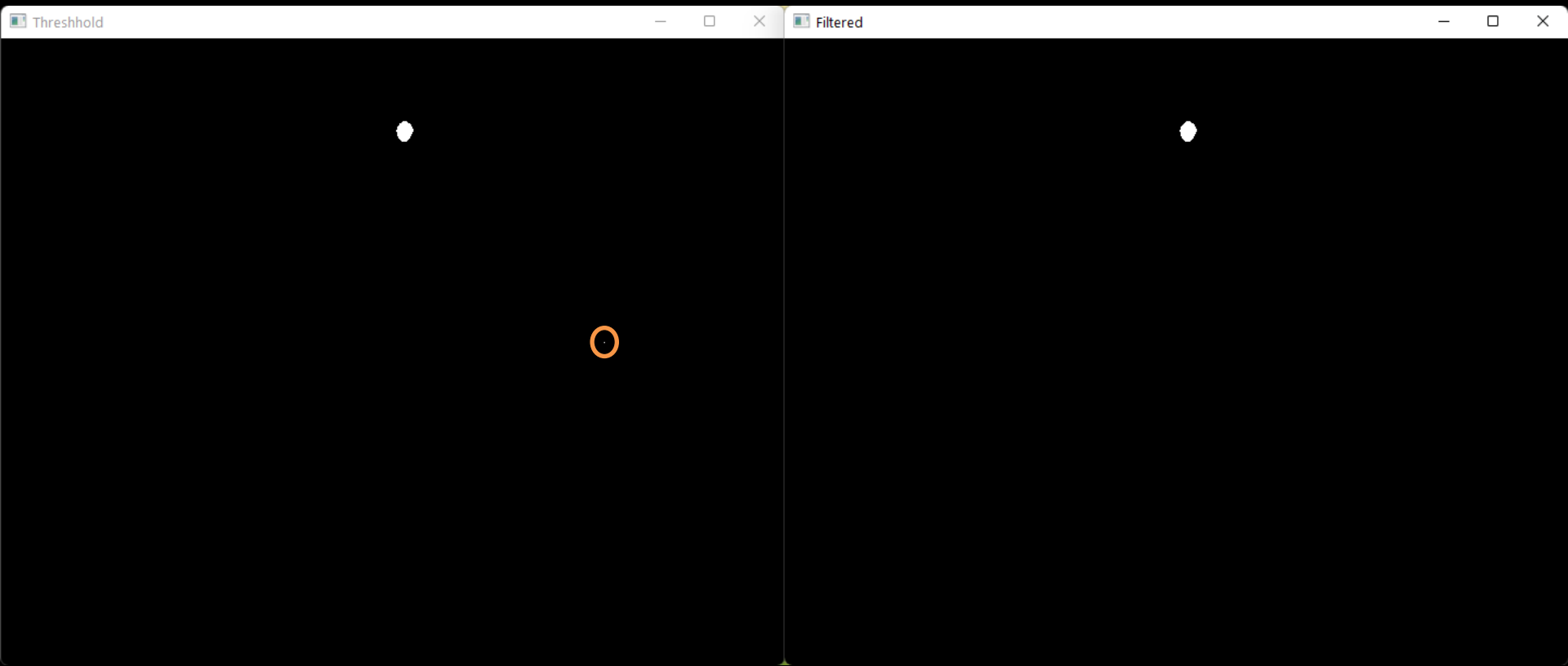
```
filtered = cv.medianBlur(thresh1, 3)
```

```
cv.imshow('Filtered', filtered)
```





