



# BCA 607 Hareket Analizi Sistemleri

## Matlab ile Görüntü İşleme 4



**SERDAR ARITAN**

serdar.aritan@hacettepe.edu.tr

Biyomekanik Araştırma Grubu  
www.biomech.hacettepe.edu.tr  
Spor Bilimleri Fakültesi  
www.sbt.hacettepe.edu.tr  
Hacettepe Üniversitesi, Ankara, Türkiye  
www.hacettepe.edu.tr



# Yansıtıcı işaret yakalama

MARKER LOCATION HISTORY

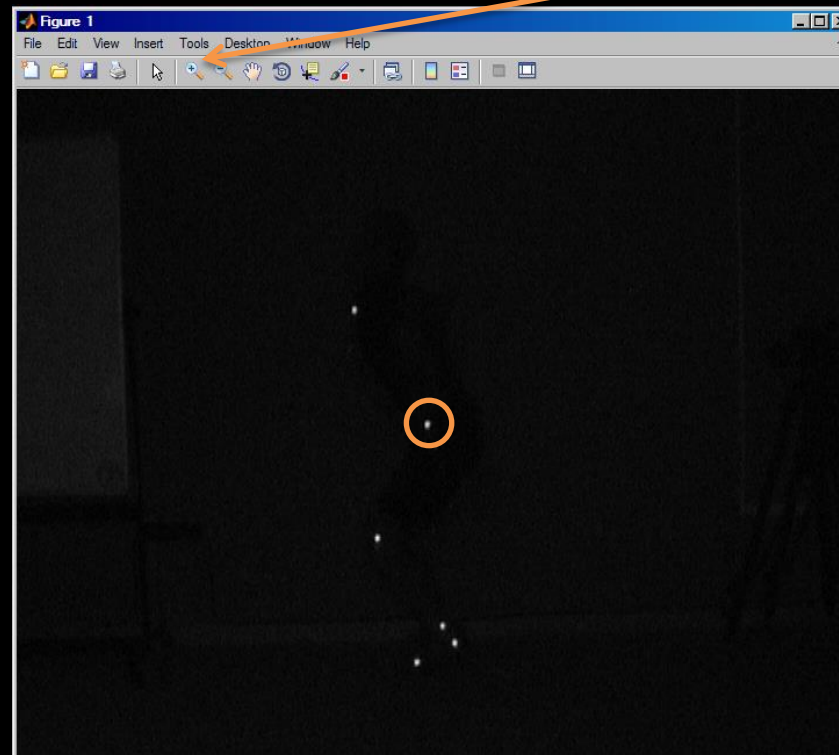


STANDARD SEGMENT MODE: GREEN LINE  
PRECISION MODE: ORANGE LINE



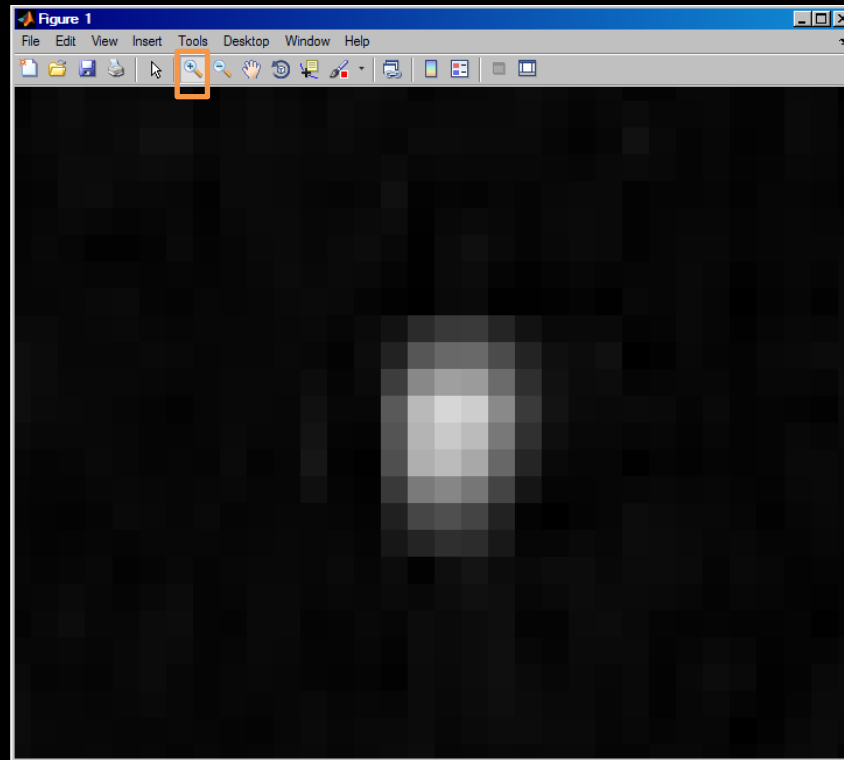
## Yansıtıcı işaret yakalama

```
RGB = imread('jump_106.jpg');  
I = rgb2gray(RGB);  
figure, imshow(I)
```



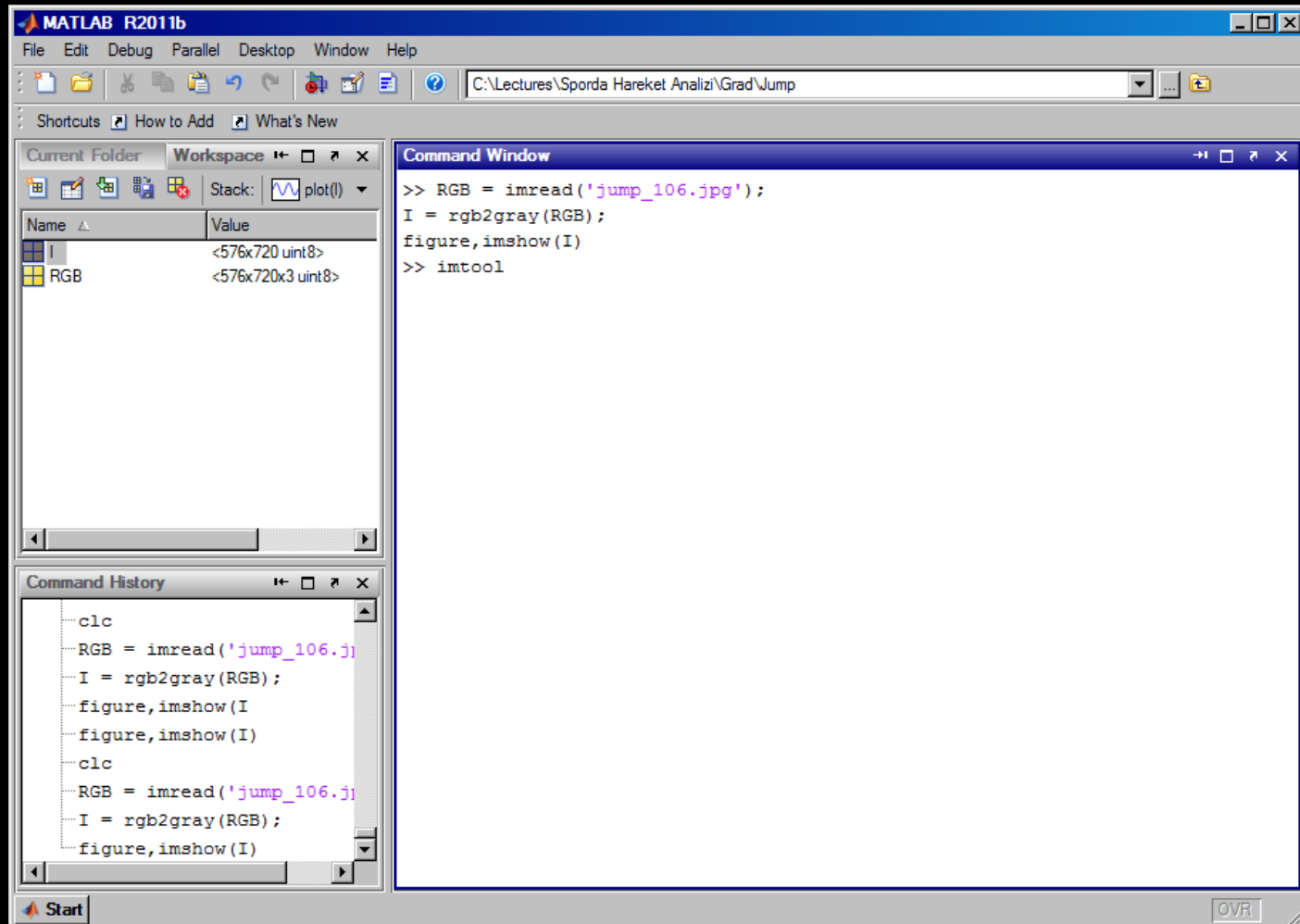
# Yansıtıcı işaret yakalama

Noktanın merkezi neresi ?



