



BCO 607 Hareket Analizi Sistemleri

OpenCV ile Görüntü İşleme 4



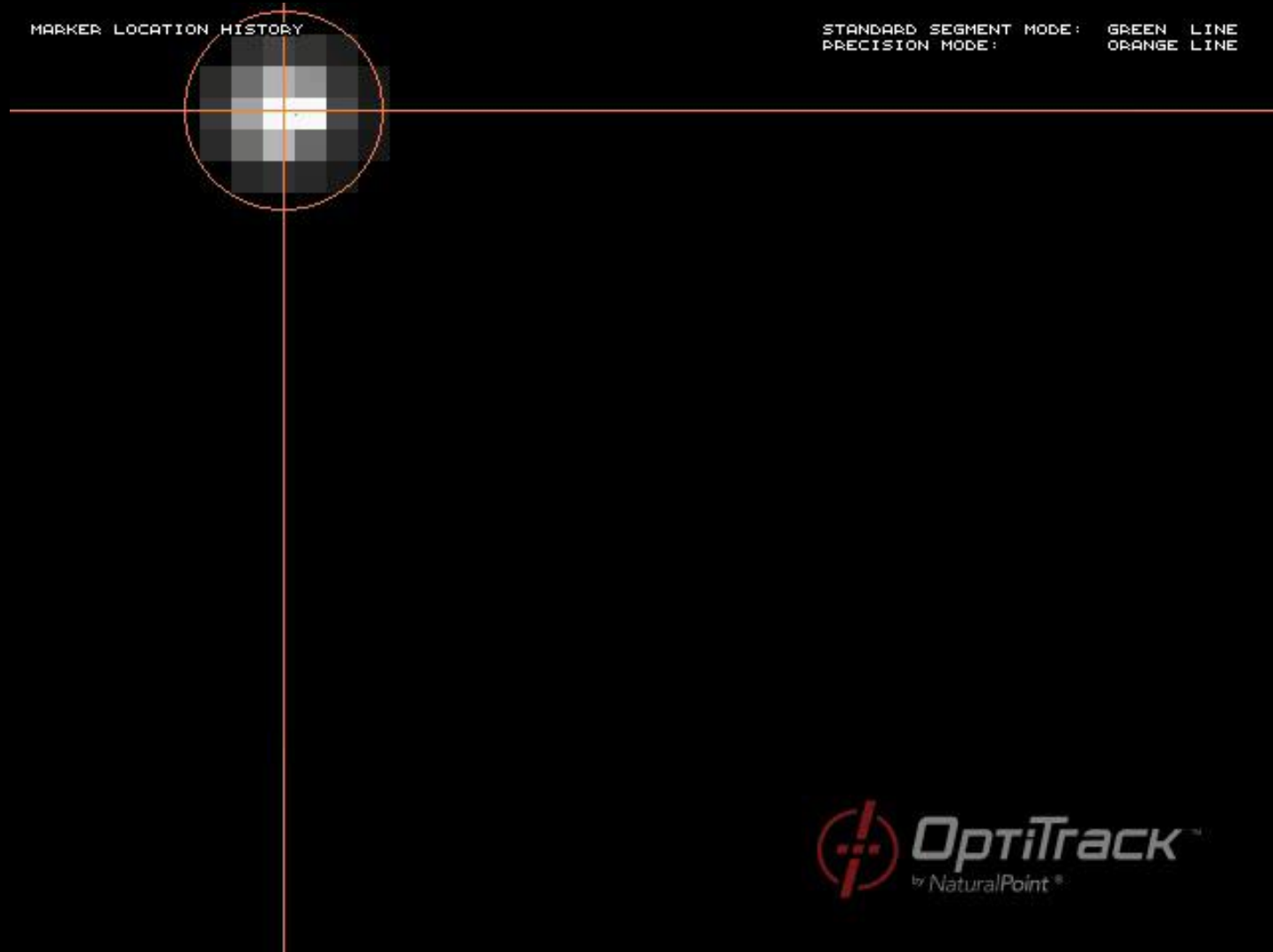
SERDAR ARITAN

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

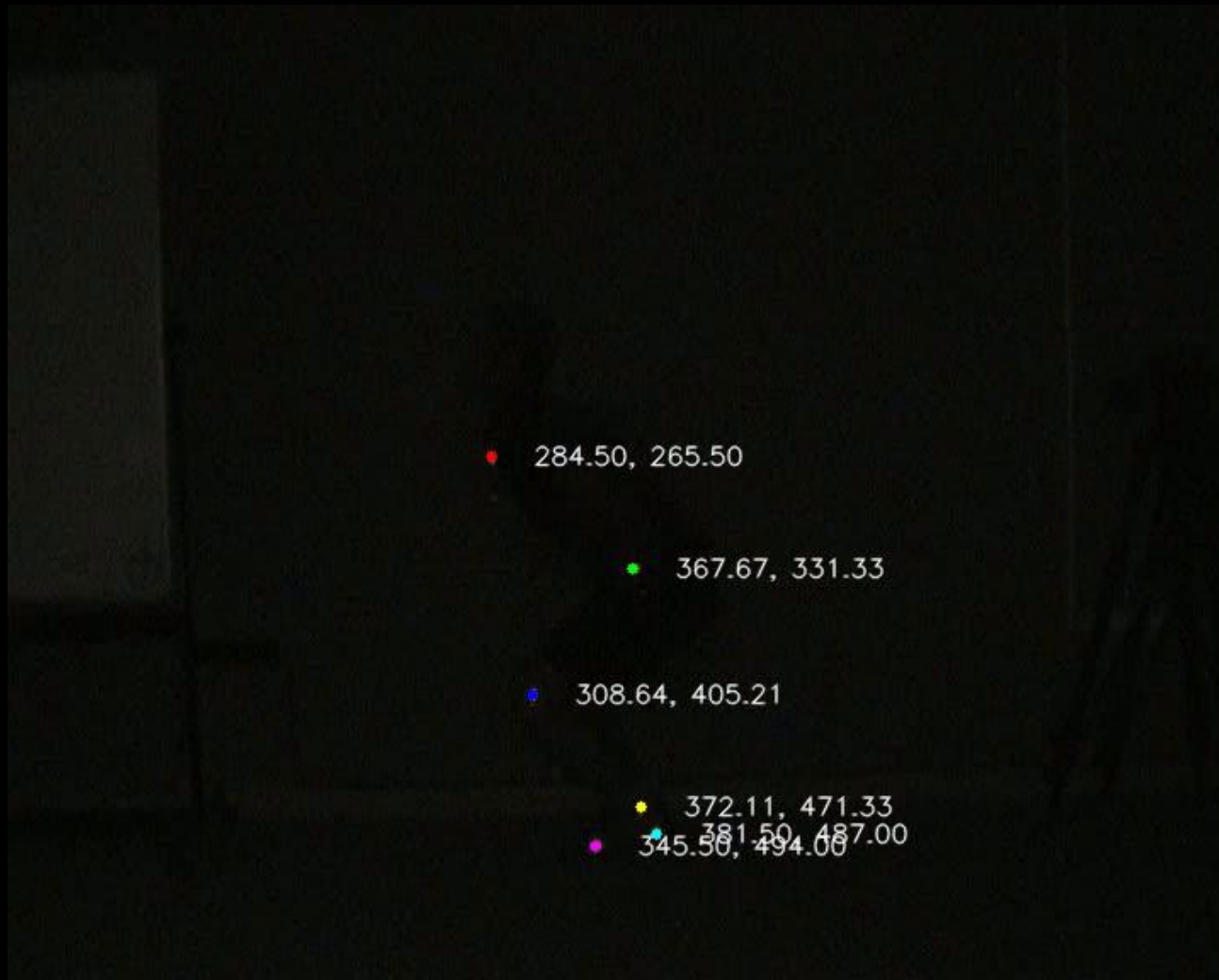


Hassas Konum Bulma





Hassas Konum Bulma



Yansıtıcı işaret yakalama

```
# Read the image
BGR = cv.imread('jump_106.jpg')
# cv.imshow('Original image',BGR)
# cv.waitKey(0)
# Convert to the gray
imgray = cv.cvtColor(BGR, cv.COLOR_BGR2GRAY)
# cv.imshow('Gray image', imgray)
# cv.waitKey(0)
# median filter salt&pepper
filtered = cv.medianBlur(imgray, 3)
# cv.imshow('Filtered', filtered)
# cv.waitKey(0)
# Convert to the Black & White binary image
_,bw = cv.threshold(filtered, 90, 255, cv.THRESH_BINARY)
# cv.imshow('BW', bw)
# cv.waitKey(0)
```