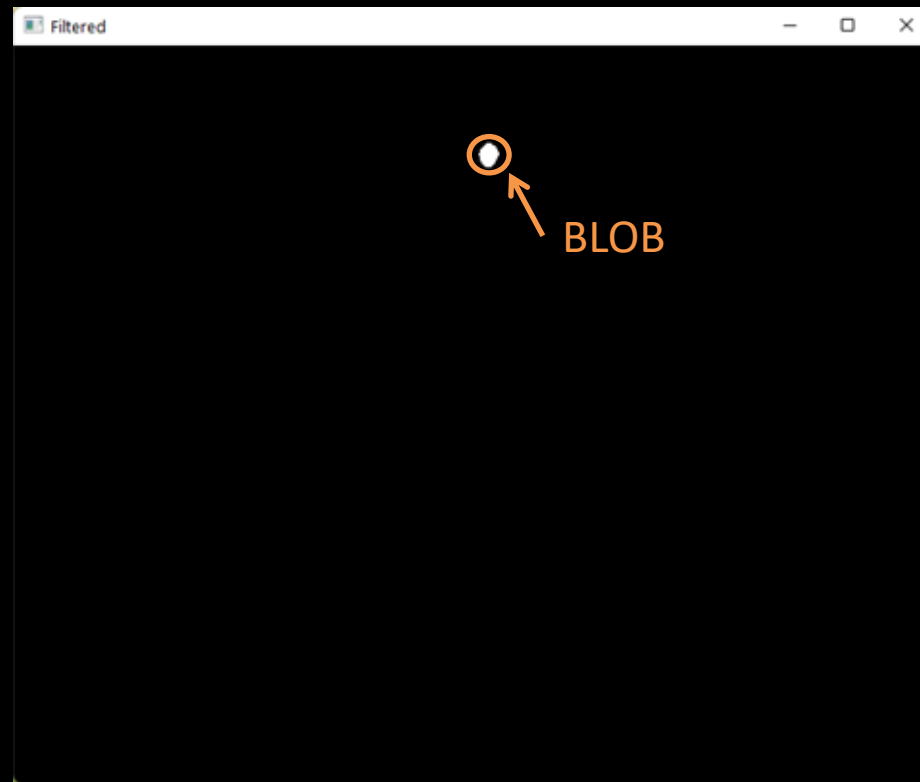
## FİLTRELEMELER GEREKLİ Mİ ?



## İkili Büyük Cismin (BLOB) Merkezini Bul (Sentroid)

### Binary Large Object

Matematik ve fizikte, 2 boyutlu bir şeklin ağırlık merkezi veya geometrik merkezi olarak tanımlanabilir. Bir şekil sentroidinden bir ipin ucuna asıldığında dengede kalır.



## Image Moments

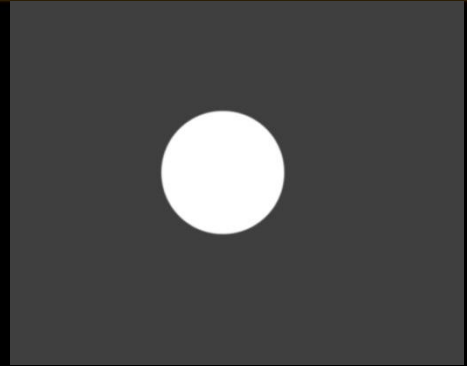
Image Moment is a particular weighted average of image pixel intensities, with the help of which we can find some specific properties of an image, like radius, area, centroid etc. To find the centroid of the image, we generally convert it to binary format and then find its center.

The **centroid** is given by the formula:-

$$C_x = \frac{M_{10}}{M_{00}} \quad C_y = \frac{M_{01}}{M_{00}}$$

$C_x$  is the x coordinate and  $C_y$  is the y coordinate of the centroid and  $M$  denotes the Moment