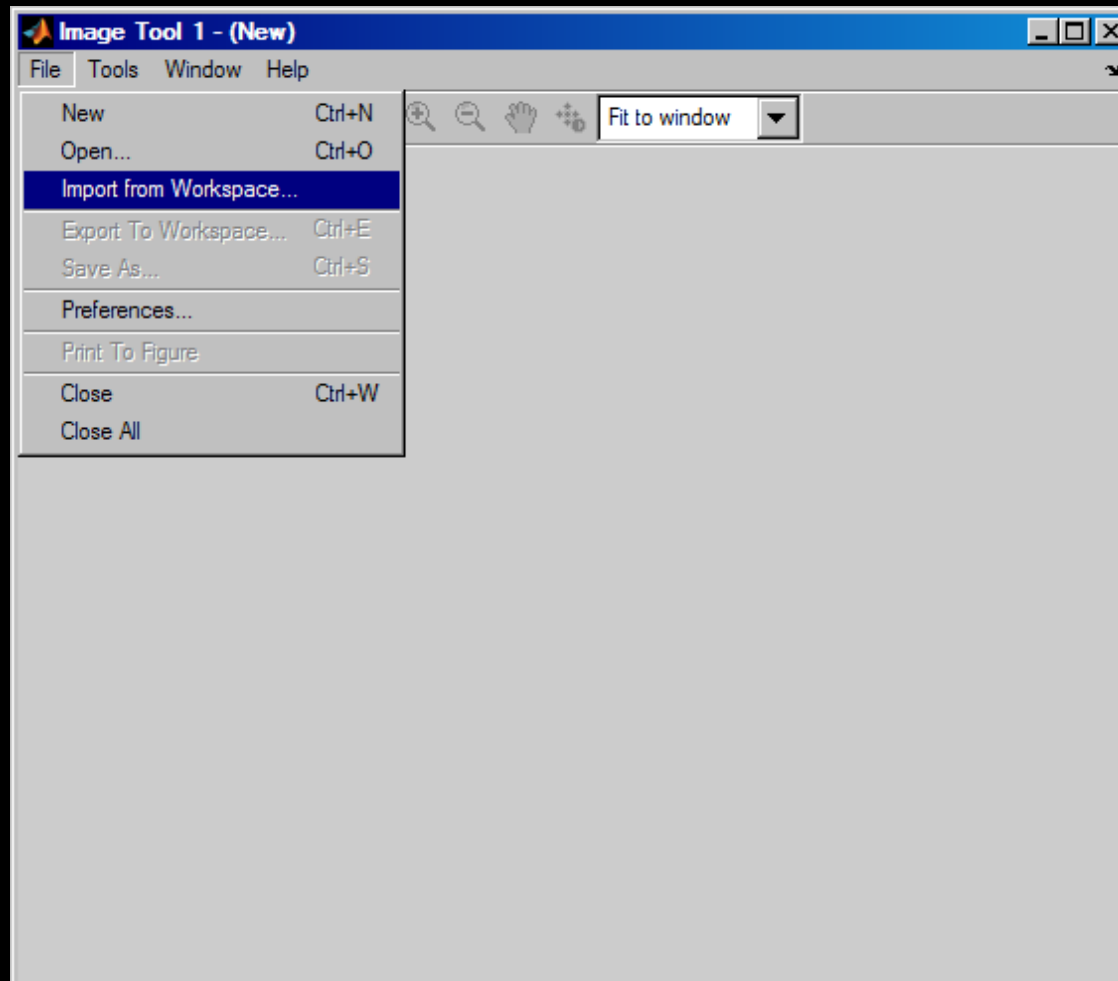
# Yansıtıcı işaret yakalama

## imtool kullanımı



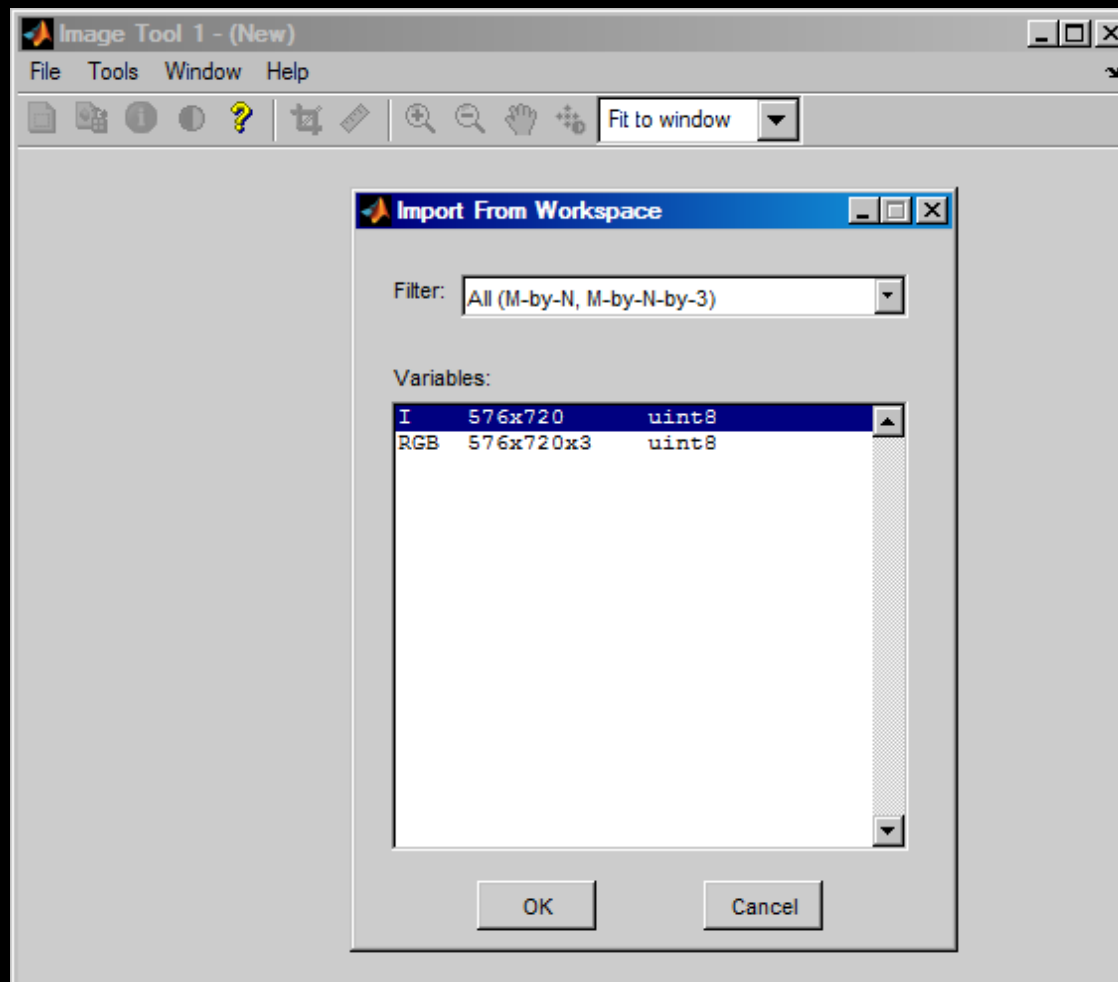
# Yansıtıcı işaret yakalama

## imtool kullanımı



# Yansıtıcı işaret yakalama

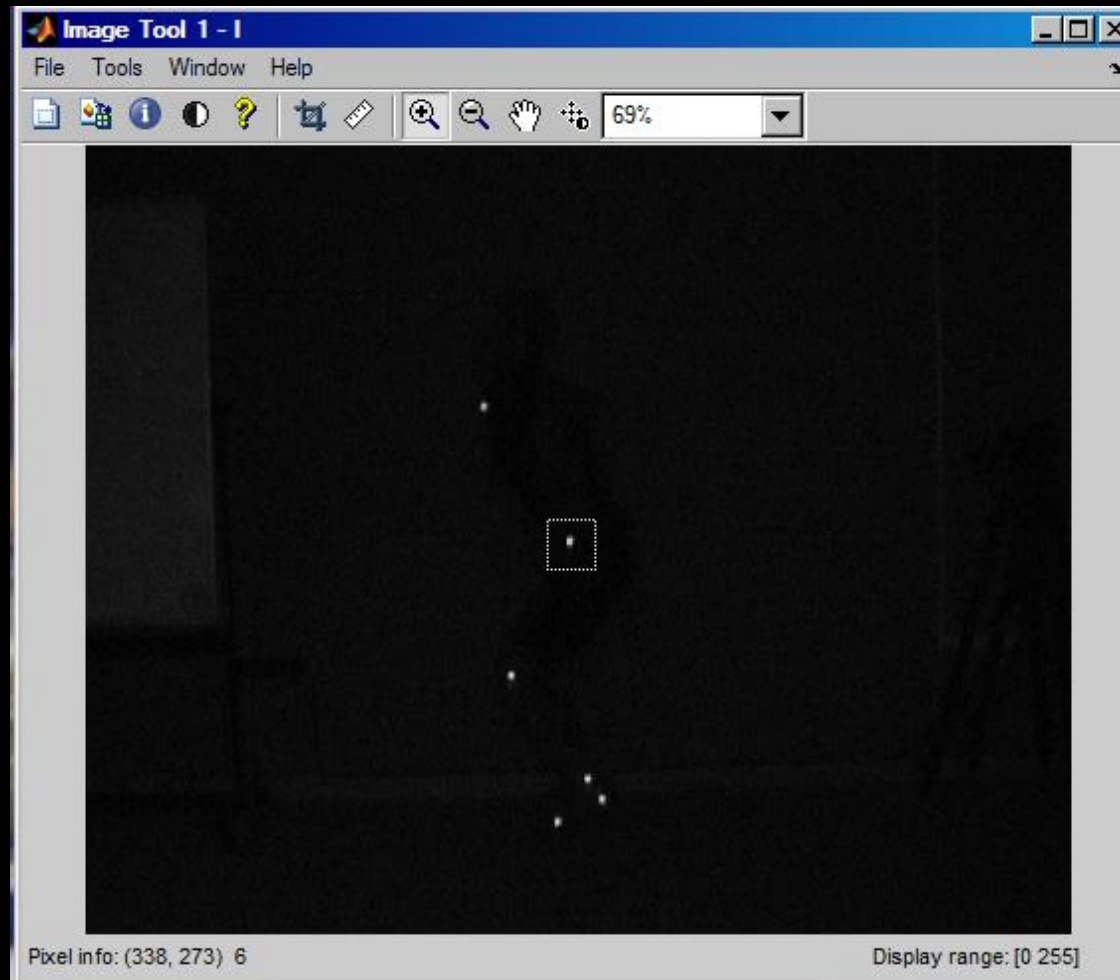
## imtool kullanımı





# Yansıtıcı işaret yakalama

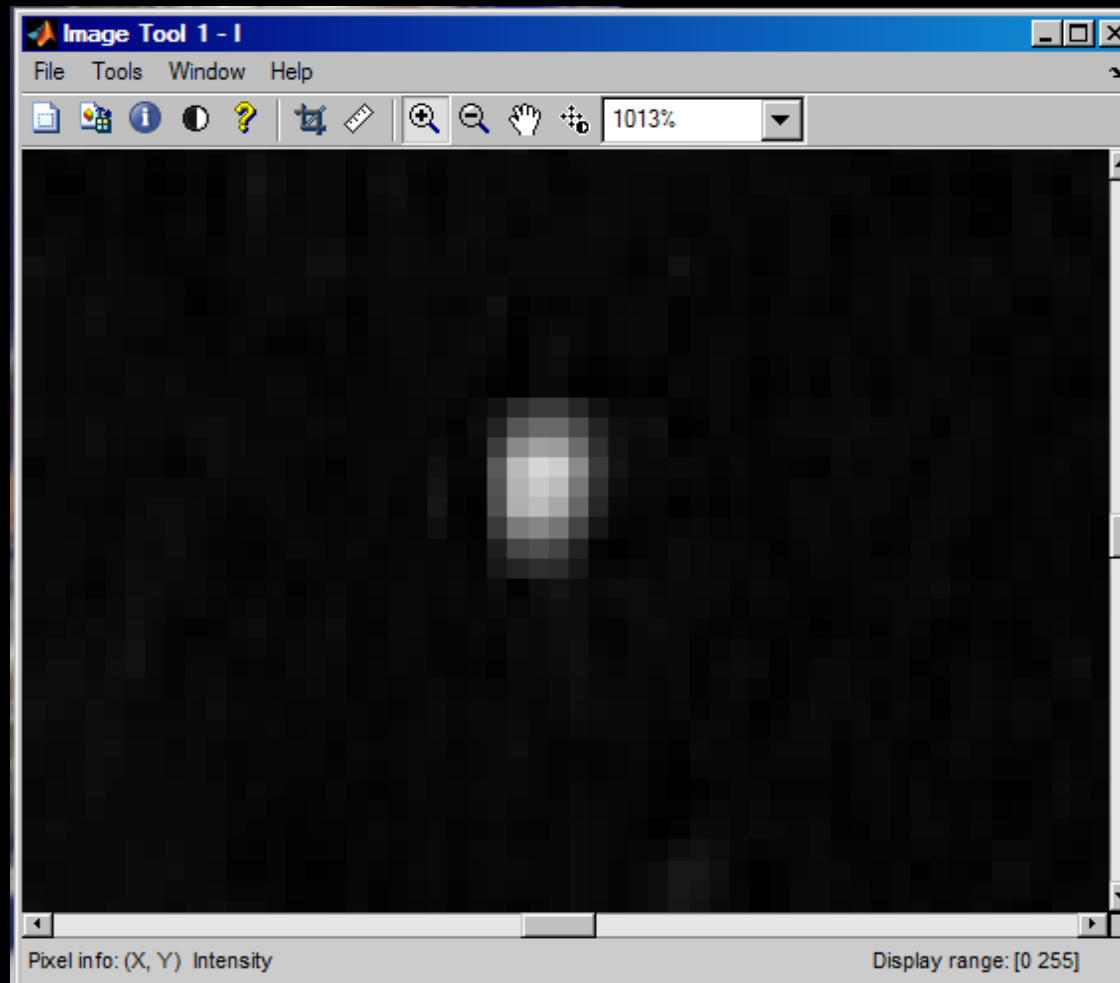
## imtool kullanımı





# Yansıtıcı işaret yakalama

## imtool kullanımı





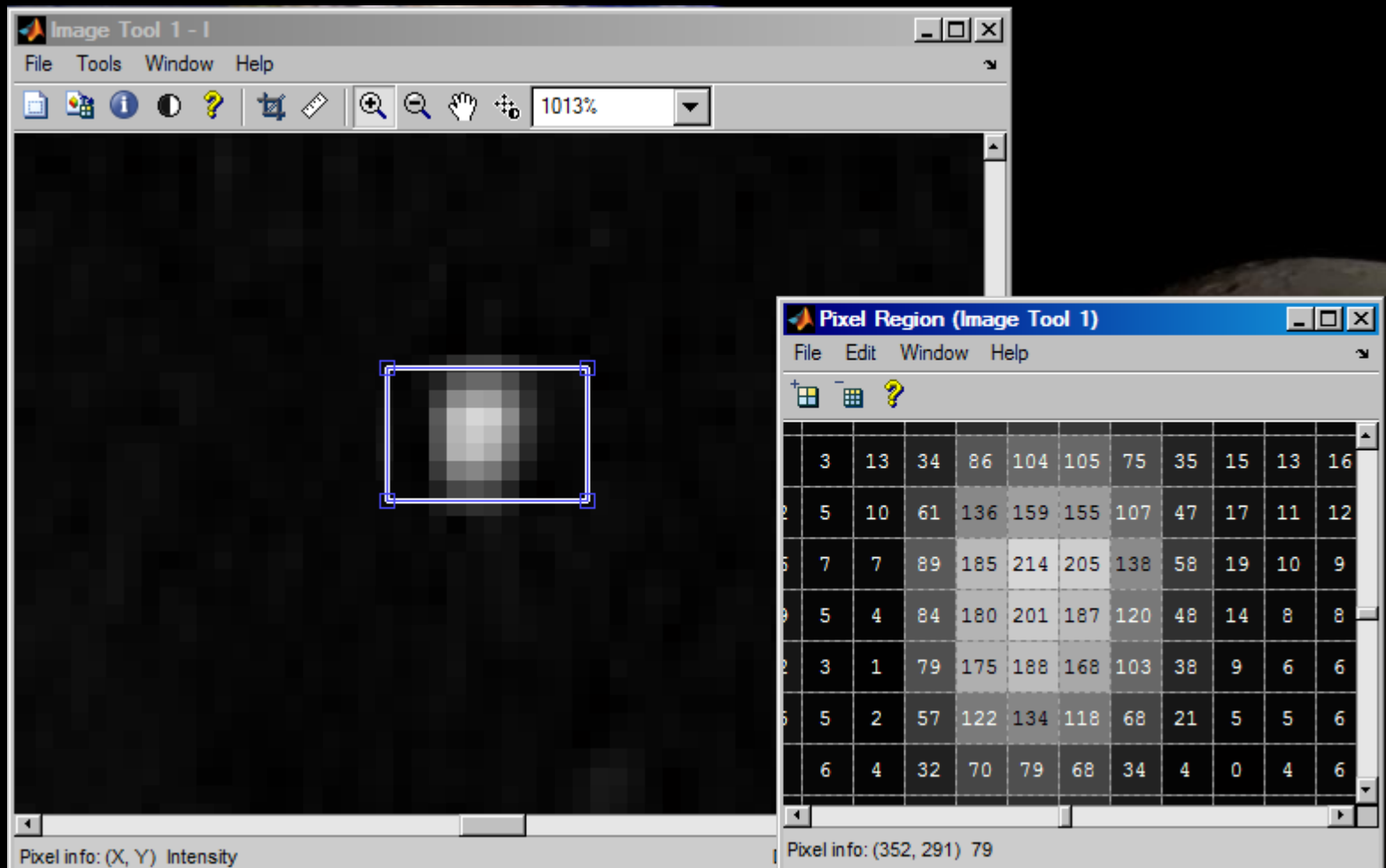
# Yansıtıcı işaret yakalama

imtool kullanımı

.

# Yansıtıcı işaret yakalama

## imtool kullanımı



The screenshot displays the 'Image Tool 1 - I' window with a grayscale image of a hand. A rectangular region of interest is selected, and the 'Pixel Region (Image Tool 1)' window is open, showing a table of pixel intensity values. The table has 11 columns and 8 rows. The status bar at the bottom of the 'Pixel Region' window indicates the current pixel is at (352, 291) with an intensity of 79.

	3	13	34	86	104	105	75	35	15	13	16
2	5	10	61	136	159	155	107	47	17	11	12
5	7	7	89	185	214	205	138	58	19	10	9
9	5	4	84	180	201	187	120	48	14	8	8
2	3	1	79	175	188	168	103	38	9	6	6
6	5	2	57	122	134	118	68	21	5	5	6
	6	4	32	70	79	68	34	4	0	4	6