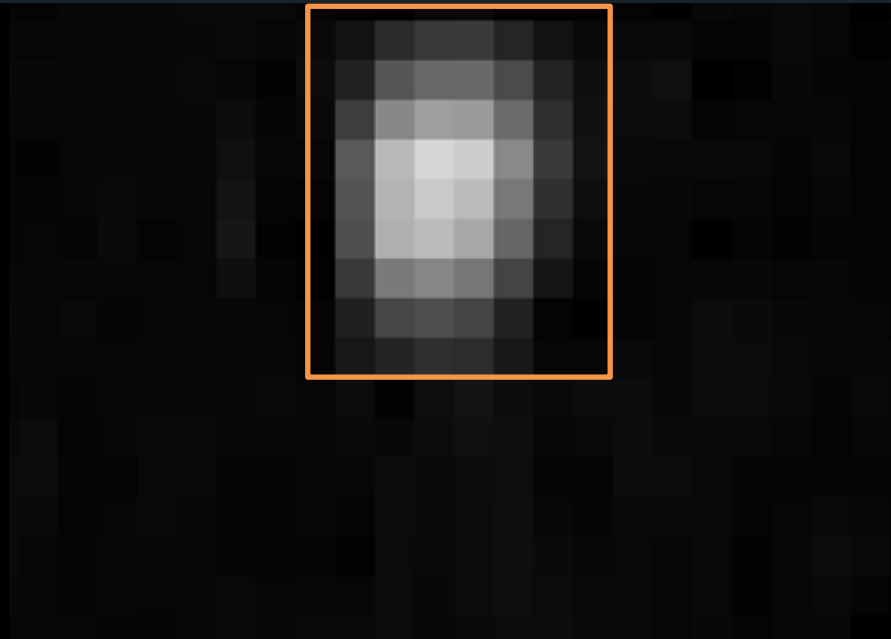
Yansıtıcı işaret yakalama



Yansıtıcı işaret yakalama

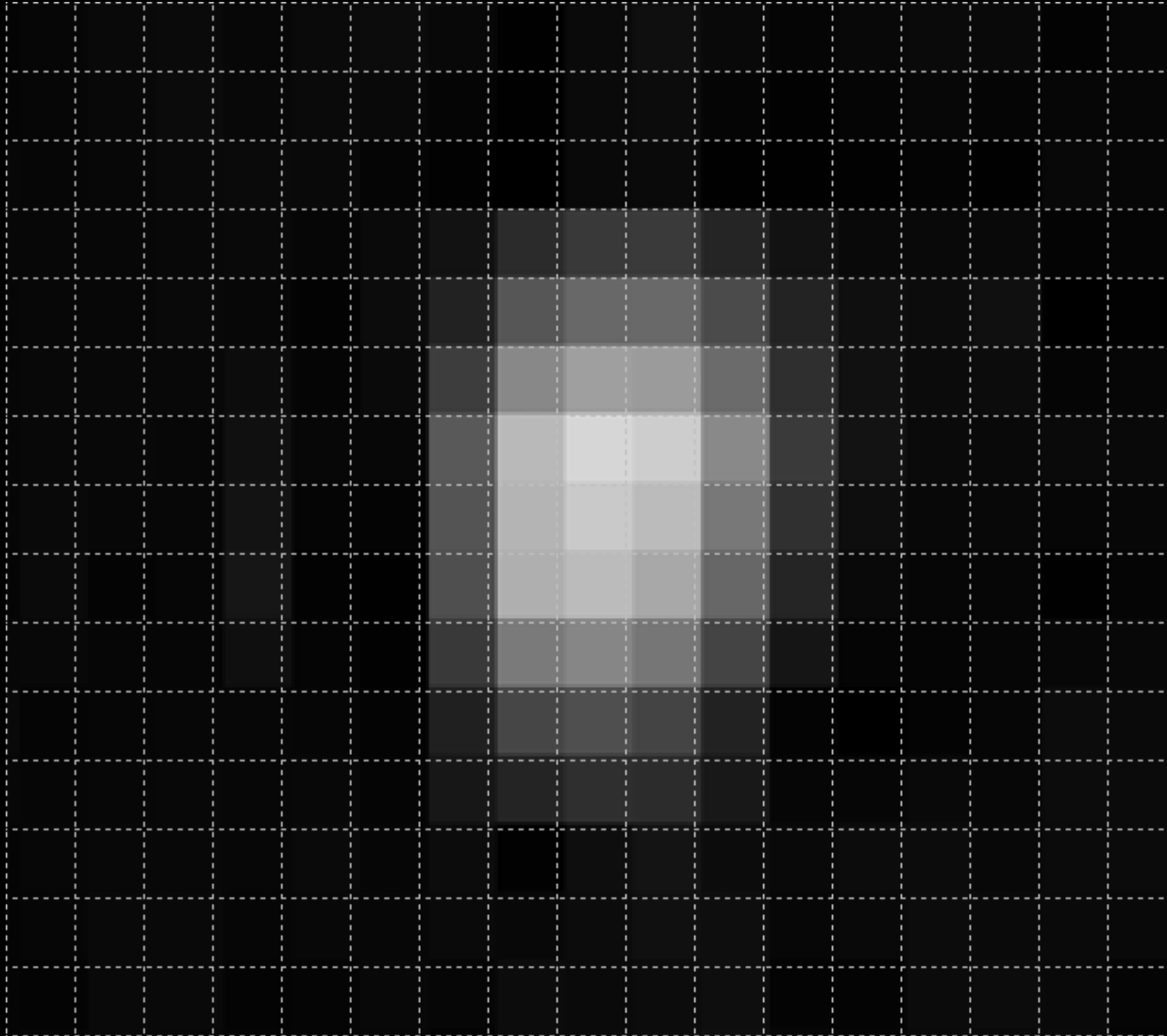
Noktanın merkezi neresi ?

Name ▲	Type	Size	Value
BGR	Array of uint8	(576, 720, 3)	<pre>[[[6 8 8] [4 6 6] [0 0 0 ... 0 0 0] [0 0 0 ... 0 0 0]]]</pre>
bw	Array of uint8	(576, 720)	<pre>[[0 0 0 ... 0 0 0] [0 0 0 ... 0 0 0]]]</pre>
filtered	Array of uint8	(576, 720)	<pre>[[8 6 6 ... 8 8 5] [8 6 6 ... 7 6 5]]]</pre>
imgray	Array of uint8	(576, 720)	<pre>[[8 6 6 ... 8 13 5] [11 8 6 ... 7 8 5]]]</pre>





Yansıtıcı işaret yakalama





Yansıtıcı işaret yakalama

Name ▲	Type	Size	Value
BGR	Array of uint8	(576, 720, 3)	[[[6 8 8] [4 6 6] [0 0 0] ... 0 0 0] [0 0 0] ... 0 0 0]
bw	Array of uint8	(576, 720)	[[8 6 6 ... 8 8 5] [8 6 6 ... 7 6 5] [8 6 6 ... 8 13 5] [11 8 6 ... 7 8 5]
filtered	Array of uint8	(576, 720)	
imgray	Array of uint8	(576, 720)	

imgray - NumPy object array

	349	350	351	352	353	354	355	356	
282	10	12	8	1	10	15	9	5	
283	10	9	5	1	10	13	5	2	
284	9	5	2	0	9	10	1	1	
285	6	10	18	43	57	58	38	18	
286	3	13	34	86	104	105	75	35	
287	5	10	61	136	159	155	107	47	
288	7	7	89	185	214	205	138	58	
289	5	4	84	180	201	187	120	48	
290	3	1	79	175	188	168	103	38	
291	5	2	57	122	134	118	68	21	
292	6	4	32	70	79	68	34	4	
293	8	5	23	36	47	44	24	6	
294	10	6	13	2	14	20	13	9	
295	8	7	10	8	12	16	14	7	

Format Resize ☒ Background color Save and Close Close



Yansıtıcı isaret vakalama

	349	350	351	352	353	354	355	356	
282	10	12	8	1	10	15	9	5	
283	10	9	5	1	10	13	5	2	
284	9	5	2	0	9	10	1	1	
285	6	10	18	43	57	58	38	18	
286	3	13	34	86	104	105	75	35	
287	5	10	61	136	159	155	107	47	
288	7	7	89	185	214	205	138	58	
289	5	4	84	180	201	187	120	48	
290	3	1	79	175	188	168	103	38	
291	5	2	57	122	134	118	68	21	
292	6	4	32	70	79	68	34	4	
293	8	5	23	36	47	44	24	6	
294	10	6	13	2	14	20	13	9	
295	8	7	10	8	12	16	14	7	