## Image Moments



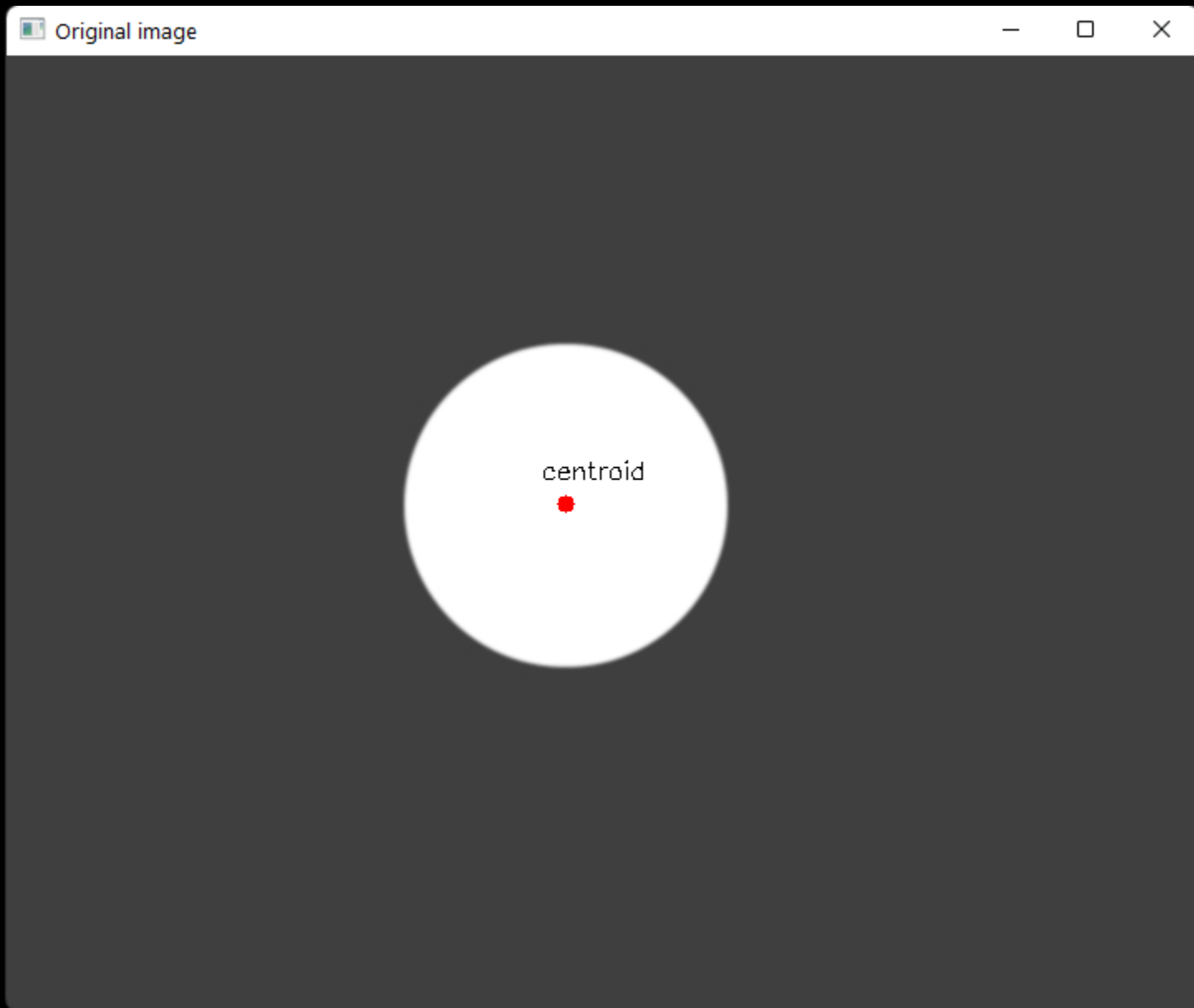
```
import cv2 as cv

BGR = cv.imread('moment.jpg')
imgray = cv.cvtColor(BGR, cv.COLOR_BGR2GRAY) # Convert to Gray

_, thresh1 = cv.threshold(imgray, 127, 255, cv.THRESH_BINARY)

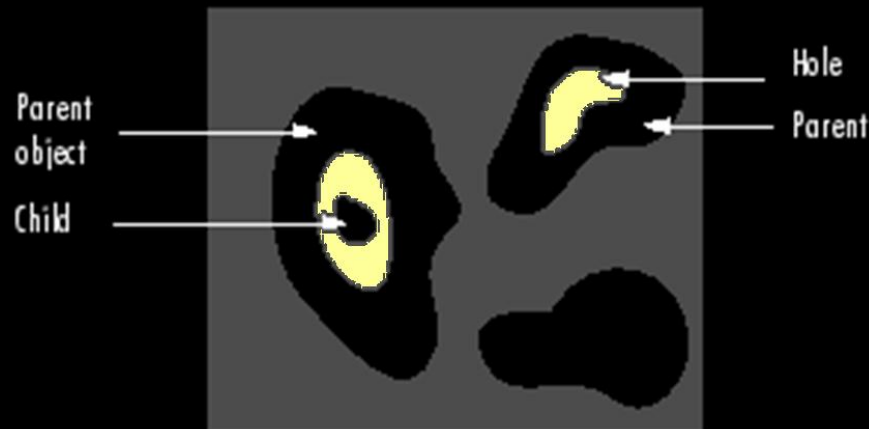
M = cv.moments(thresh1)
# calculate x,y coordinate of center
cX = int(M["m10"] / M["m00"])
cY = int(M["m01"] / M["m00"])

# put text and highlight the center
cv.circle(BGR, (cX, cY), 5, (0, 0, 255), -1)
cv.putText(BGR, "centroid", (cX-15, cY-15),
cv.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 1)
cv.imshow('Original image', BGR)
cv.waitKey(0)
```