Pixel info: (X, Y) Intensity

Pixel info: (352, 291) 79

# Yansıtıcı işaret yakalama

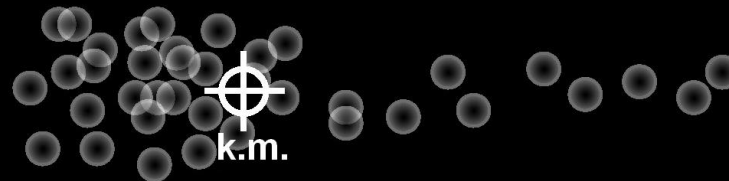
## kütle merkezi hesaplama

Kütle Merkezi : Bir cismin toplam kütesinin ortalama konumu



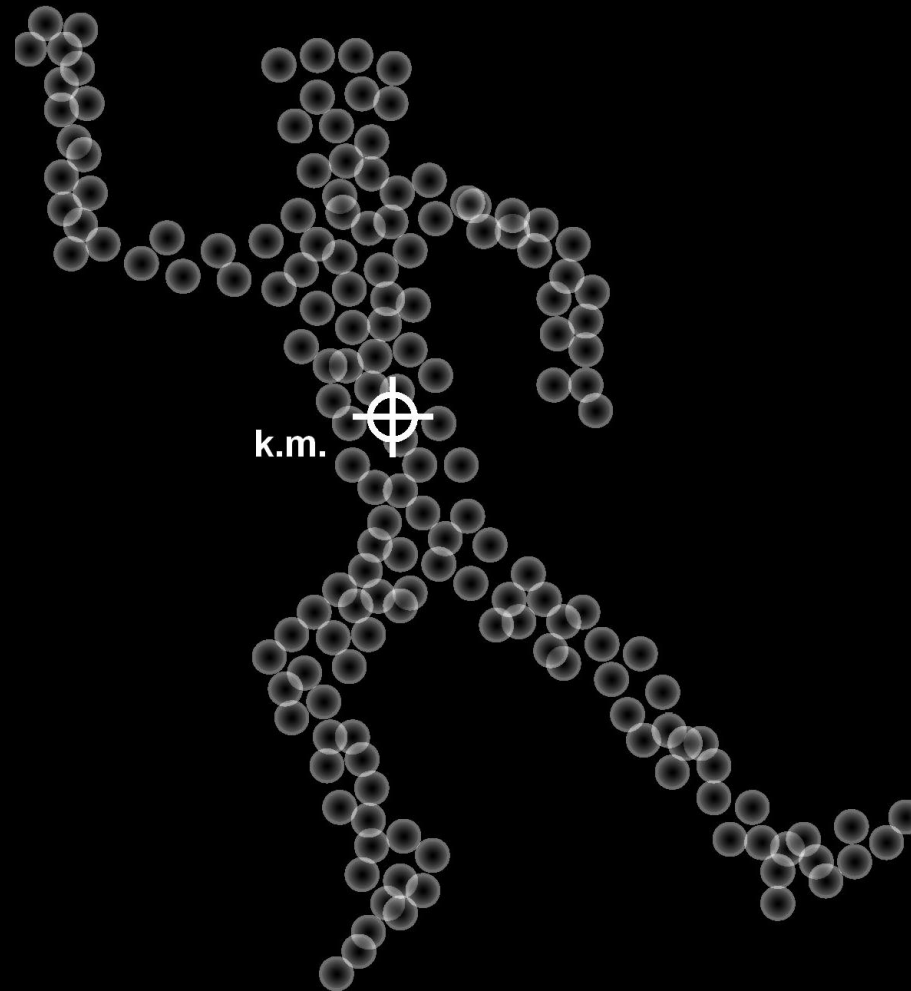
# Yansıtıcı işaret yakalama

kütle merkezi hesaplama



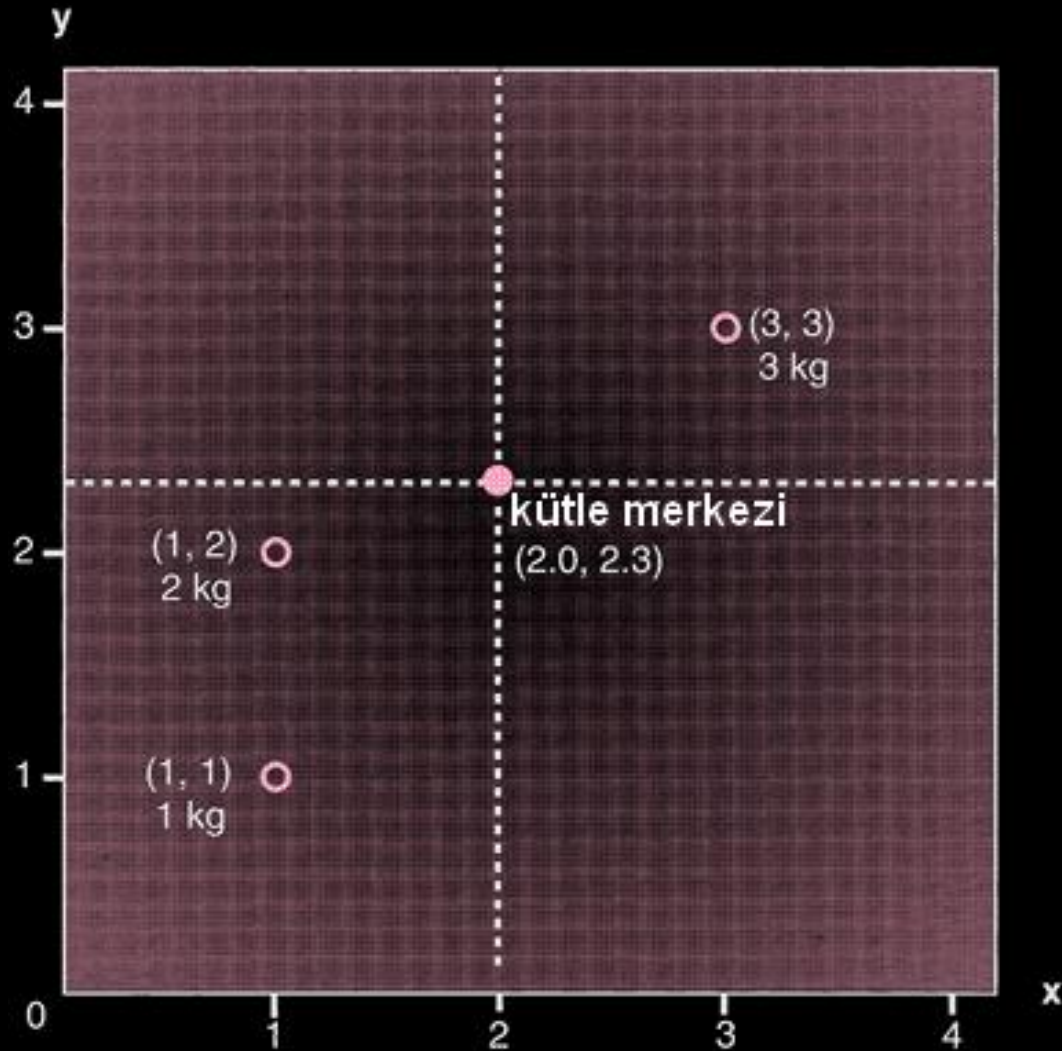
# Yansıtıcı işaret yakalama

kütle merkezi hesaplama



# Yansıtıcı işaret yakalama

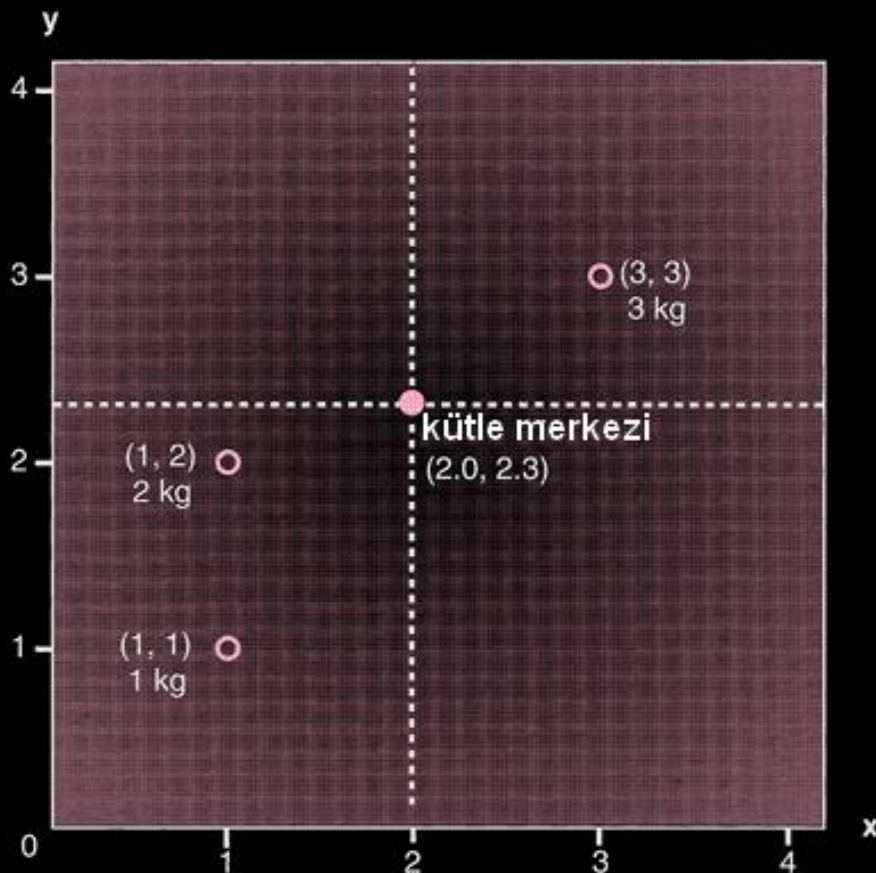
kütle merkezi hesaplama





# Yansıtıcı işaret yakalama

## kütle merkezi hesaplama



$$m_1gx_1+m_2gx_2+m_3gx_3= Mgx_{km}$$

$$m_1x_1+m_2x_2+m_3x_3= Mx_{km}$$

$$1.(1)+2.(1)+3.(3)= 6.x_{km}$$

$$x_{km}= 12 / 6 = 2$$

$$m_1y_1+m_2y_2+m_3y_3= My_{km}$$

$$1.(1)+2.(2)+3.(3)= 6.y_{km}$$

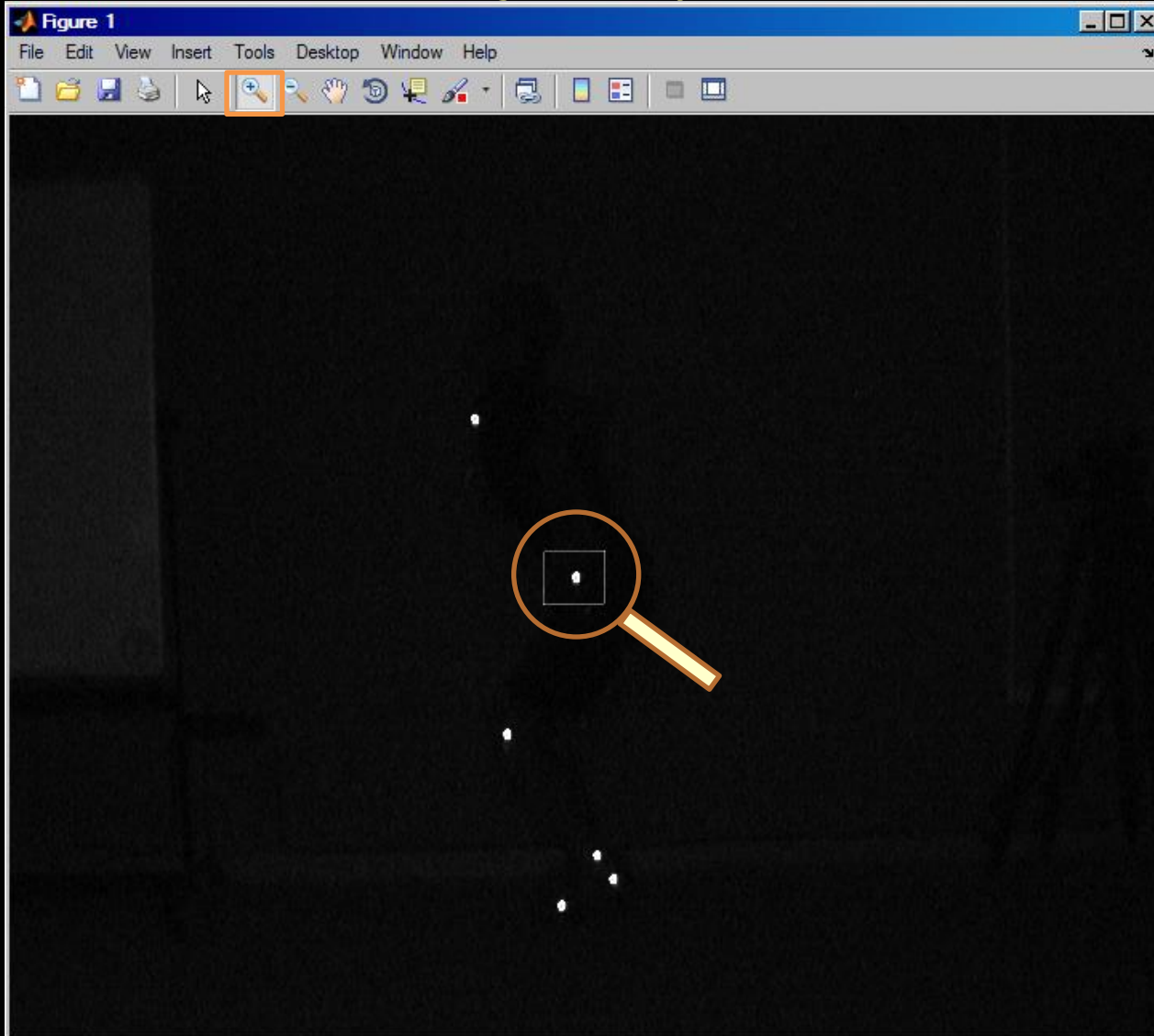
$$y_{km}= 14 / 6 = 2.3$$

# Yansıtıcı işaret yakalama

kütle merkezi hesaplama

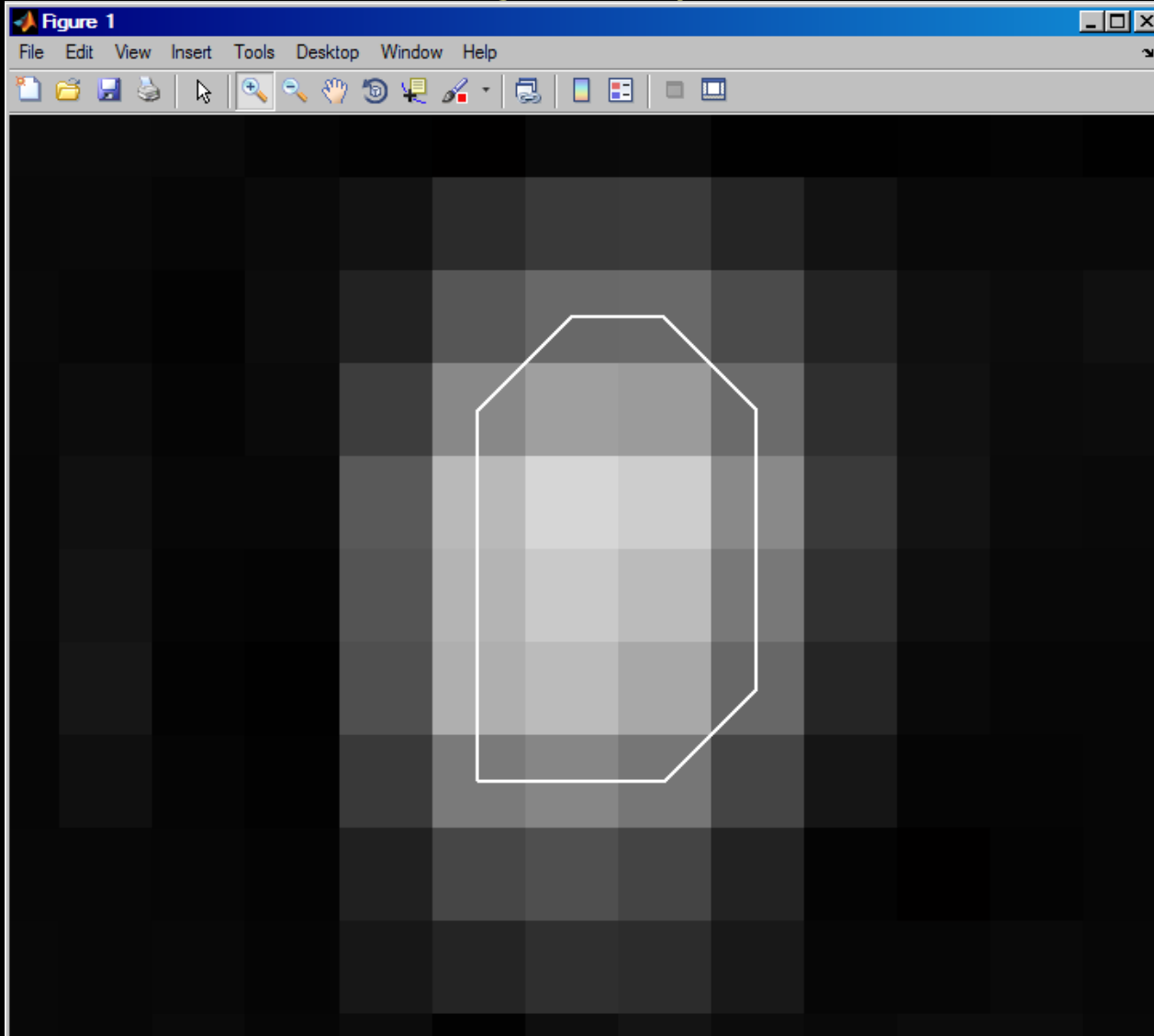
```
RGB = imread('jump_106.jpg');  
I = rgb2gray(RGB);  
figure,imshow(I), hold on;  
[level EM] = graythresh(I);  
bw = im2bw(I, EM);  
bw= medfilt2(bw,[3 3]);  
bw = bwareaopen(bw, 4);  
[B,L] = bwboundaries(bw,'noholes');  
for j = 1:length(B)  
    boundary = B{j};  
    plot(boundary(:,2),boundary(:,1),...  
        'w', 'LineWidth',2);  
end
```