Format Resize ☒ Background color

Save and Close

Close

283	0	0	0	0	0	0	0	0	
284	0	0	0	0	0	0	0	0	
285	0	0	0	0	0	0	0	0	
286	0	0	0	0	255	255	0	0	
287	0	0	0	255	255	255	255	0	
288	0	0	0	255	255	255	255	0	
289	0	0	0	255	255	255	255	0	
290	0	0	0	255	255	255	255	0	
291	0	0	0	0	255	255	0	0	
292	0	0	0	0	0	0	0	0	
293	0	0	0	0	0	0	0	0	
294	0	0	0	0	0	0	0	0	
295	0	0	0	0	0	0	0	0	

Yansıtıcı işaret yakalama

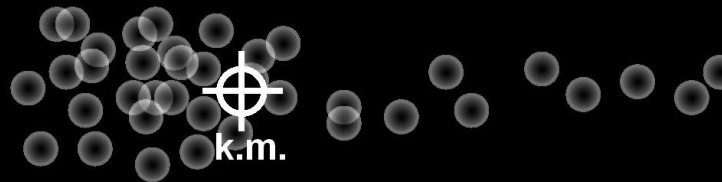
kütle merkezi hesaplama

Kütle Merkezi : Bir cismin toplam kütlesinin ortalama konumu



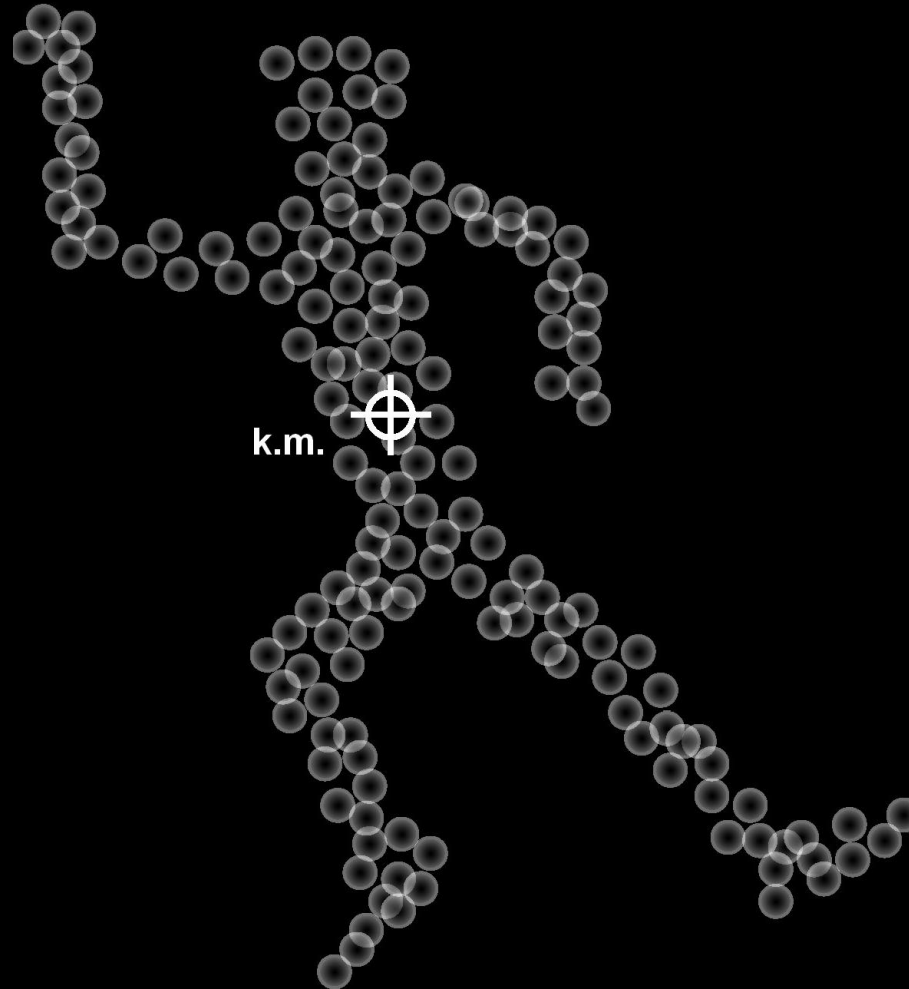
Yansıtıcı işaret yakalama

kütle merkezi hesaplama



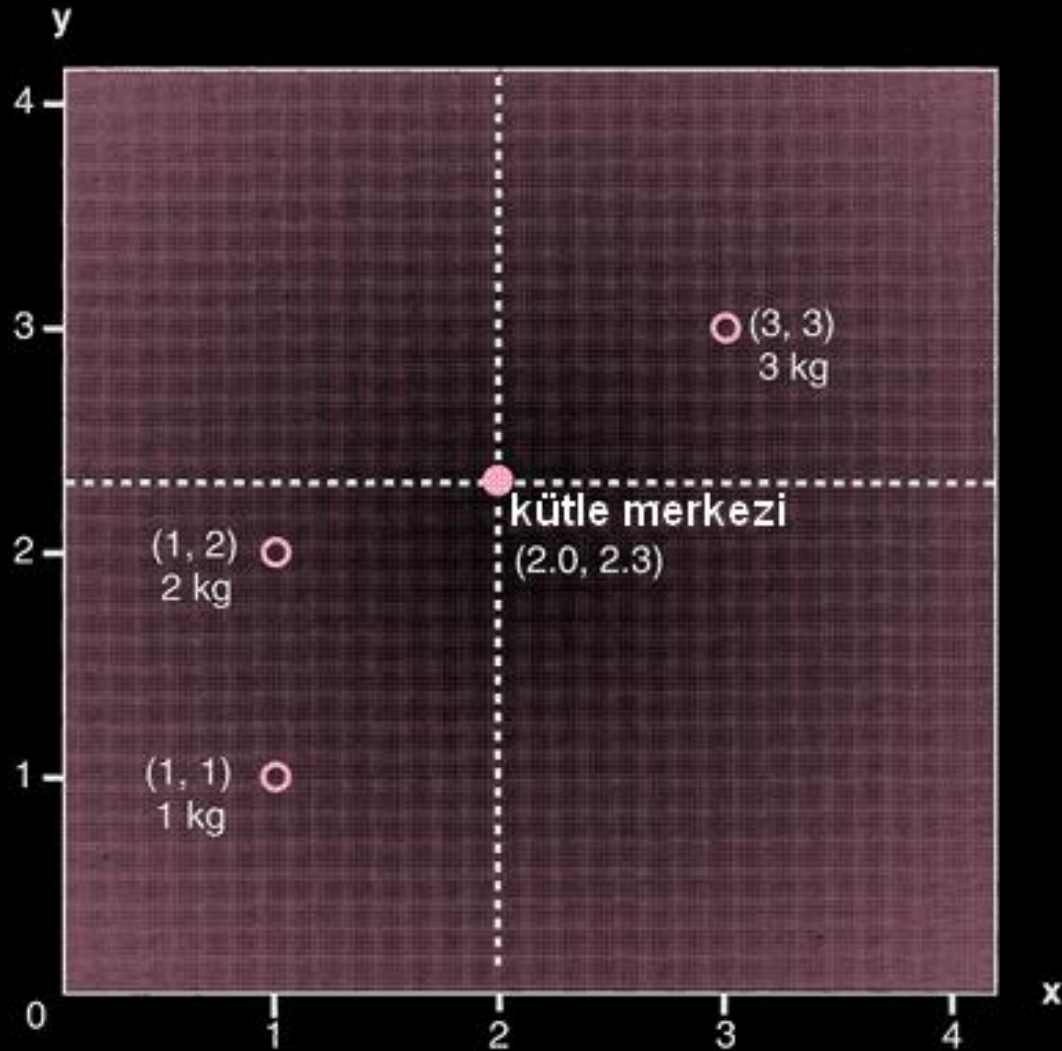
Yansıtıcı işaret yakalama

kütle merkezi hesaplama



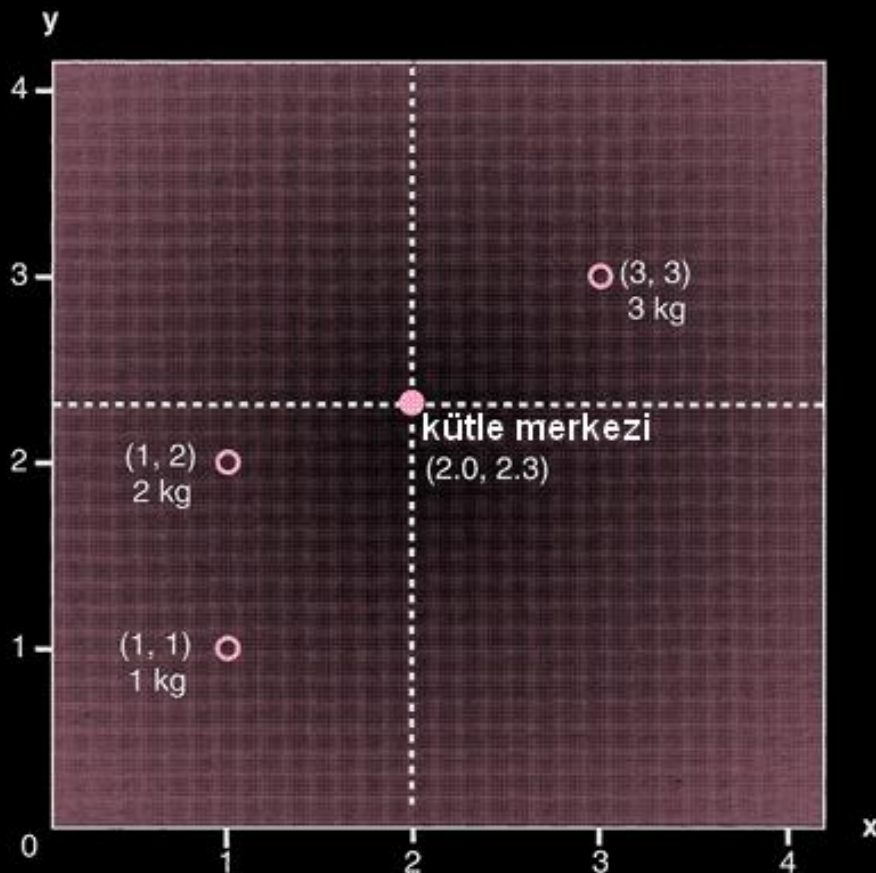
Yansıtıcı işaret yakalama

kütle merkezi hesaplama



Yansıtıcı işaret yakalama