## Yansıtıcı işaret yakalama

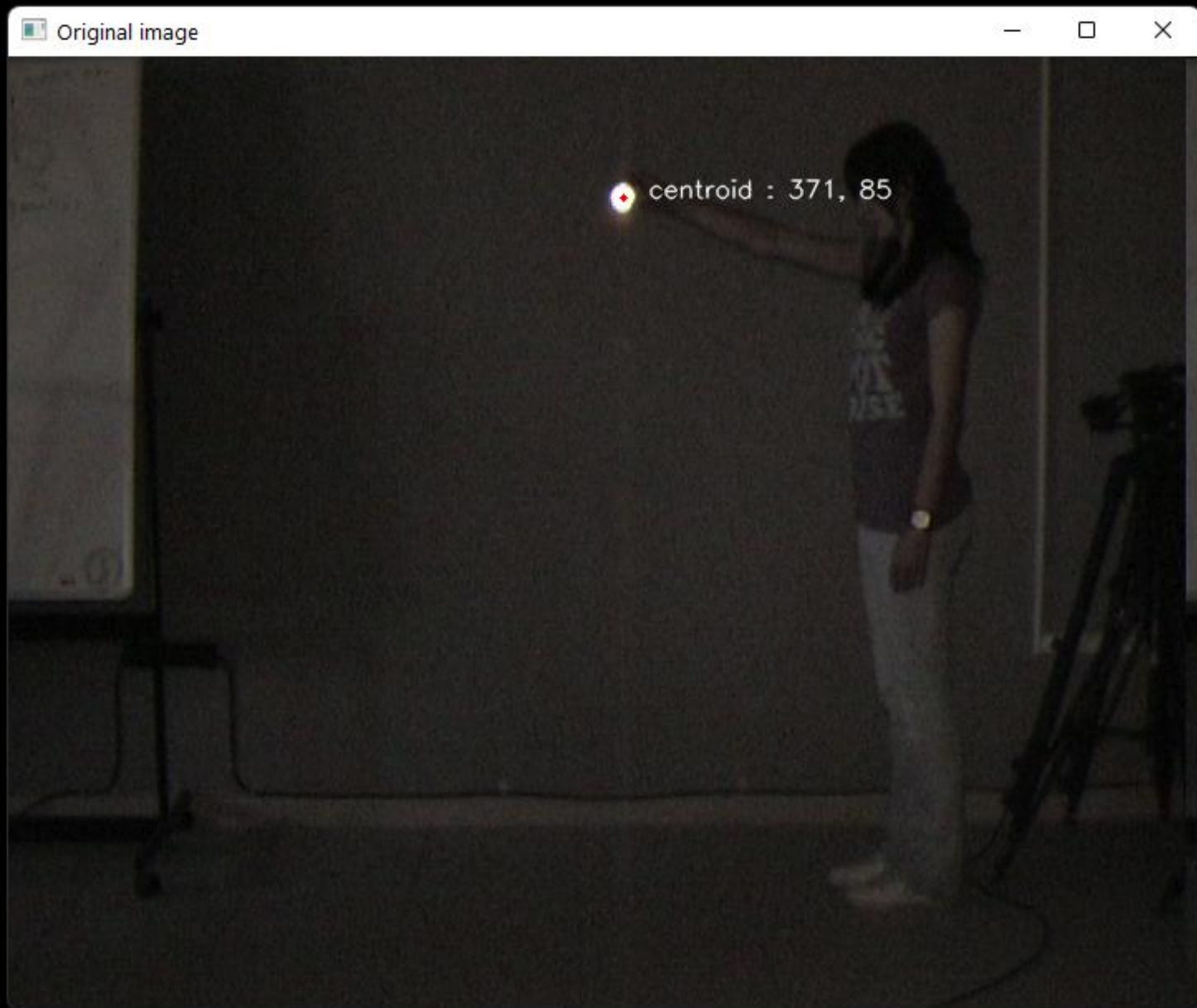
`cv.RETR_EXTERNAL`, it returns only extreme outer flags. All child contours are left behind.



```
## Contour Tracing
```

```
# cv.CHAIN_APPROX_SIMPLE for circle returns only one point
```

```
# cv.CHAIN_APPROX_NONE returns all contours
```





```
import cv2 as cv
import glob

img_array = []
for filename in glob.glob('vid*.jpg'):
    img = cv.imread(filename)
    height, width, layers = img.shape
    size = (width,height)
    img_array.append(img)

out = cv.VideoWriter('myVideo.avi',
cv.VideoWriter_fourcc(*'MJPG'), 15, size)

for i in range(len(img_array)):
    out.write(img_array[i])
out.release()
```