# Yansıtıcı işaret yakalama





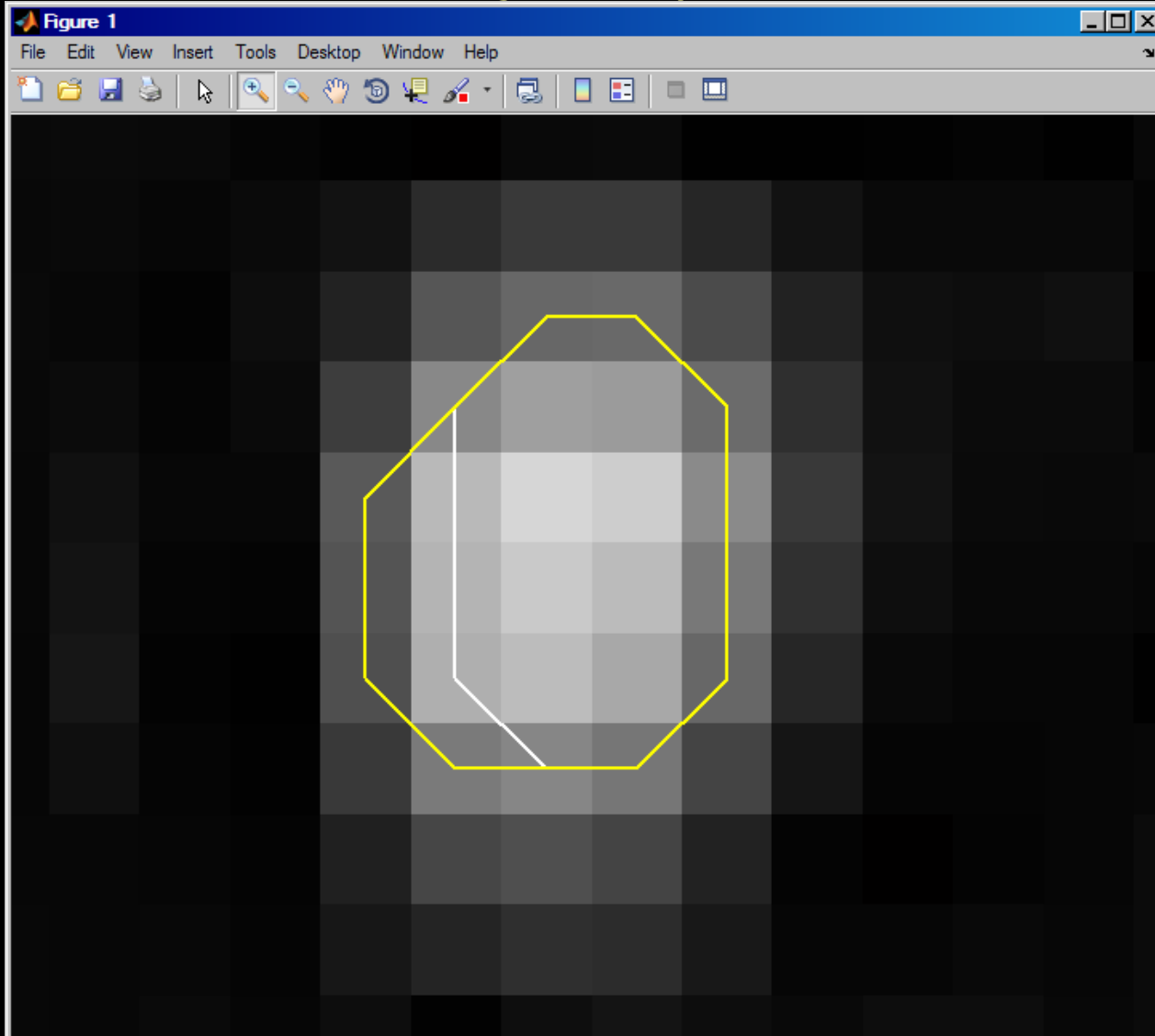
# Yansıtıcı işaret yakalama



## Yansıtıcı işaret yakalama

```
.  
.   
bw = im2bw(I, EM - 0.1);  
.   
.   
.   
for j = 1:length(B)  
    boundary = B{j};  
    plot(boundary(:,2),boundary(:,1),...  
        'y', 'LineWidth',2);  
end
```

# Yansıtıcı işaret yakalama

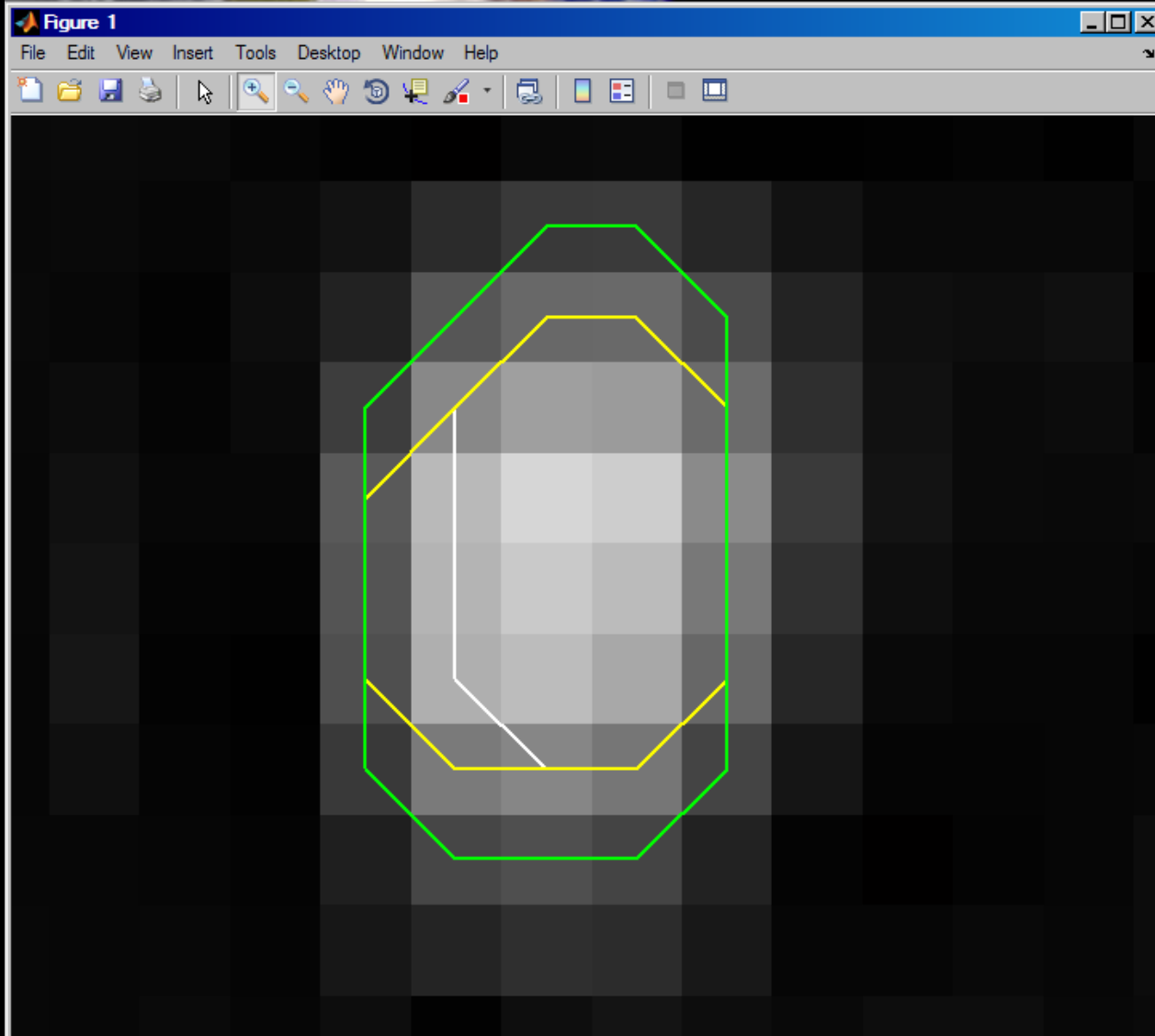


## Yansıtıcı işaret yakalama

```
.  
.   
bw = im2bw(I, EM - 0.2);  
.   
.   
.   
for j = 1:length(B)  
    boundary = B{j};  
    plot(boundary(:,2),boundary(:,1),...  
        'g', 'LineWidth',2);  
end
```



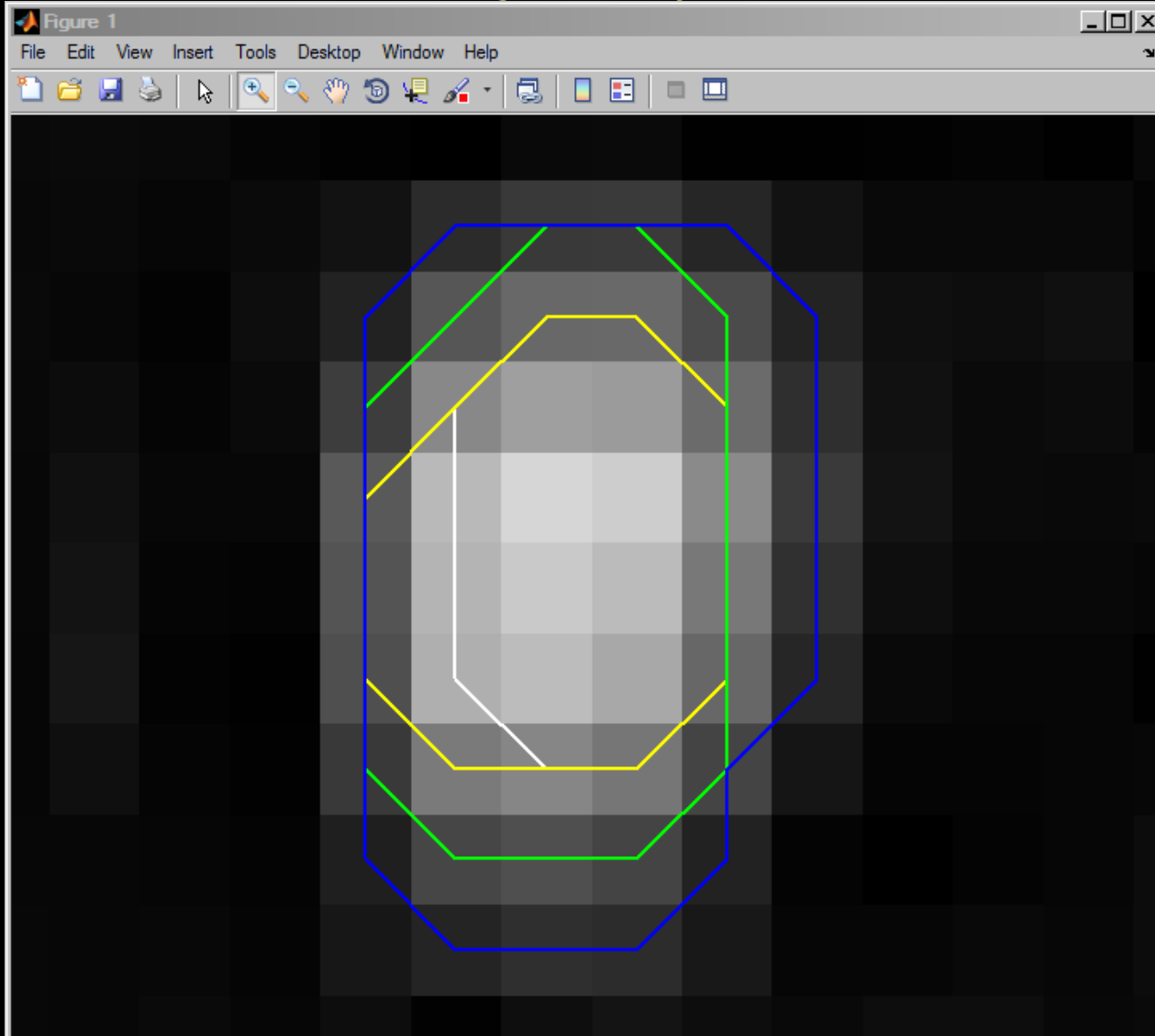
# Yansıtıcı işaret yakalama



## Yansıtıcı işaret yakalama

```
.  
.   
bw = im2bw(I, EM - 0.3);  
.   
.   
.   
for j = 1:length(B)  
    boundary = B{j};  
    plot(boundary(:,2),boundary(:,1),...  
        'b', 'LineWidth',2);  
end
```

# Yansıtıcı işaret yakalama



# Yansıtıcı işaret yakalama

## Matlab da kütle merkezi hesaplama

Matlab da kütle merkezini hesaplamak için **regionprops** kullanıyoruz

**regionprops**(L, I, 'Area', 'WeightedCentroid', 'Centroid', 'Perimeter');

Kütle merkezi hesabı için etiket matrisi ile birlikte gri resim bilgiside gerekiyor

# Yansıtıcı işaret yakalama

Matlab da kütle merkezi hesaplama

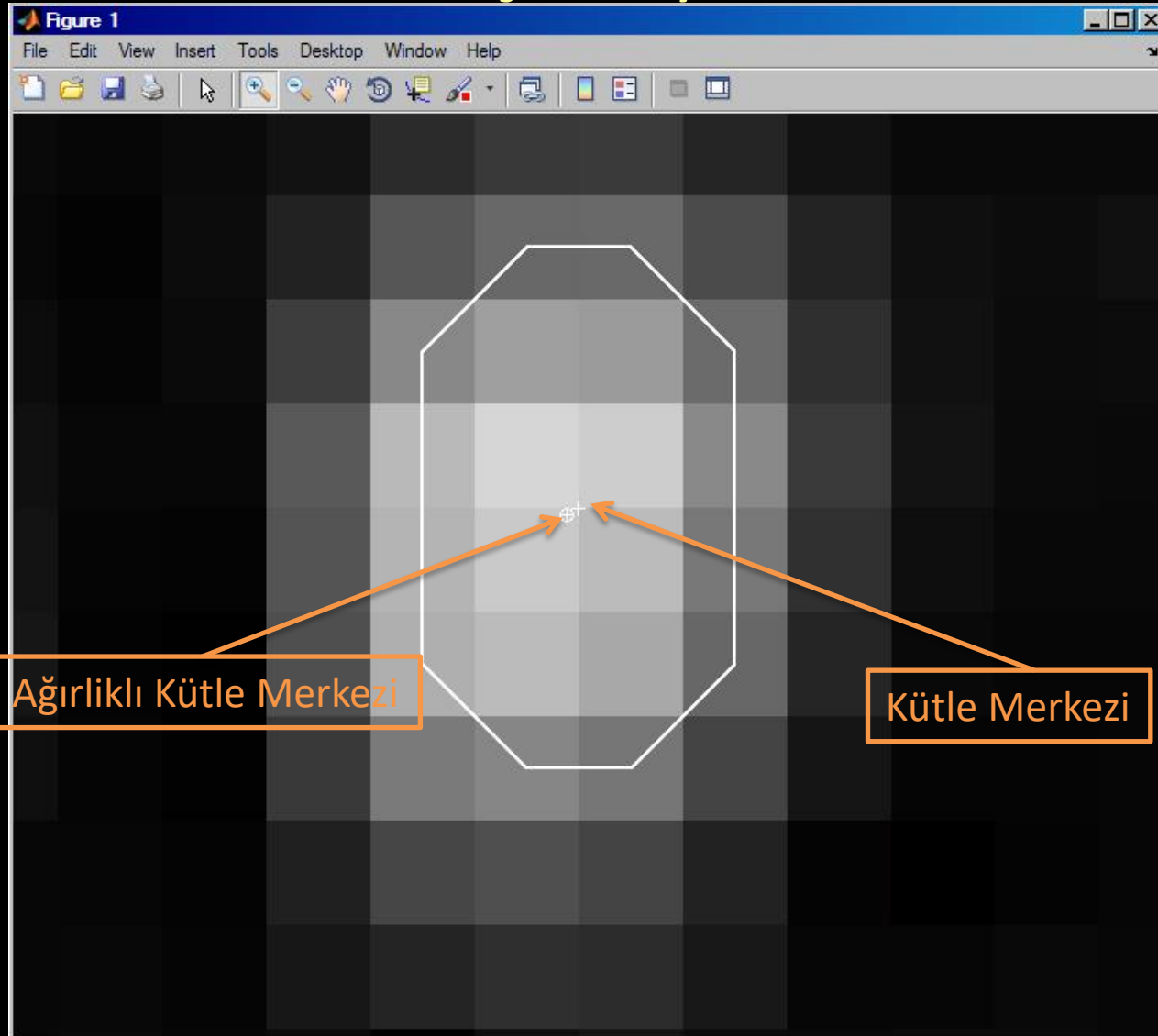
```
.  
.   
bw = im2bw(I, EM - 0.3);  
.   
.   
stats = regionprops(L, I, 'Area', ...  
                    'WeightedCentroid', ...  
                    'Centroid', 'Perimeter');  
for j = 1:length(B)  
.   
.   
.   
end
```