kütle merkezi hesaplama



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

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

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

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

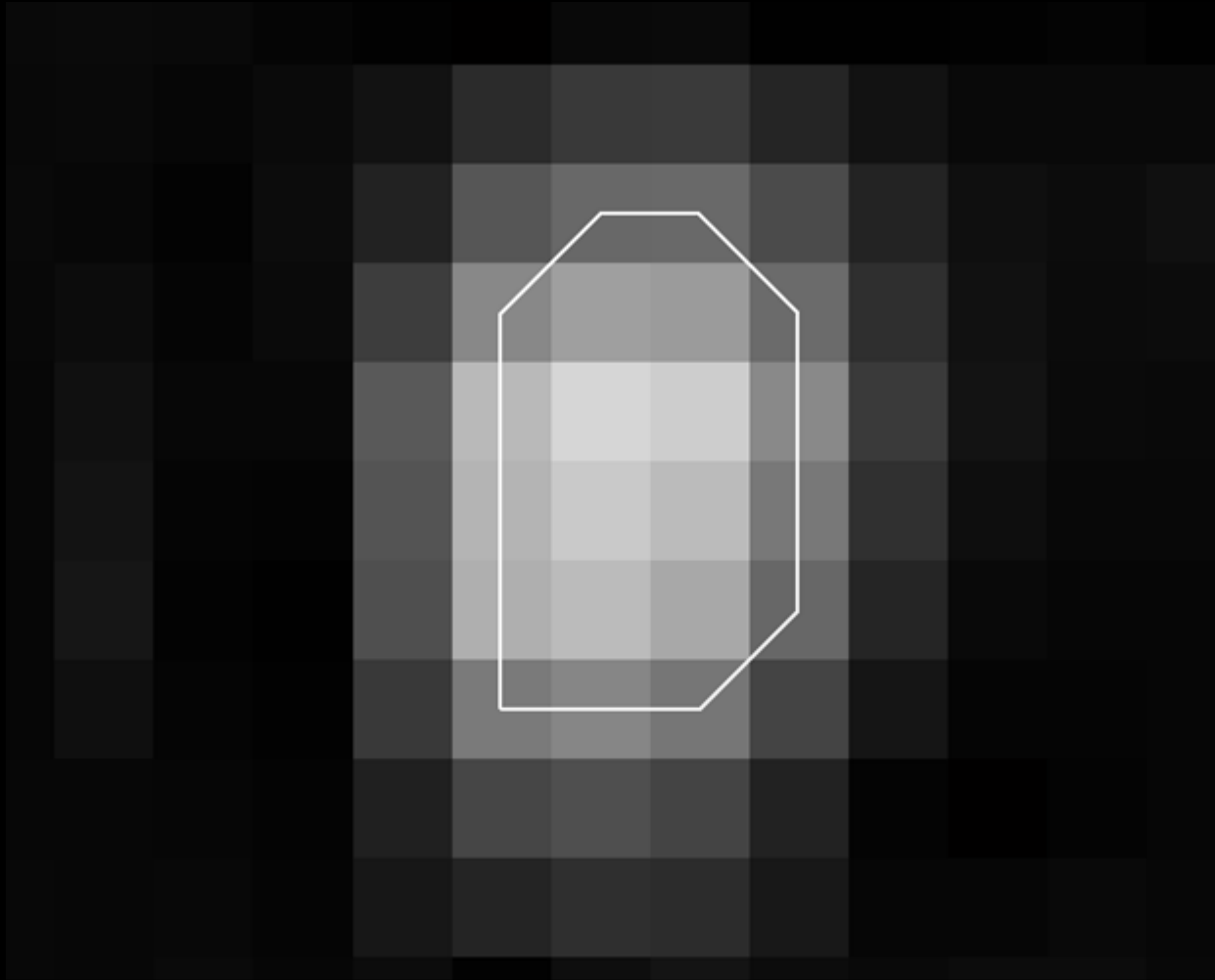
$$m_1y_1+m_2y_2+m_3y_3= My_{km}$$

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

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