**Hafta2 Ödev: görüntüleri işleyerek video oluşturan bir program yazınız**





# Adaptive Thresholding

if an image has different lighting conditions in different areas. In that case, adaptive thresholding can help.

 **OpenCV** 4.13.0-dev ▾  
Open Source Computer Vision

[Main Page](#) | [Related Pages](#) | [Namespaces ▾](#) | [Classes ▾](#) | [Files ▾](#) | [Examples](#) | [Java documentation](#)

### ◆ adaptiveThreshold()

```
void cv::adaptiveThreshold ( InputArray src,  
                           OutputArray dst,  
                           double      maxValue,  
                           int          adaptiveMethod,  
                           int          thresholdType,  
                           int          blockSize,  
                           double       C )
```

**Python:**

```
cv.adaptiveThreshold( src, maxValue, adaptiveMethod, thresholdType, blockSize, C[, dst] ) -> dst
```

# Adaptive Thresholding

Original Image



Global Thresholding ( $\nu = 127$ )



Adaptive Mean Thresholding



Adaptive Gaussian Thresholding



# Adaptive Thresholding

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

img = cv.imread('sudoku.png', cv.IMREAD_GRAYSCALE)

img = cv.medianBlur(img,5)
ret,th1 = cv.threshold(img,127,255,cv.THRESH_BINARY)
th2 = cv.adaptiveThreshold(img,255,cv.ADAPTIVE_THRESH_MEAN_C,\
cv.THRESH_BINARY,11,2)

th3 = cv.adaptiveThreshold(img,255,cv.ADAPTIVE_THRESH_GAUSSIAN_C,\
cv.THRESH_BINARY,11,2)

titles = ['Original Image', 'Global Thresholding (v = 127)',
'Adaptive Mean Thresholding', 'Adaptive Gaussian Thresholding']

images = [img, th1, th2, th3]

for i in range(4):
    plt.subplot(2,2,i+1),plt.imshow(images[i],'gray')
    plt.title(titles[i])
    plt.xticks([],plt.yticks([]))
plt.show()
```



## Gelecek Hafta





## Gelecek Hafta