# Yansıtıcı işaret yakalama

## Matlab da kütle merkezi hesaplama

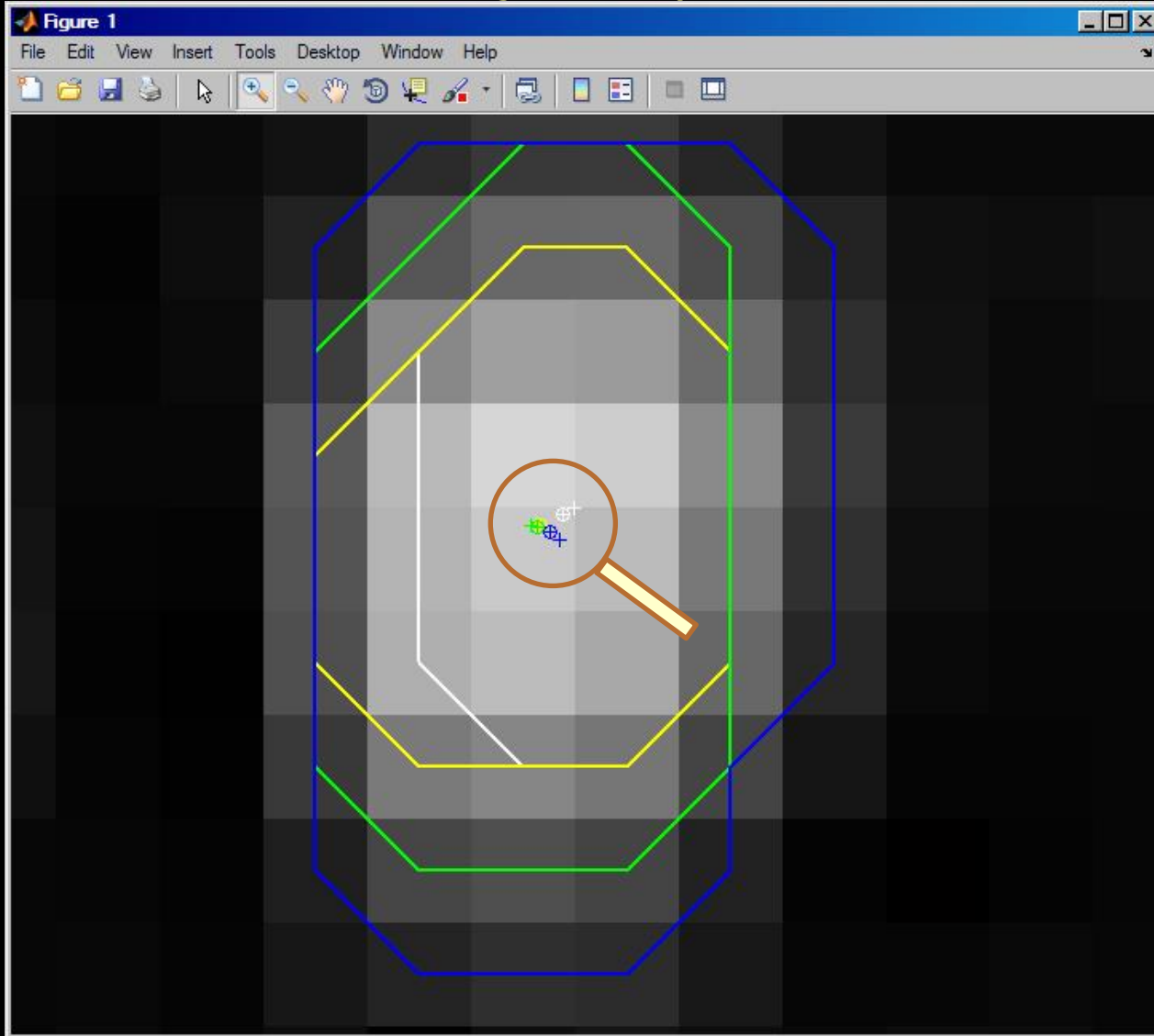
```
.  
.stats = regionprops(L, I, 'Area',  
'WeightedCentroid', 'Centroid', 'Perimeter');  
for j = 1:length(B)  
    boundary = B{j};  
    plot(boundary(:,2),boundary(:,1),'w', 'LineWidth',2);  
    centroid = stats(j).Centroid;  
    plot(centroid(1), centroid(2),'w+');  
    WeightedCentroid = stats(j).WeightedCentroid;  
    plot(WeightedCentroid(1), WeightedCentroid(2),'w+',  
        WeightedCentroid(1), WeightedCentroid(2),'wo');  
end
```

# Yansıtıcı işaret yakalama

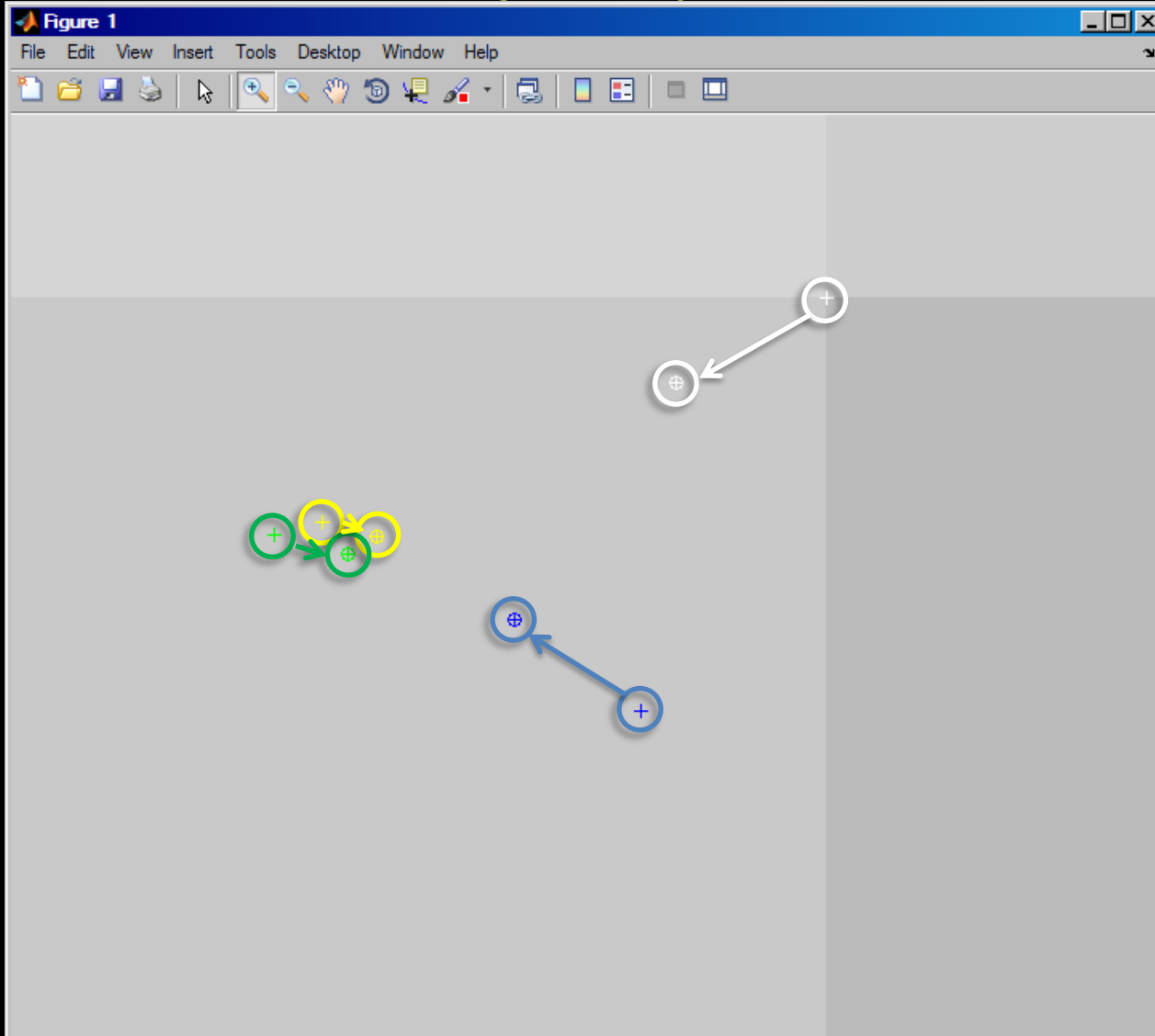




# Yansıtıcı işaret yakalama



# Yansıtıcı işaret yakalama

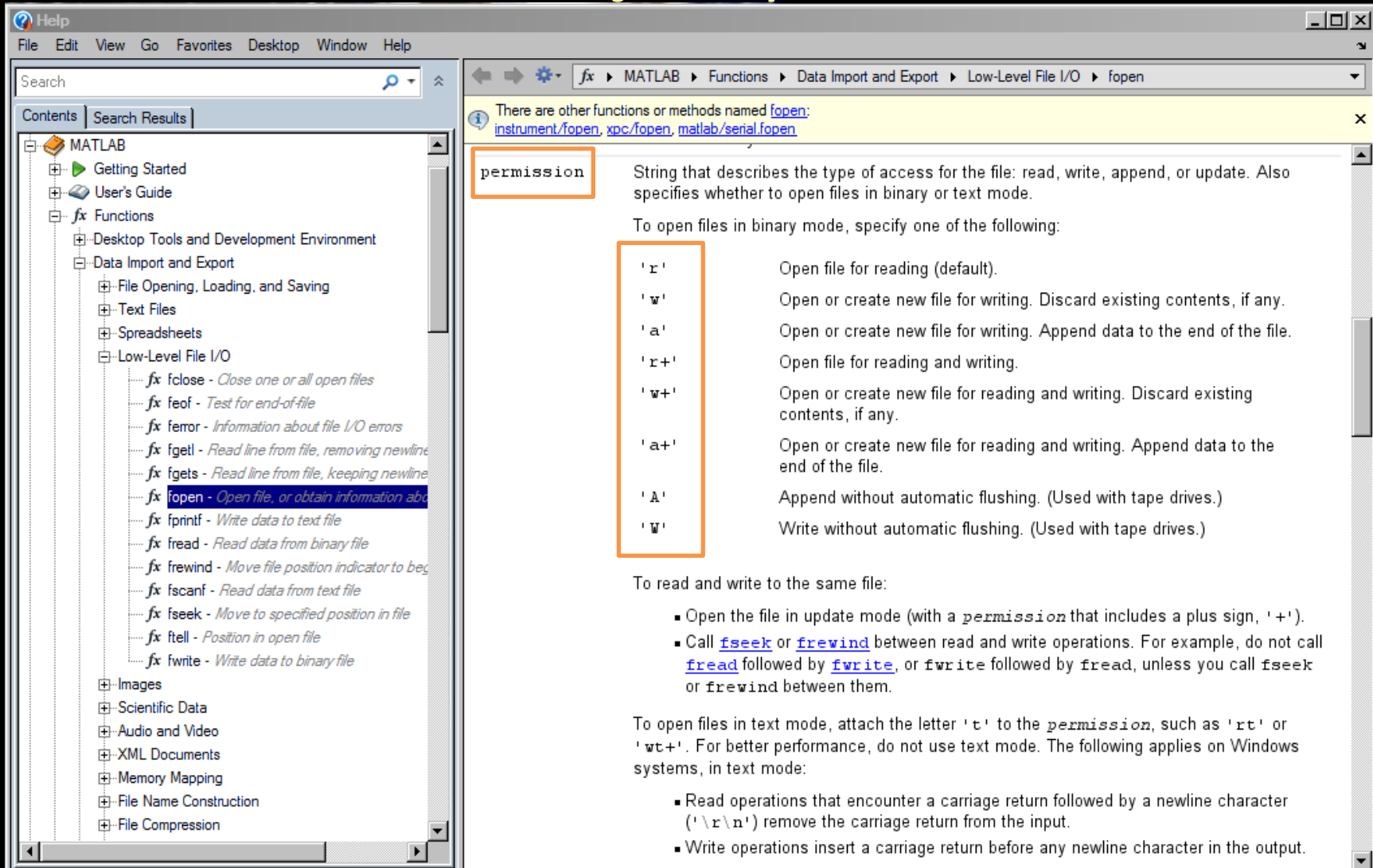


## Yansıtıcı işaret yakalama

Yansıtıcı işaret koordinatlarını dosyaya yazmak

```
RGB = imread('jump_106.jpg');  
% fopen ile dosya acip dosya ID sini belirle  
fid = fopen('jump_106.txt', 'wt');  
if fid < 0  
% eger dosya IDsi 0 dan kucukse dosya acilmamis  
% demek uyari mesajı verip programı sonlandır  
    warning('jump_106.txt dosyasi acilmadi!');  
    return;  
end  
.  
.  
.  
fclose(fid);
```

# Yansıtıcı işaret yakalama



Help

File Edit View Go Favorites Desktop Window Help

Search

Contents Search Results

MATLAB

- Getting Started
- User's Guide
- fx Functions
  - Desktop Tools and Development Environment
  - Data Import and Export
    - File Opening, Loading, and Saving
    - Text Files
    - Spreadsheets
    - Low-Level File I/O
      - fx fclose - Close one or all open files
      - fx feof - Test for end-of-file
      - fx ferror - Information about file I/O errors
      - fx fgetl - Read line from file, removing newline
      - fx fgets - Read line from file, keeping newline
      - fx fopen - Open file, or obtain information about file
      - fx fprintf - Write data to text file
      - fx fread - Read data from binary file
      - fx frewind - Move file position indicator to beginning
      - fx fscanf - Read data from text file
      - fx fseek - Move to specified position in file
      - fx ftell - Position in open file
      - fx fwrite - Write data to binary file
    - Images
    - Scientific Data
    - Audio and Video
    - XML Documents
    - Memory Mapping
    - File Name Construction
    - File Compression

There are other functions or methods named [fopen](#):  
[instrument/fopen](#), [xpc/fopen](#), [matlab/serial/fopen](#)

**permission** String that describes the type of access for the file: read, write, append, or update. Also specifies whether to open files in binary or text mode.

To open files in binary mode, specify one of the following:

'r'	Open file for reading (default).
'w'	Open or create new file for writing. Discard existing contents, if any.
'a'	Open or create new file for writing. Append data to the end of the file.
'r+'	Open file for reading and writing.
'w+'	Open or create new file for reading and writing. Discard existing contents, if any.
'a+'	Open or create new file for reading and writing. Append data to the end of the file.
'A'	Append without automatic flushing. (Used with tape drives.)
'w'	Write without automatic flushing. (Used with tape drives.)

To read and write to the same file:

- Open the file in update mode (with a *permission* that includes a plus sign, '+').
- Call [fseek](#) or [frewind](#) between read and write operations. For example, do not call [fread](#) followed by [fwrite](#), or [fwrite](#) followed by [fread](#), unless you call [fseek](#) or [frewind](#) between them.