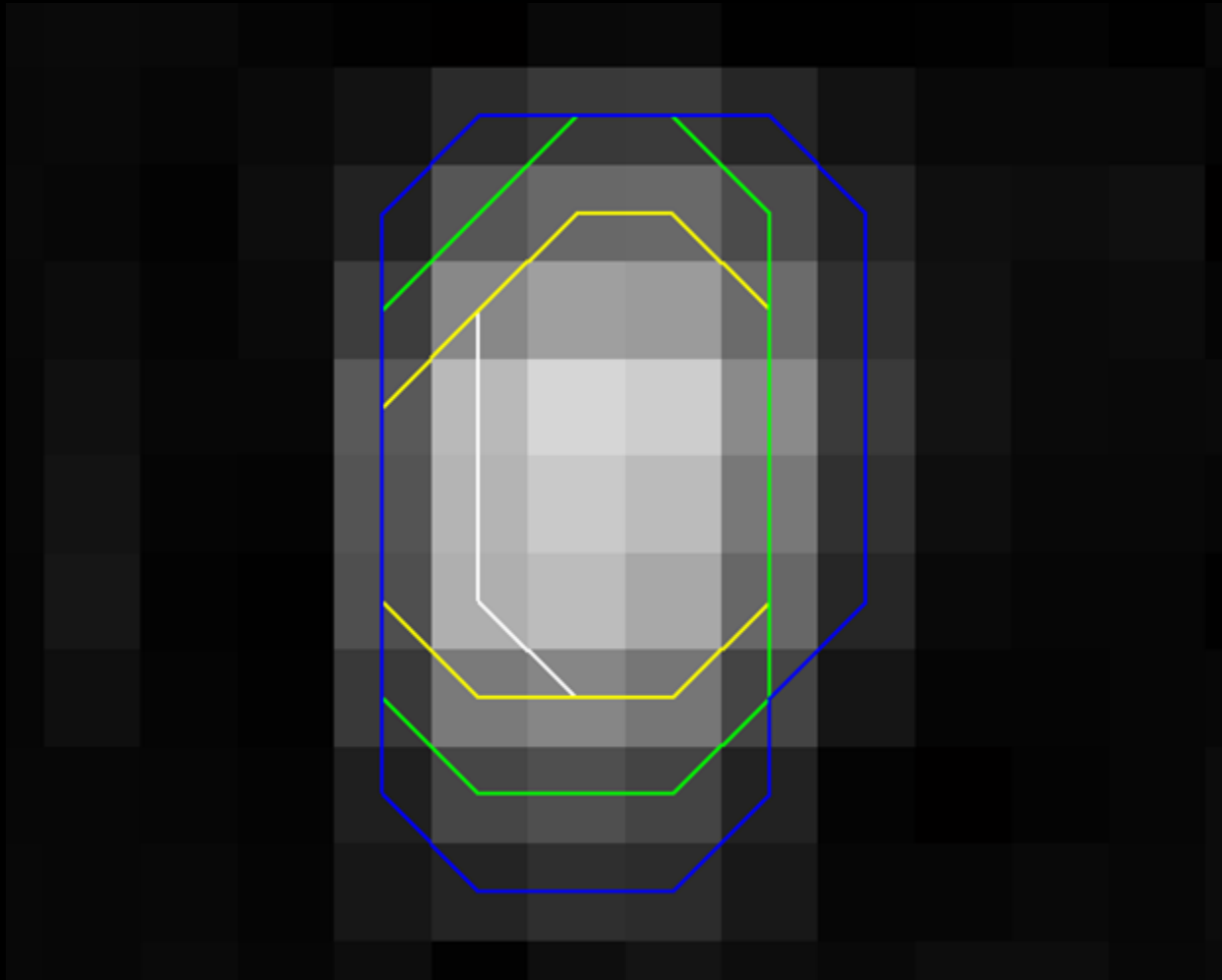


Yansıtıcı işaret yakalama

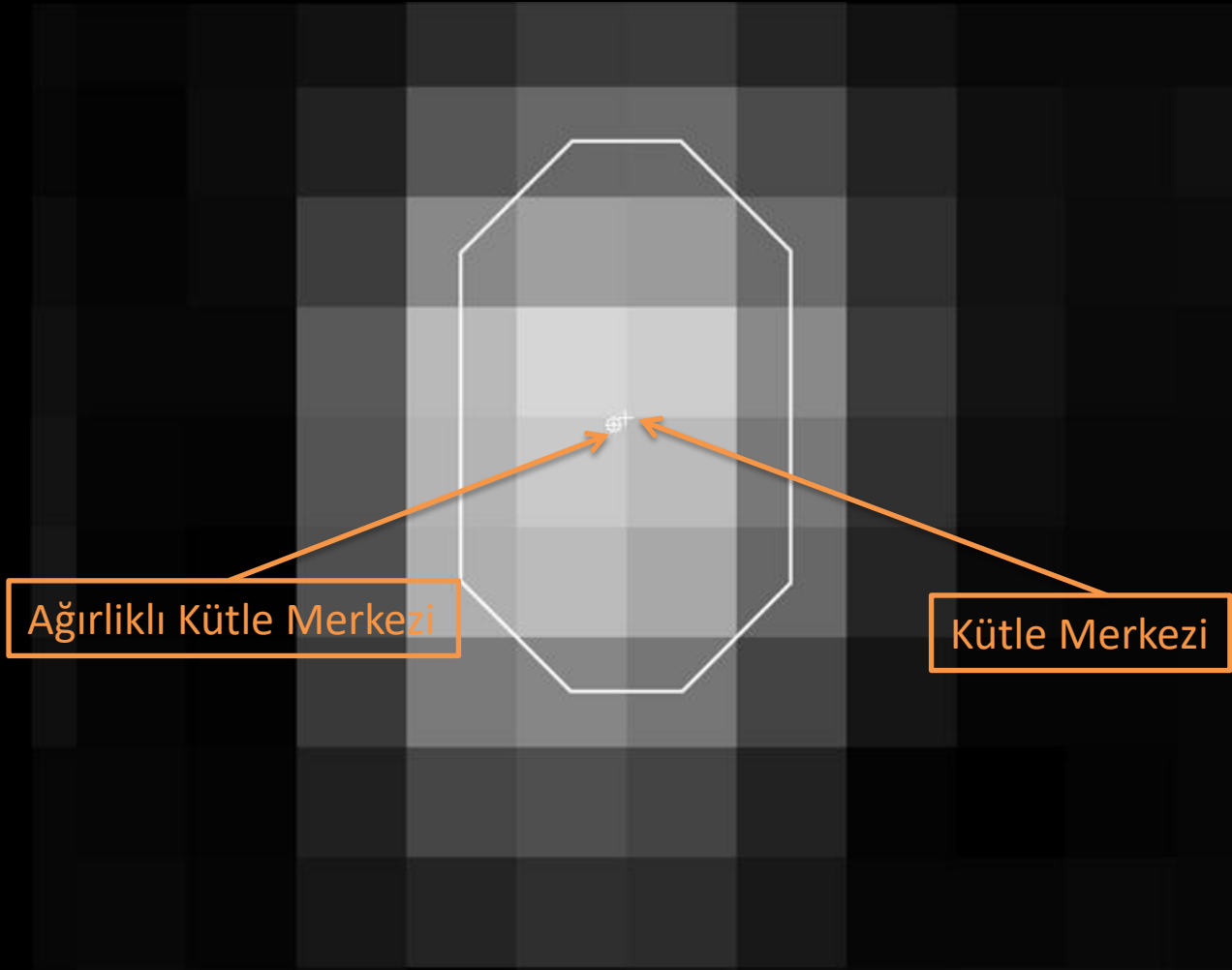




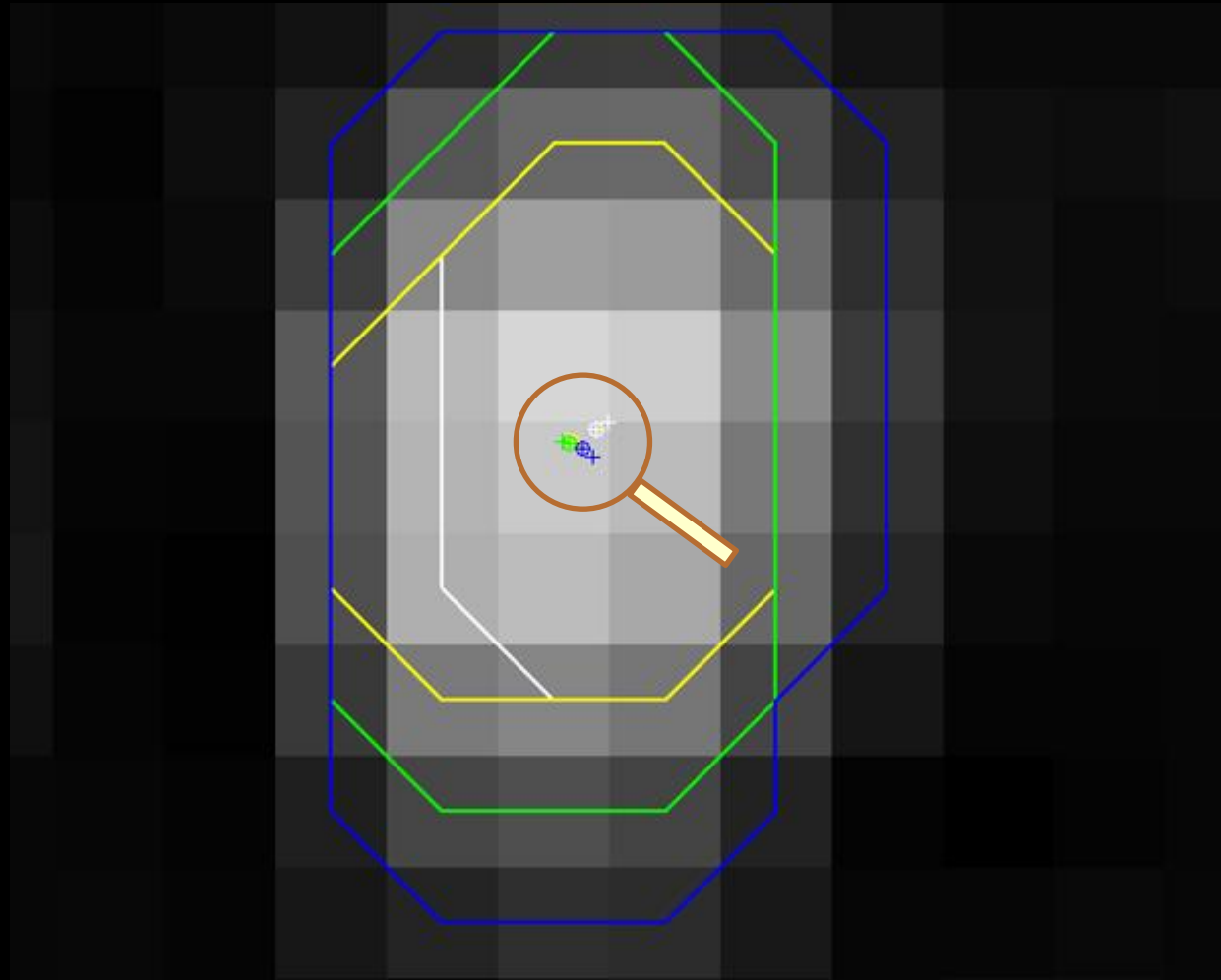
Yansıtıcı işaret yakalama



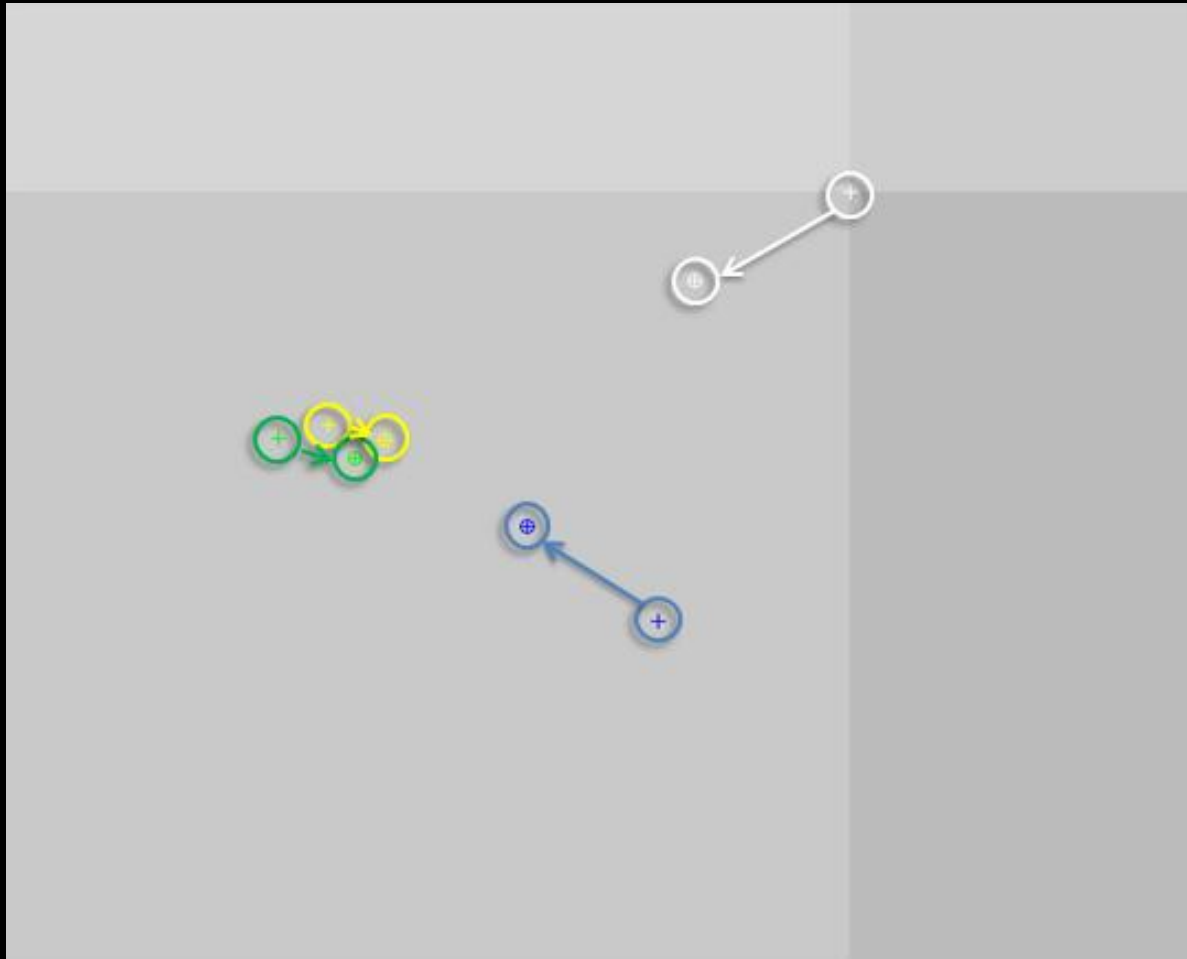
Yansıtıcı işaret yakalama



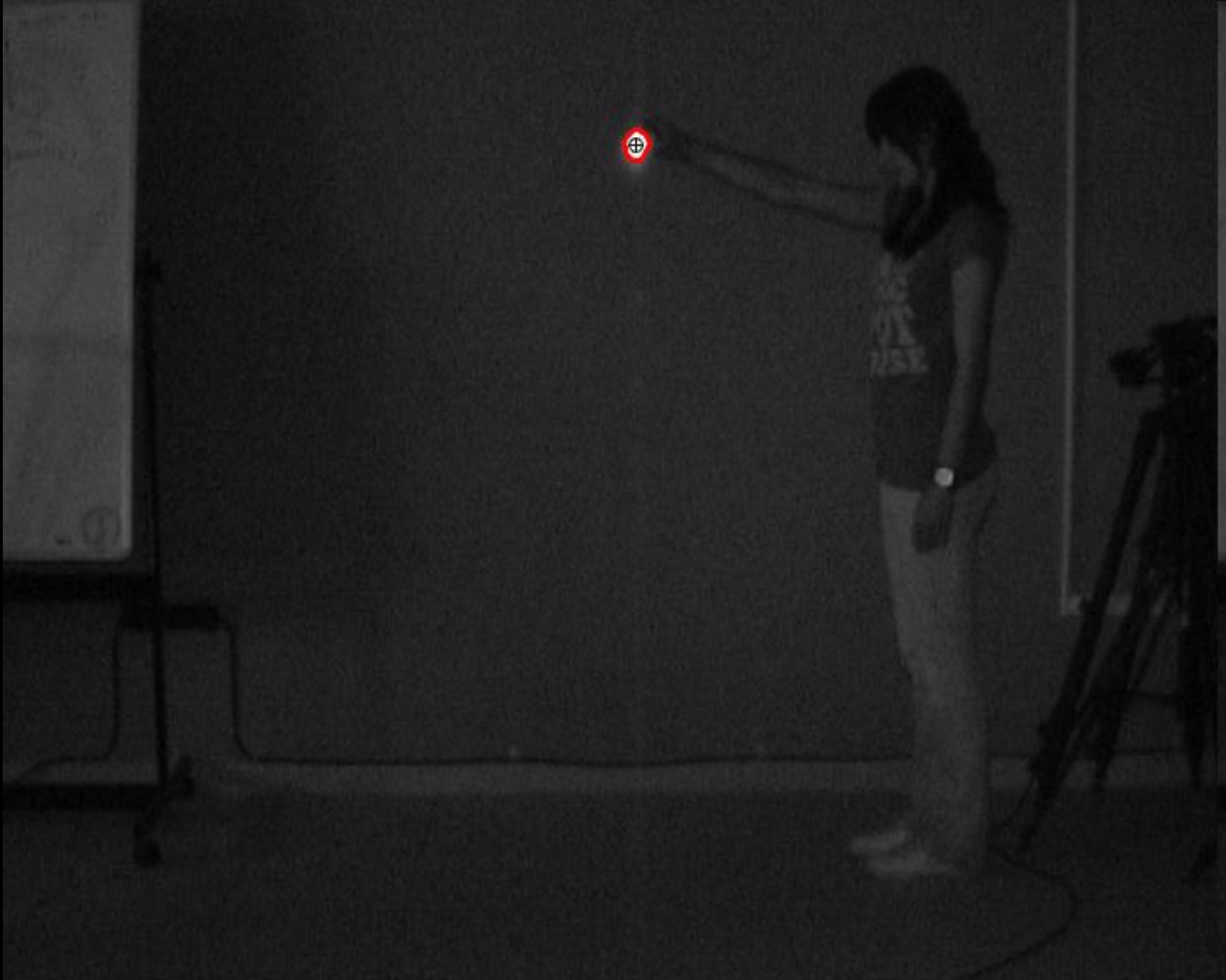
Yansıtıcı işaret yakalama



Yansıtıcı işaret yakalama



Sınıf Çalışması





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