To open files in text mode, attach the letter 't' to the *permission*, such as 'rt' or 'wt+'. For better performance, do not use text mode. The following applies on Windows systems, in text mode:

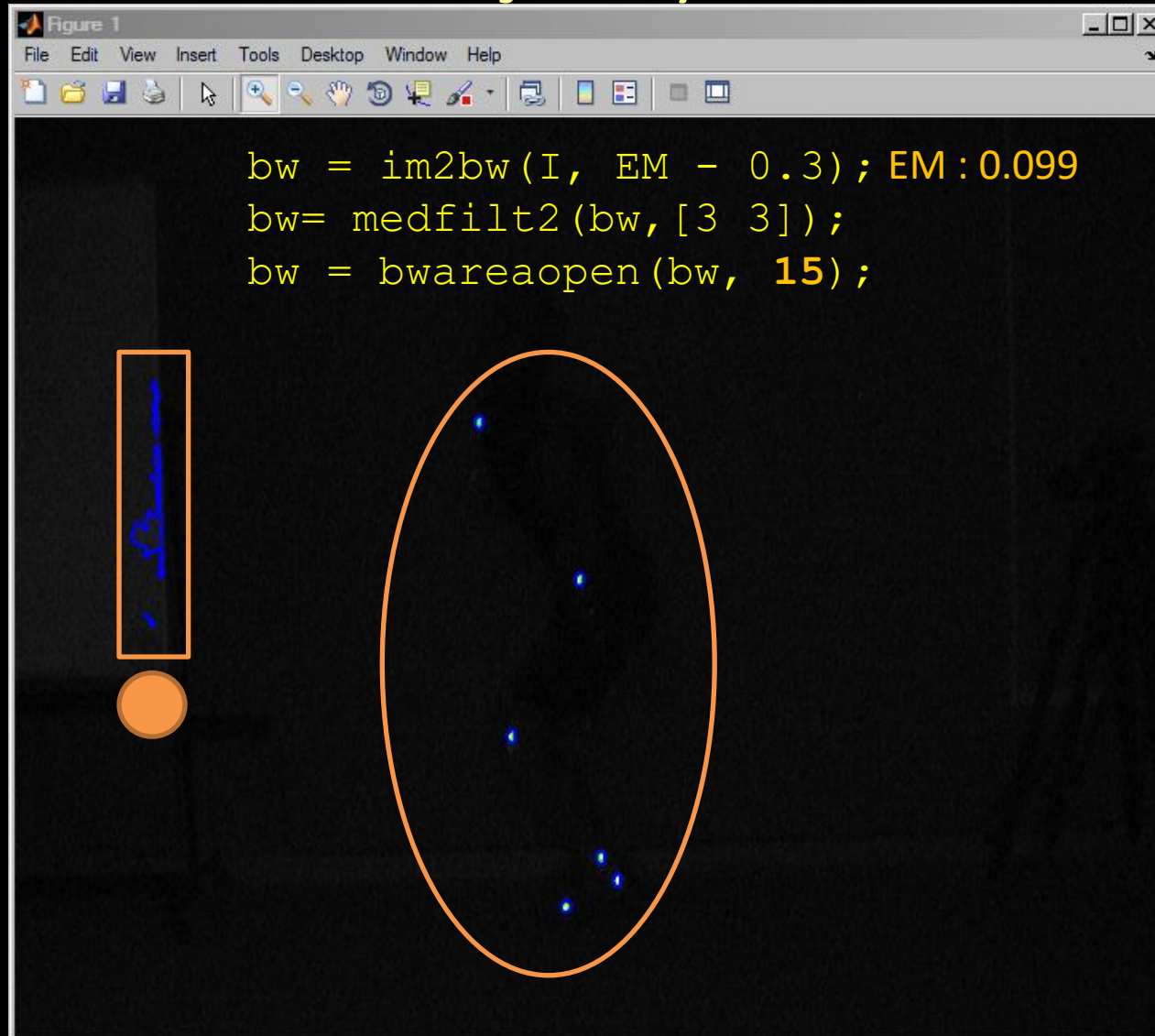
- Read operations that encounter a carriage return followed by a newline character ('\r\n') remove the carriage return from the input.
- Write operations insert a carriage return before any newline character in the output.

## Yansıtıcı işaret yakalama

Yansıtıcı işaret koordinatlarını dosyaya yazmak

```
stats = regionprops(L, I, 'Area',  
'WeightedCentroid', 'Centroid', 'Perimeter');  
for j = 1:length(B)  
  
    centroid = stats(j).Centroid;  
  
    WeightedCentroid = stats(j).WeightedCentroid;  
  
    fprintf(fid, '%d. nokta: %3.3f,%3.3f,%3.3f,%3.3f \n',  
            j, centroid(1), centroid(2), ...  
            WeightedCentroid(1), WeightedCentroid(2));  
end
```

## Yansıtıcı işaret yakalama







## Kinematik Analiz

Kinematik, fizik biliminin ilgi alanı olan mekaniğin bir alt dalıdır. Hareketin uzay-zaman (konum) özellikleri ile ilgilenir. Harekete neden olan kütle ve kuvvet gibi özellikler kinematiğin ilgi alanı değildir. Kinematik konum, hız ve ivmelenmeyi düzgün-doğrusal ve açısal olarak inceler. **Kinematik Analiz** hareketin oluşma nedenleri ile ilgilenmeksizin, hareketlerin konum ve zaman parametrelerinin incelenmesidir.



y

# Kinematik Analiz

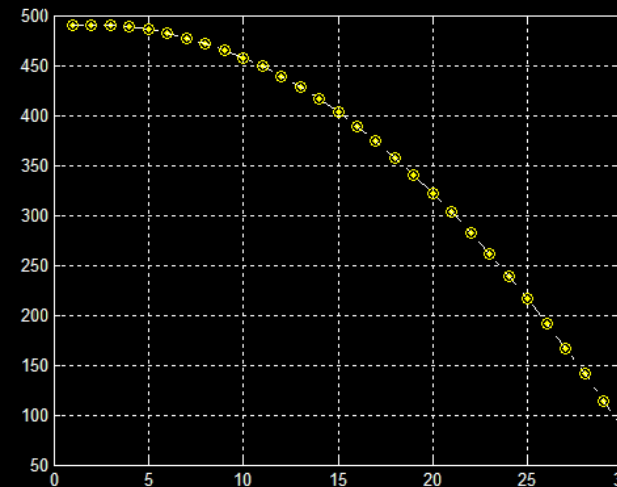


x



# Kinematik Analiz

1,	371.391,	489.972
2,	371.274,	490.200
3,	371.184,	489.558
4,	370.996,	488.202
5,	370.685,	485.621
6,	370.590,	482.226
7,	370.302,	477.591
8,	370.060,	472.092
9,	369.895,	465.504
10,	369.664,	457.775
11,	369.477,	448.911
12,	369.226,	439.051
13,	369.059,	428.146
14,	368.886,	416.360
15,	368.709,	403.166
16,	368.473,	389.203
17,	368.265,	373.938
18,	368.125,	357.829
19,	367.968,	340.511
20,	367.885,	322.318
21,	367.657,	302.889
22,	367.268,	282.728
23,	367.141,	261.199
24,	367.057,	239.383
25,	366.820,	215.907
26,	366.482,	191.915
27,	366.354,	166.749
28,	366.380,	140.802
29,	366.065,	113.941
30,	366.034,	86.399



$$y = \frac{1}{2}gt^2$$

y

# Kinematik Analiz

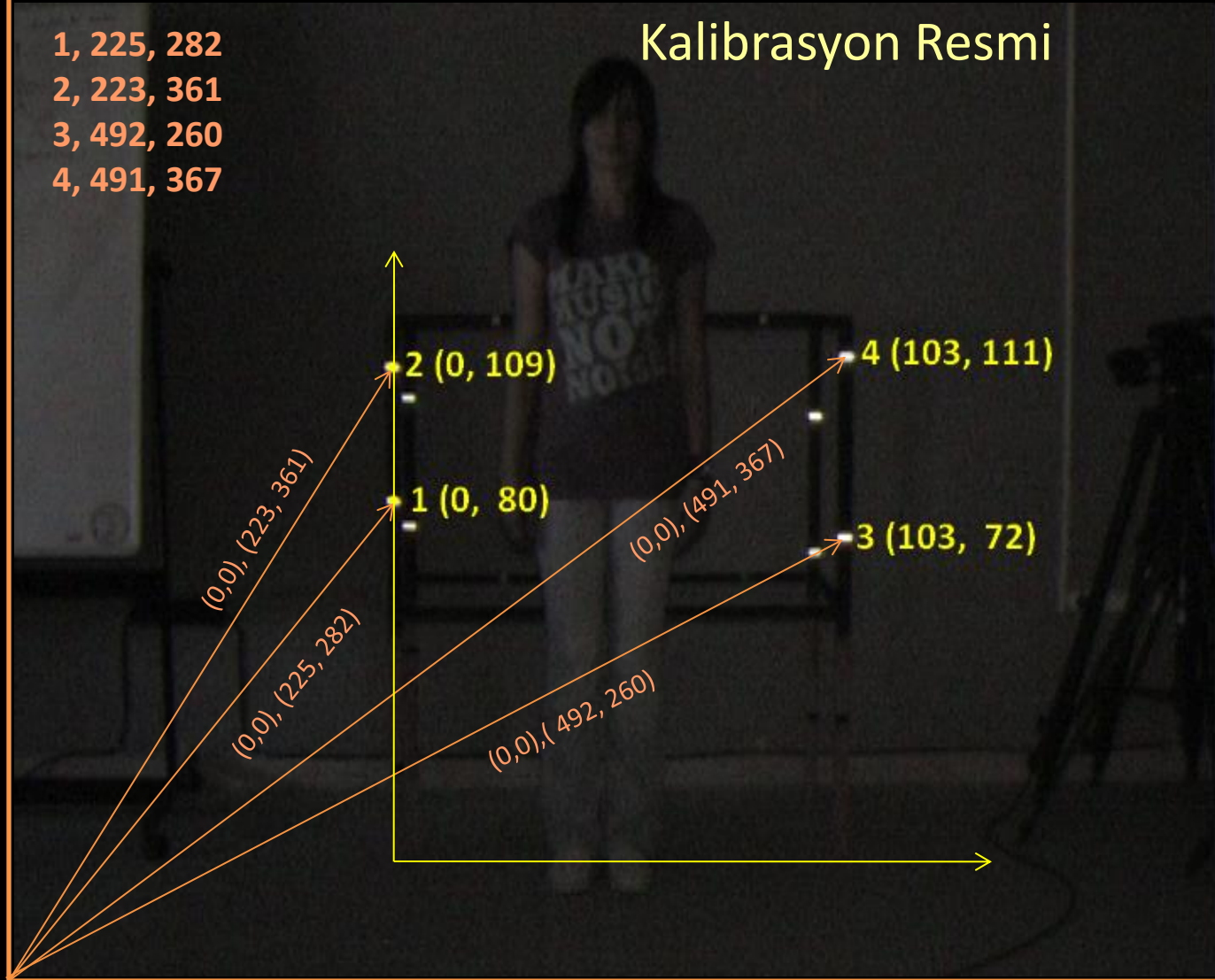
Kalibrasyon Resmi

1, 225, 282

2, 223, 361

3, 492, 260

4, 491, 367



x

# Kinematik Analiz

## İki-Boyutlu Uyum Dönüşümü (2D Conformal Transformation)

***Dört-parametrelili benzerlik dönüşümü*** olarak da bilinir

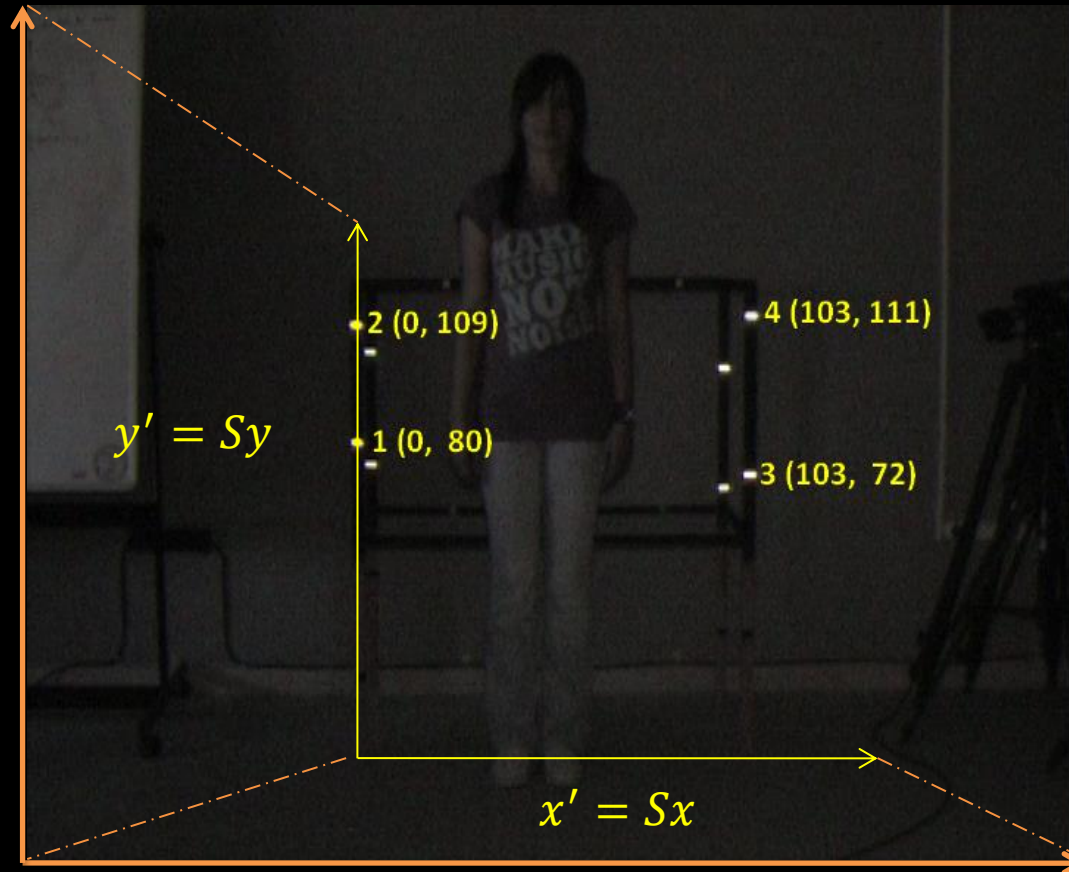
Bu dönüşüm 3 basamaktan oluşmaktadır:

1. ***Ölçekleme*** : iki koordinat sisteminde denk boyut yaratmak için
2. ***Dönme*** : iki sistemin referans eksenlerini paralel yapmak için
3. ***Öteleme*** : iki koordinat sisteminde ortak bir başlangıç noktası yaratmak için

# Kinematik Analiz

## İki-Boyutlu Uyum Dönüşümü

**Ölçekleme** : iki koordinat sisteminde denk boyut yaratmak için





# Kinematik Analiz

## İki-Boyutlu Uyum Dönüşümü

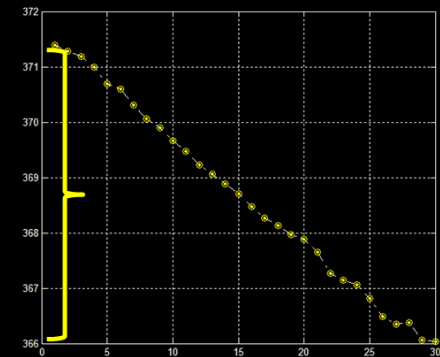
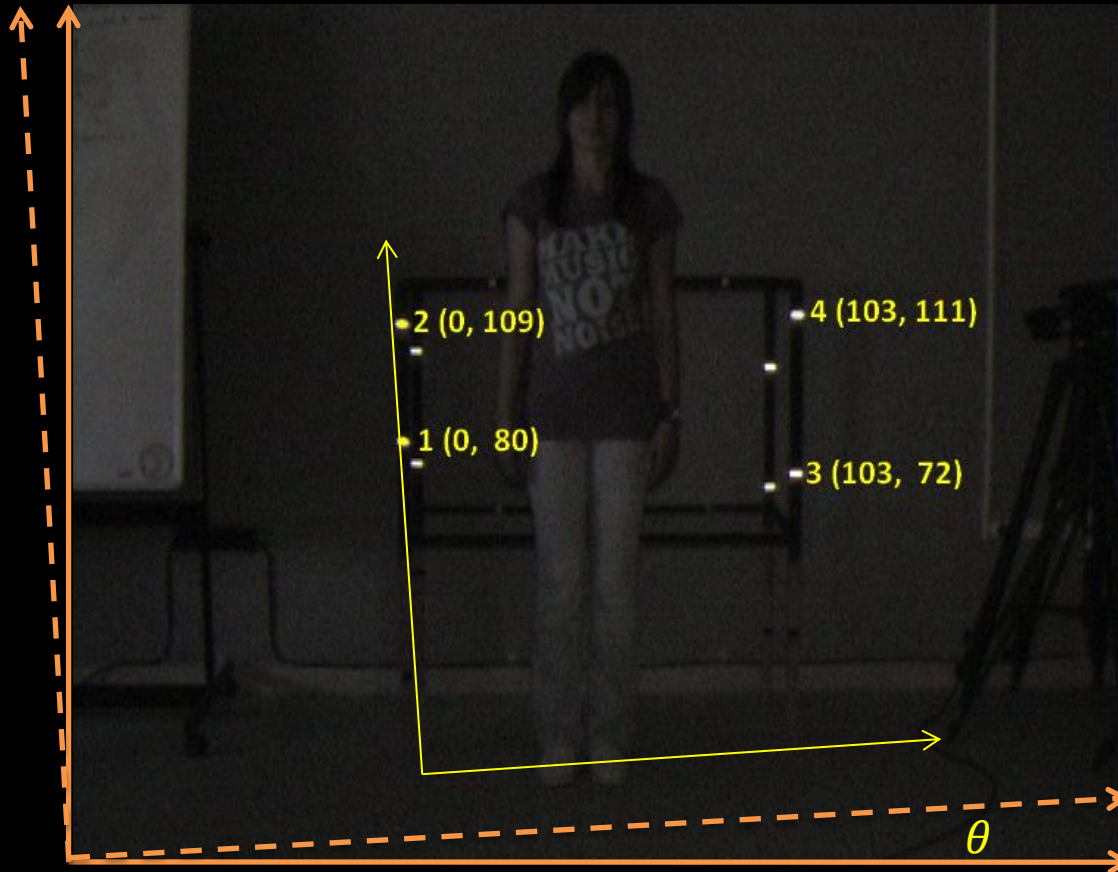
1.Basamak : Ölçekleme.  $(x,y)$  sisteminde tanımlanan çizgi uzunluklarını  $(X,Y)$  sisteminde de aynı uzunluğa getirmek için,  $(x,y)$  koordinatlarını ölçekleme faktörüyle ( $S$ ) çarpmak gerekir.. Burada ölçeklendirilmiş koordinatlar  $x'$  ve  $y'$  dür:

$$\begin{aligned}x' &= Sx \\ y' &= Sy\end{aligned}\quad \text{E.[1]}$$

# Kinematik Analiz

## İki-Boyutlu Uyum Dönüşümü

**Dönme** : iki sistemin referans eksenlerini paralel yapmak için





# Kinematik Analiz

## İki-Boyutlu Uyum Dönüşümü

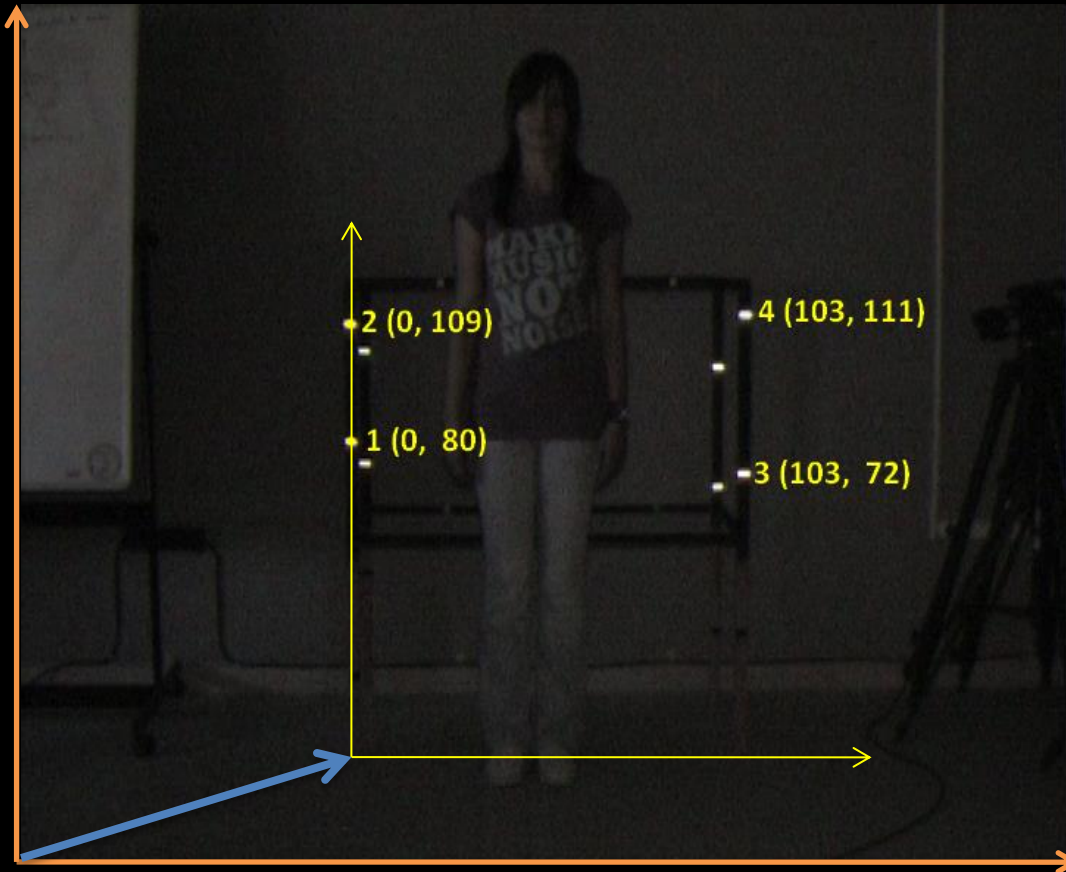
2.Basamak : Dönme. Ölçeklendirilmiş sistemde ( $x'$ ,  $y'$ ) iki sistemin referans eksenlerini paralel yapmak için ölçeklendirilmiş sistemin ( $\theta$ ) açısı kadar döndürülmelidir. Burada döndürülmüş koordinatlar  $X'$  ve  $Y'$  dür:

$$\begin{aligned} X' &= x' \cos(\theta) - y' \sin(\theta) \\ Y' &= x' \sin(\theta) + y' \cos(\theta) \end{aligned} \quad \text{E.[2]}$$

# Kinematik Analiz

## İki-Boyutlu Uyum Dönüşümü

**Öteleme** : iki koordinat sisteminde ortak bir başlangıç noktası yaratmak için



# Kinematik Analiz

## İki-Boyutlu Uyum Dönüşümü

3.Basamak : Öteleme. iki koordinat sisteminde ortak bir başlangıç noktası yaratmak için,  $(X', Y')$  sisteminin başlangıç noktasını  $(X, Y)$  sisteminin başlangıç noktasına ötelenmesi gerekir. Burada ötelenmiş koordinatlar  $X$  ve  $Y$  dir:

$$\begin{aligned} X &= X' + T_X \\ Y &= Y' + T_Y \end{aligned} \quad \text{E.[3]}$$

# Kinematik Analiz

## İki-Boyutlu Uyum Dönüşümü

Eğer [1.],[2.] ve [3.] eşitlikler birleştirildiğinde sistem uyumlu tek bir eşitlik sistemine dönüşür. Bu da  $(x, y)$  koordinatlarını doğrudan  $(X, Y)$  sistemine dönüştürebilir.

$$\begin{aligned} X &= (S \cos \Theta)x - (S \sin \Theta)y + T_x \\ Y &= (S \sin \Theta)x + (S \cos \Theta)y + T_y \end{aligned} \quad \text{E.[4]}$$



# Kinematik Analiz

## İki-Boyutlu Uyum Dönüşümü

Eşitlik 4. de  $S.\cos(\theta) = a$ ,  $S.\sin(\theta) = b$ ,  $T_x = c$  ve  $T_y = d$  diyerek eşitliği tekrar yazarsak

$$ax - by + c = X$$

$$ay + bx + d = Y$$

E.[5]

# Kinematik Analiz

## İki-Boyutlu Uyum Dönüşümü

Eşitlik [5] 4 bilinmeyenli (a,b,c ve d) 2-Boyutlu uyum dönüşümü göstermektedir. Buradaki bilinmeyenler dönüşüm parametreleri olan  $S$ ,  $\theta$ ,  $T_x$  ve  $T_y$  de içermektedir. Her bir kalibrasyon noktası için 2 eşitlik yazıldığından sistemin tek çözümü için sadece 2 nokta yeterli olacaktır. İki kalibrasyon noktasından fazla olan artık sistemlerde (redundant system) ise En-Küçük Kareler metodu kullanılarak çözüm bulunur.

Örneğin : 3 kalibrasyon noktası için 6 ayrı eşitlik yazılabilir.

$$ax_a - by_a + c = X_A$$

$$ay_a + bx_a + d = Y_A$$

$$ax_b - by_b + c = X_B$$

$$ay_b + bx_b + d = Y_B$$

$$ax_c - by_c + c = X_C$$

$$ay_c + bx_c + d = Y_C$$

E.[6]

# Kinematik Analiz

## İki-Boyutlu Uyum Dönüşümü

Eşitlik [6] matris formunda yazıldığında;

$$A = \begin{bmatrix} x_a & -y_a & 1 & 0 \\ y_a & x_a & 0 & 1 \\ x_b & -y_b & 1 & 0 \\ y_b & x_b & 0 & 1 \\ x_c & -y_c & 1 & 0 \\ y_c & x_c & 0 & 1 \end{bmatrix} \quad X = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \quad L = \begin{bmatrix} X_A \\ Y_A \\ X_B \\ Y_B \\ X_C \\ Y_C \end{bmatrix}$$

$$\Theta = \tan^{-1} \left( \frac{b}{a} \right)$$

$$S = \frac{a}{\cos(\Theta)}$$

Öteleme

$$T_x = c$$

$$T_y = d$$





# Kinematik Analiz

## İki-Boyutlu Uyum Dönüşümü

En-Küçük Kareler çözüm yolu;

$$Sx = I_{calib}$$

$$S^T Sx = S^T I_{calib}$$

$$x = (S^T S)^{-1} S^T I_{calib}$$

# Kinematik Analiz

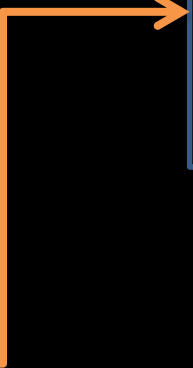
## Nasıl Hesaplayacağız?



```
% main program
clc, clear
% Calibration frame values
S = [ 0 80;      % 1.
      0 109;     % 2.
      103 72;    % 3.
      103 111]; % 4.

load calib_im.txt;
I = calib_im;
```

Calib\_im.txt



225	282
223	361
492	260
491	367

```
x = calculate_conformal(I, S, 1);
```

 **function**

```
teta = atand(x(2)/x(1));
scale = x(1)/cosd(teta);
Tx = x(3);
Ty = x(4);
```



```
function P = calculate_conformal(I, S, method)
%'      Conformal Map Transformation      ';
%'      Serdar Arıtan 2009 BAG            ';
%'      -----                        ';
%' Image Coordinates of Calibration Points : I';
%' Space Coordinates of Calibration Points : S';
[rS, cS] = size(S);
[rI, cI] = size(I);
% check the matrix size
if cS ~= cI || rS ~= rI
    error('matrix dimension');
end
% Coefficients Matrix
A = [I(:,1) -I(:,2) ones(rI,1) zeros(rI,1);
     I(:,2)  I(:,1) zeros(rI,1) ones(rI,1)];

% There are two type of solutions for over determined systems of eq.
% 1. Least Square Method [ \ ]
% 2. Psedou Inverse [ pinv() ]
if method == 1
    P = A\S(:);           %S = [S(:)]; % x(1), ... ,x(n) , y(1), ... , y(n)
elseif method == 2
    P = pinv(A)*S(:);
else
    disp('missing method!!');
end
```



```
load ball_drop.txt;  
H = calculate_reconformal(x, ball_drop);
```

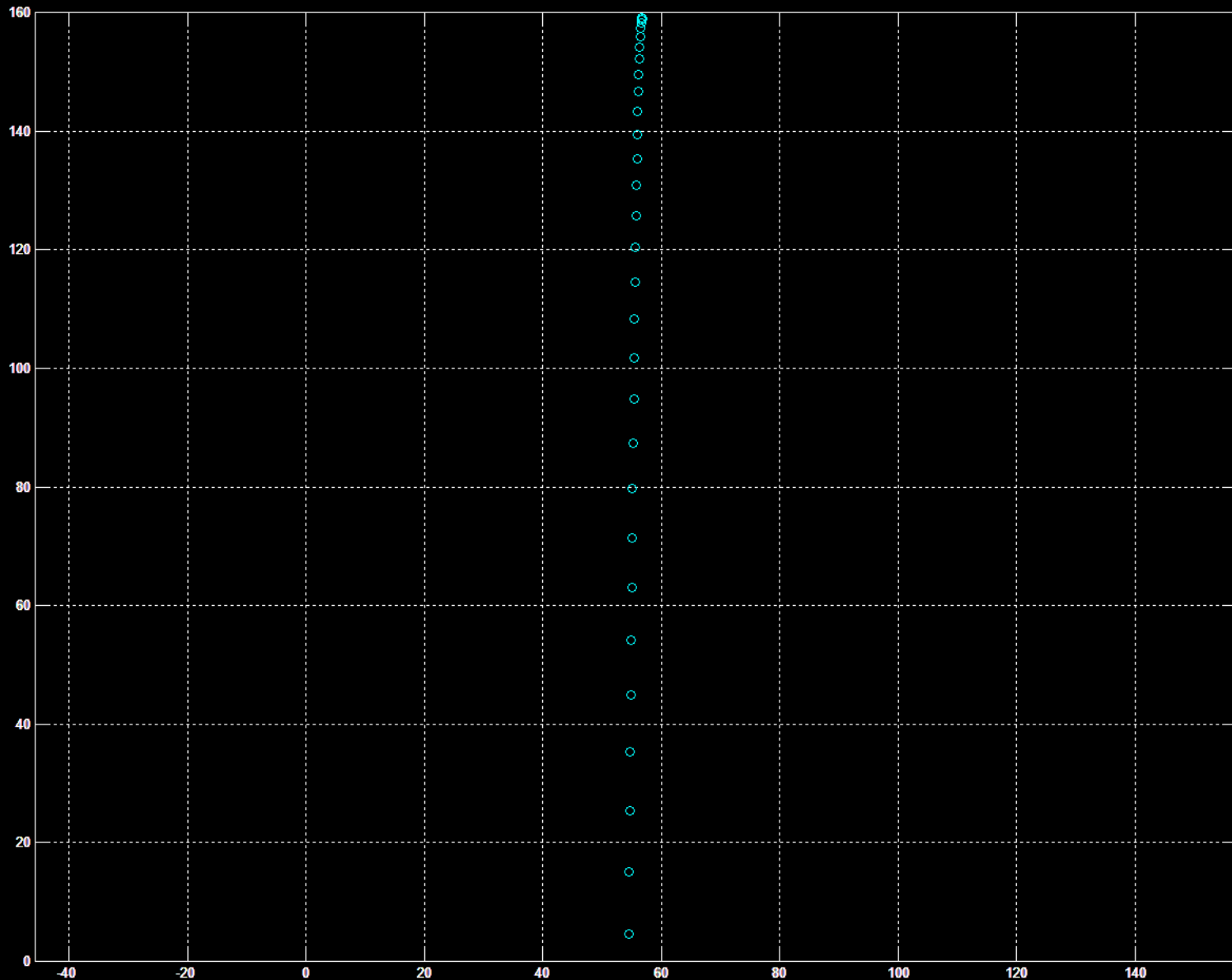
```
function H = calculate_reconformal(P, I)  
% '      Conformal Map Transformation      '  
% '      Serdar Arıtan 2009 BAG              '  
% ' ----- '  
% ' Image Coordinates of Data Points : I';  
% ' Conformal Coefficients             : P';  
  
[rI, cI] = size(I);  
  
% Coefficients Matrix  
A = [I(:,1) -I(:,2) ones(rI,1) zeros(rI,1);  
      I(:,2)  I(:,1) zeros(rI,1) ones(rI,1)];  
  
H = A*P;  
H = [H(1:length(H)/2) H((length(H)/2)+1:end)];
```