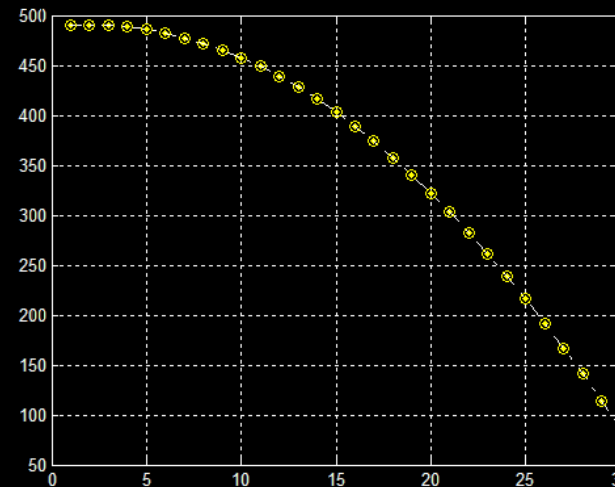
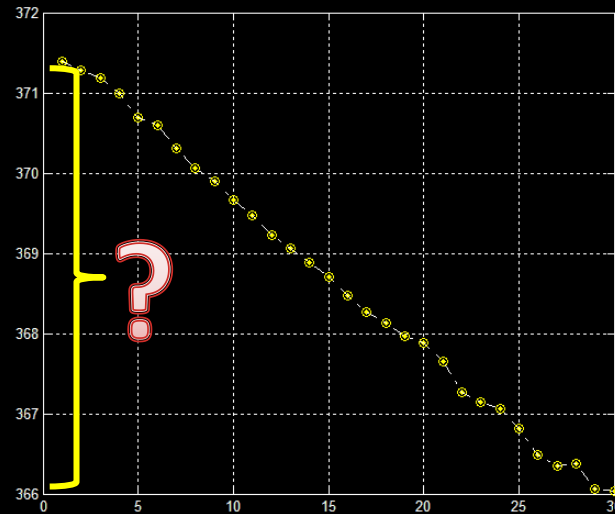
Kinematik Analiz



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

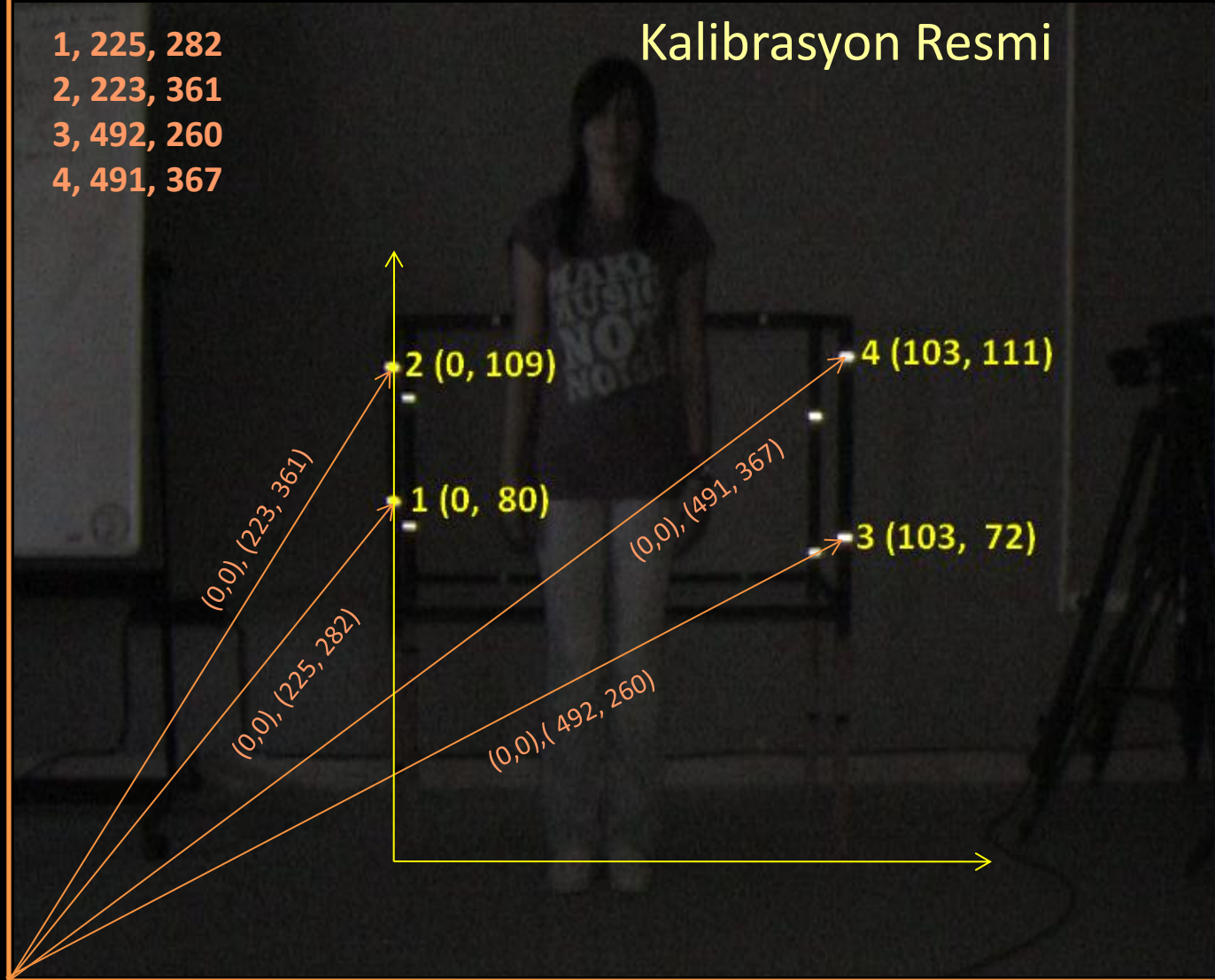
Kalibrasyon Resmi

1, 225, 282

2, 223, 361

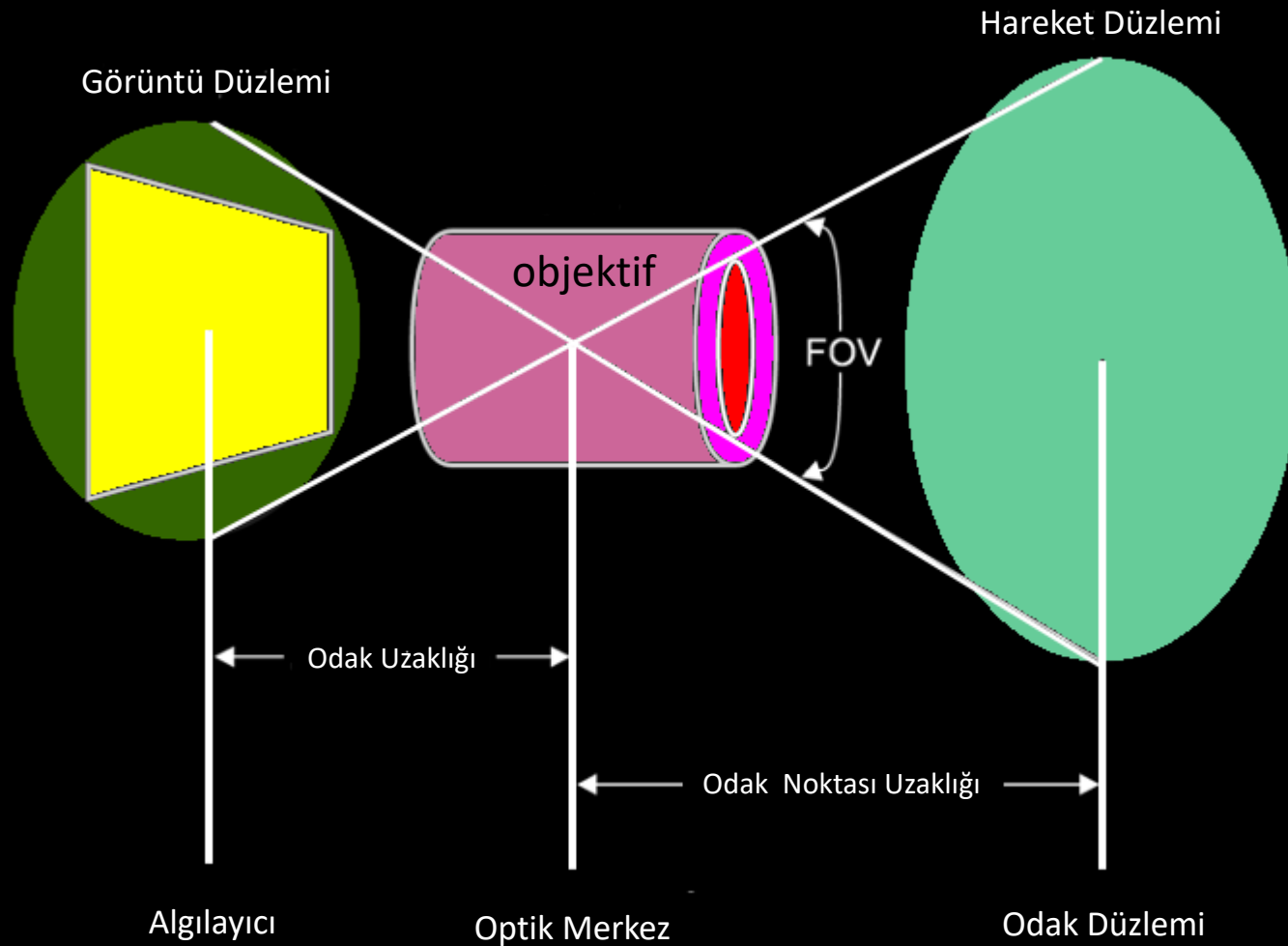
3, 492, 260

4, 491, 367



x

Kinematik Analiz



Kinematik Analiz

İki-Boyutlu Uyum Dönüşümü (2D Conformal Transformation)