Ball\_drop.txt

```
371.391 489.972  
371.274 490.200  
371.184 489.558  
      :      :  
366.380 140.802  
366.065 113.941  
366.034  86.399
```



```
figure(1), plot(H(:,1),H(:,2), 'ro');axis([0 100 0 160]);axis equal;grid on;
```



# Kinematik Analiz

Sonlu Farklar Analizi : Merkezden Fark metodu

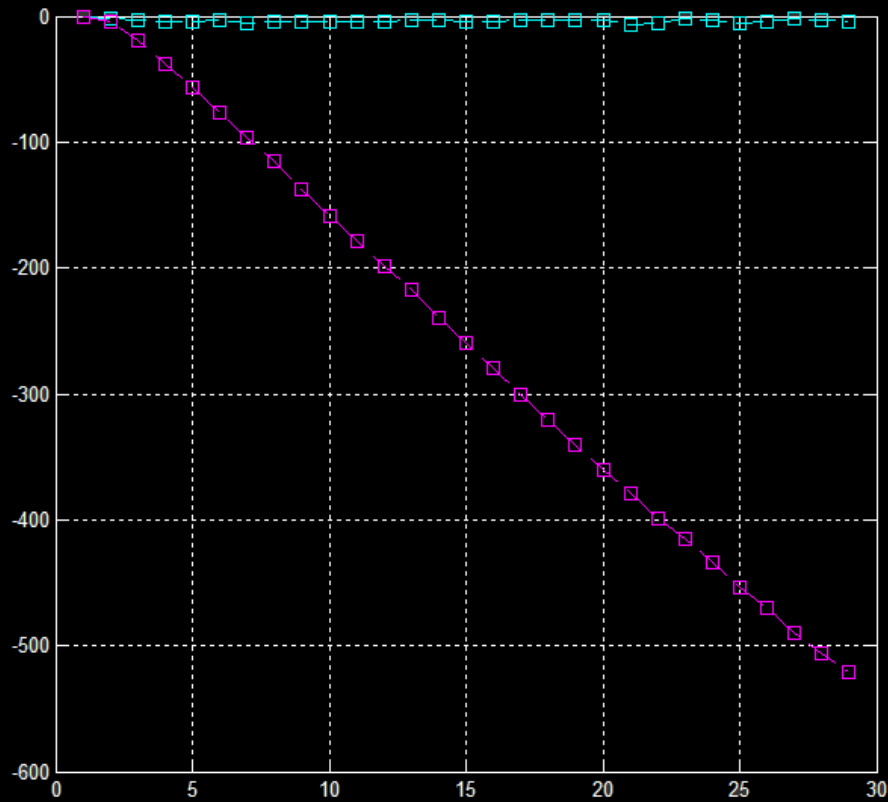
$$v_i = \frac{s_{i+1} - s_{i-1}}{2\Delta t}$$

$$a_i = \frac{v_{i+1} - v_{i-1}}{2\Delta t} = \frac{s_{i+2} - 2s_i + s_{i-2}}{4(\Delta t)^2}$$

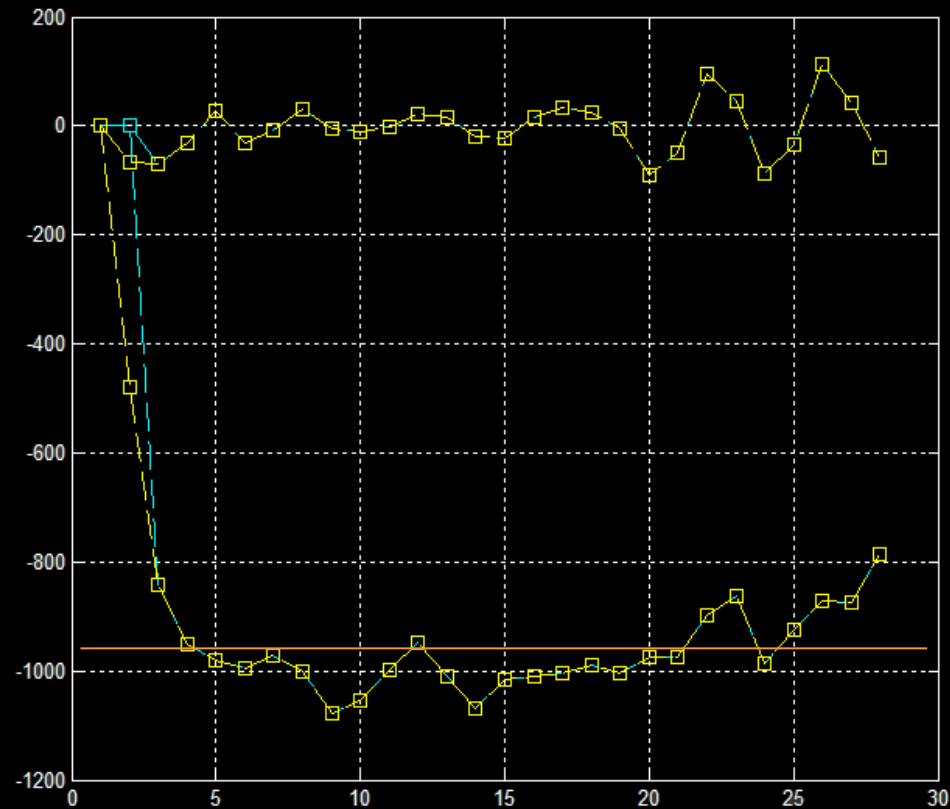
$$a_i = \frac{s_{i+1} - 2s_i + s_{i-1}}{\Delta t^2}$$

# Kinematik Analiz

Hız

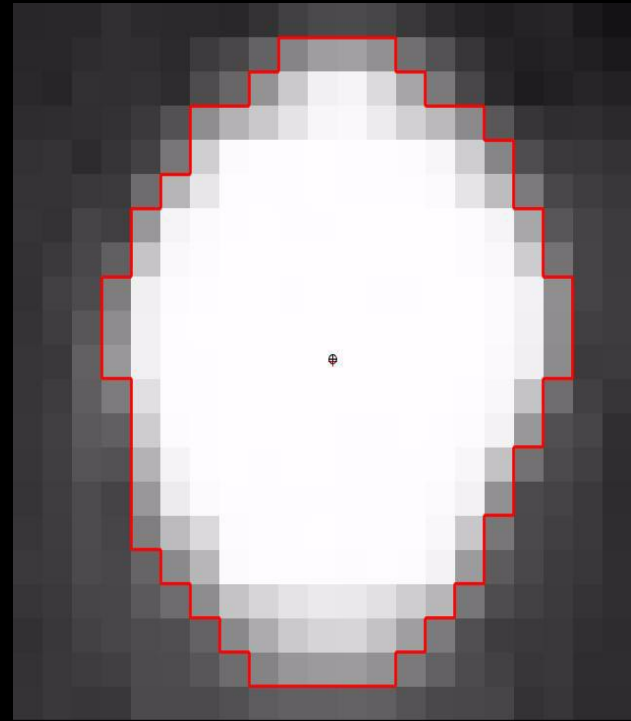
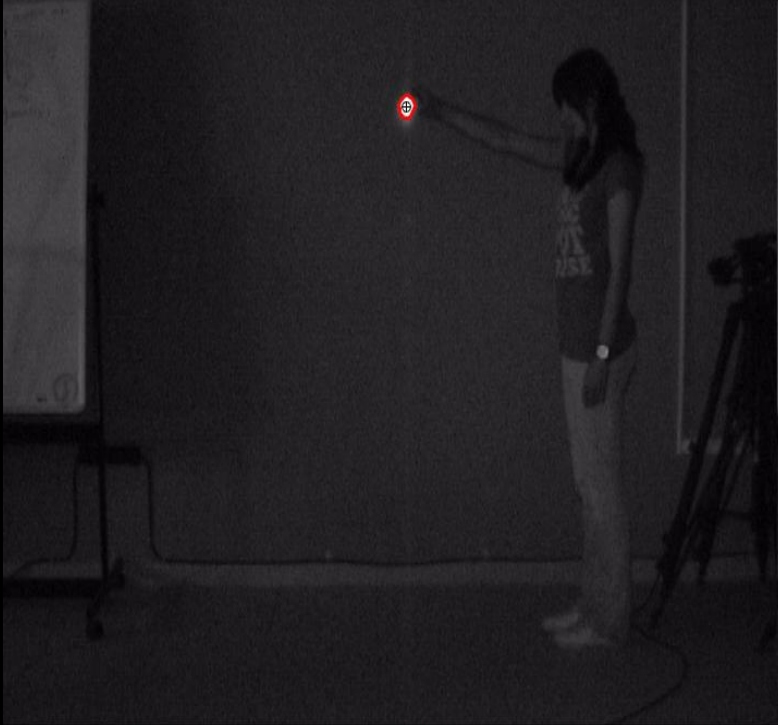


İvme



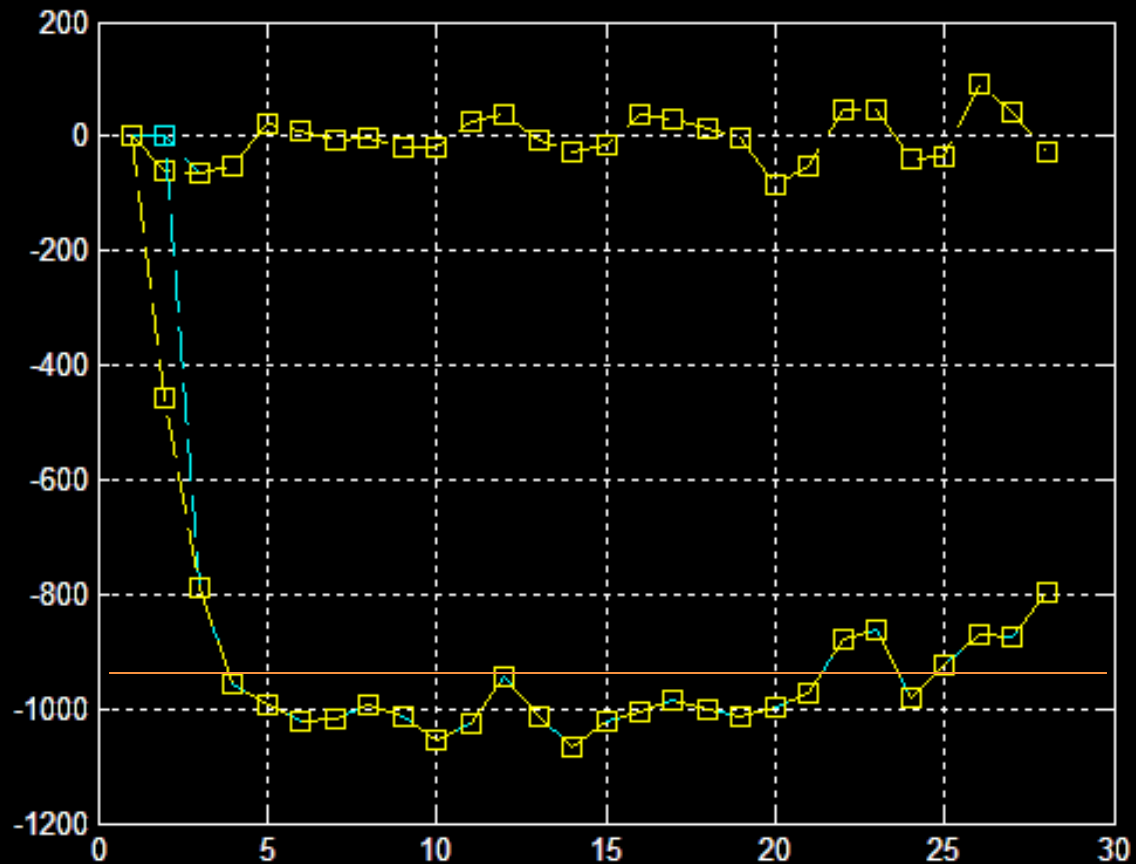


# Kinematik Analiz



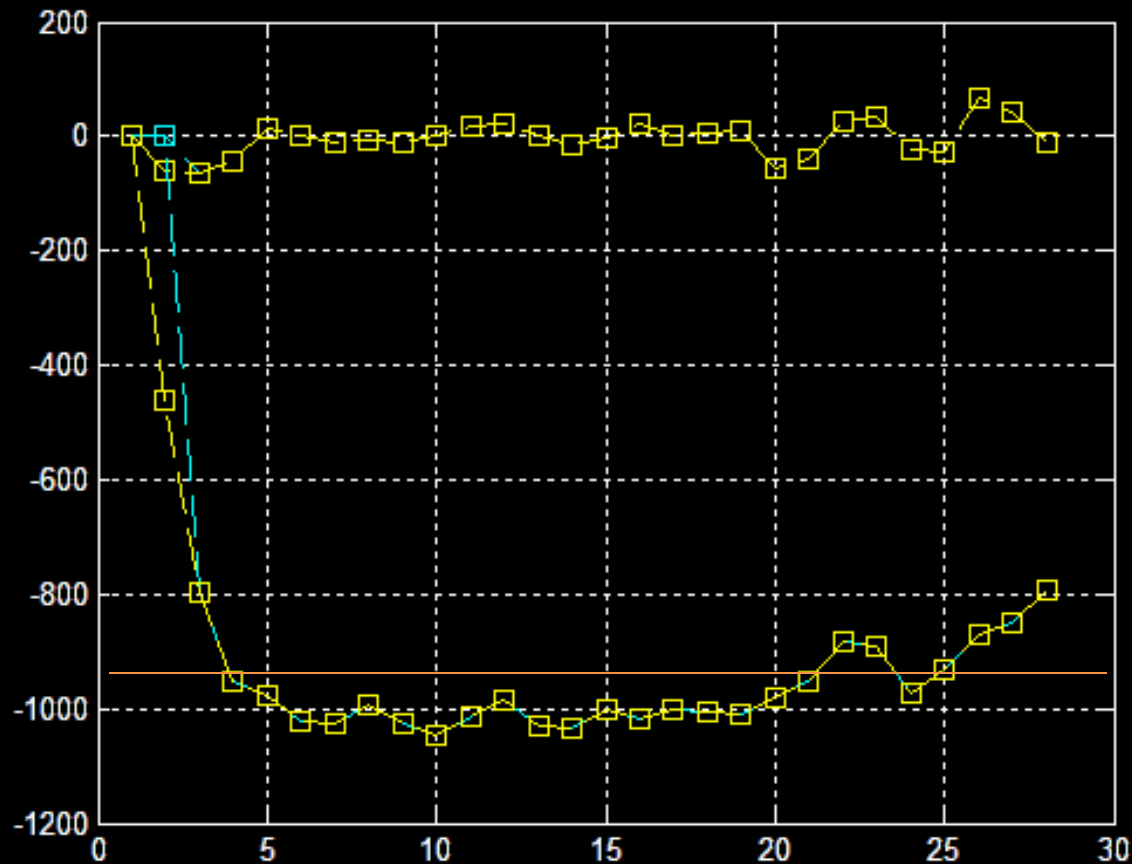
# Kinematik Analiz

## Kütle Merkezi



# Kinematik Analiz

## Ağırlıklı Kütle Merkezi





## Yansıtıcı işaret yakalama

Top Bırak görüntülerindeki topun merkezini dersde anlatılan iki yöntemi kullanarak hesaplayınız Görüntüyü kalibre ederek hız ve ivmeyi hesaplayınız.

Teslim Tarihi : 13 Kasım 2019 Çarşamba  
Saat 10:00



## Öğrendiğimiz MATLAB fonksiyonları:

`imtool`

`warning`

`return`

`regionprops(...'WeightedCentroid'...)`

`fopen('filename.xxx','permission')`

`fprintf(fid,'format',variables)`

`fclose(fid)`