Dört-parametrelili benzerlik dönüşümü olarak da bilinir

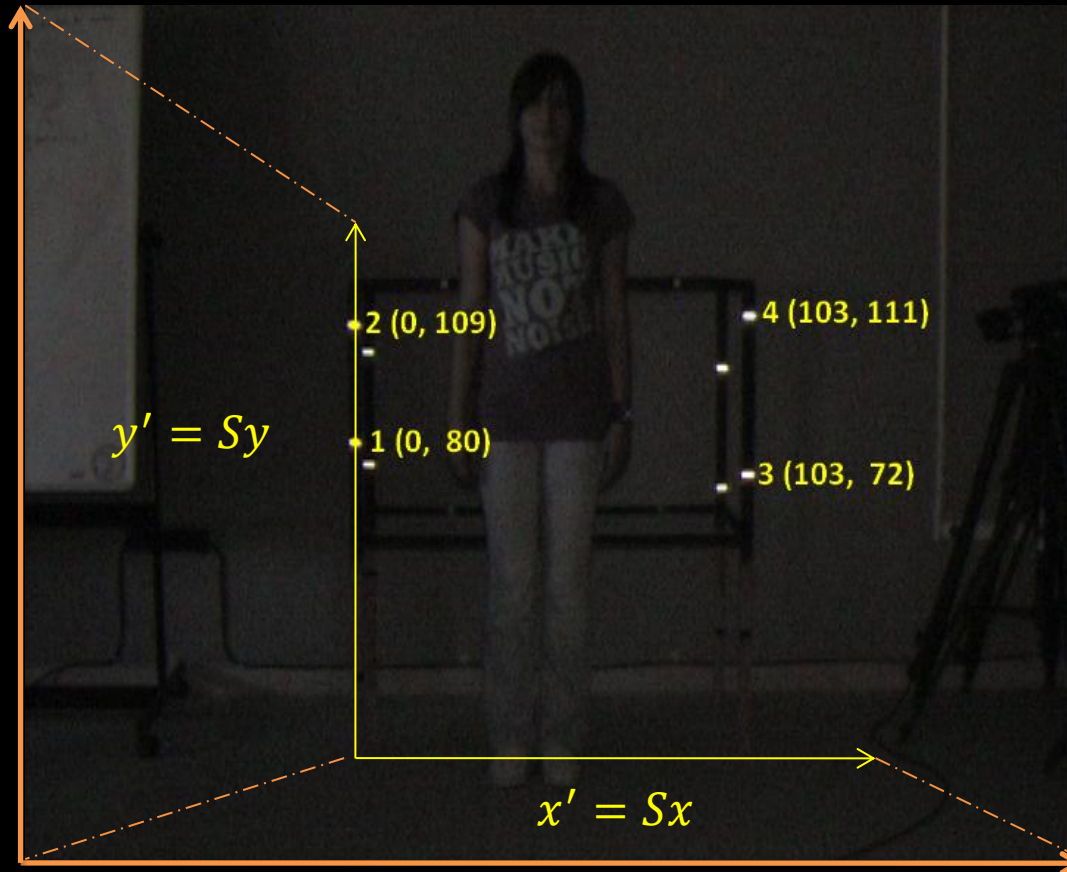
Bu dönüşüm 3 basamaktan oluşmaktadır:

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

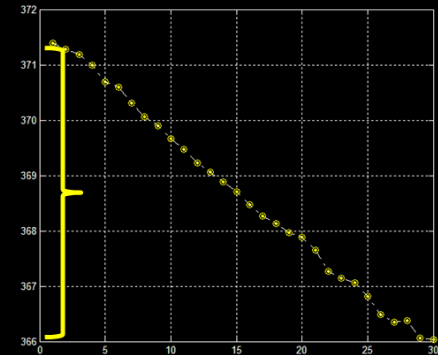
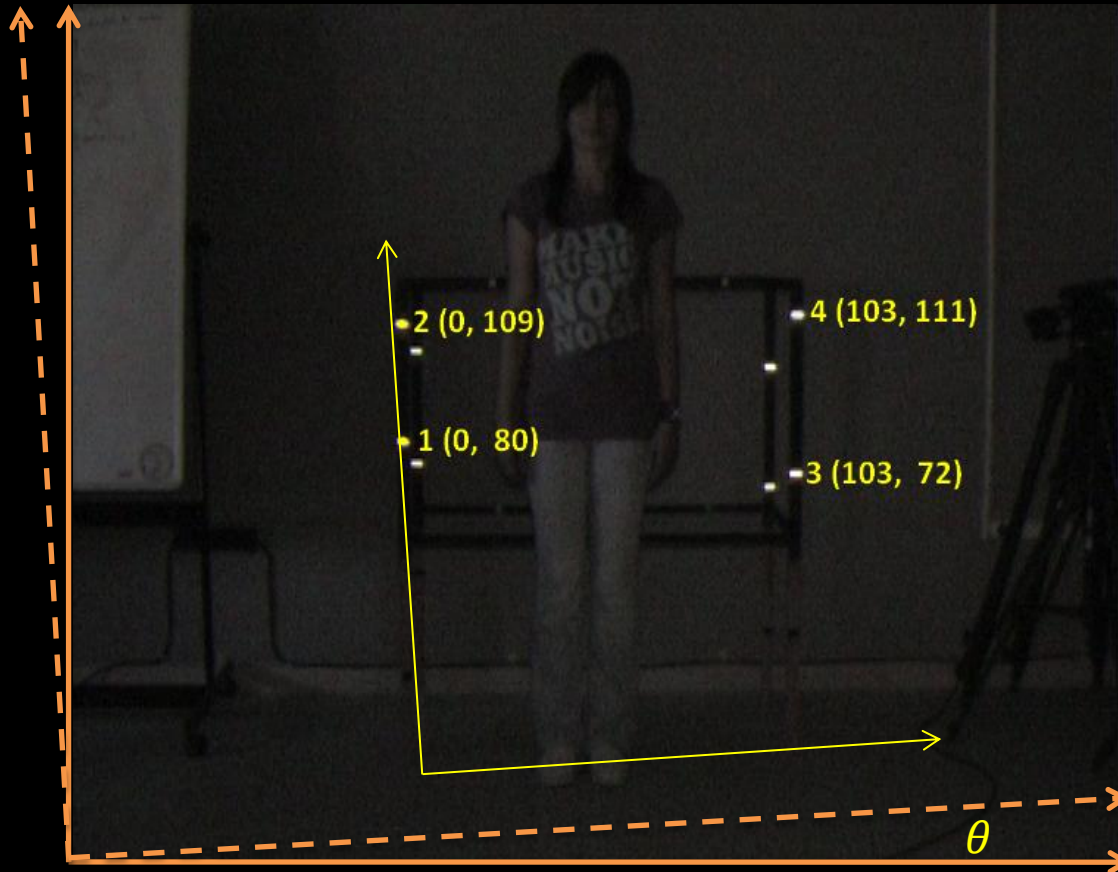
1.Basamak : Ölçekleme. (x,y) sisteminde tanımlanan çizgi uzunluklarını (X,Y) sisteminde de aynı uzunluğa getirmek için, (x,y) koordinatlarını ölçekleme faktörüyle (S) çarpmak gerekir.. Burada ölçeklendirilmiş koordinatlar x' ve y' dür:

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

Kinematik Analiz

İki-Boyutlu Uyum Dönüşümü

Dönme : iki sistemin referans eksenlerini paralel yapmak için



Kinematik Analiz

İki-Boyutlu Uyum Dönüşümü

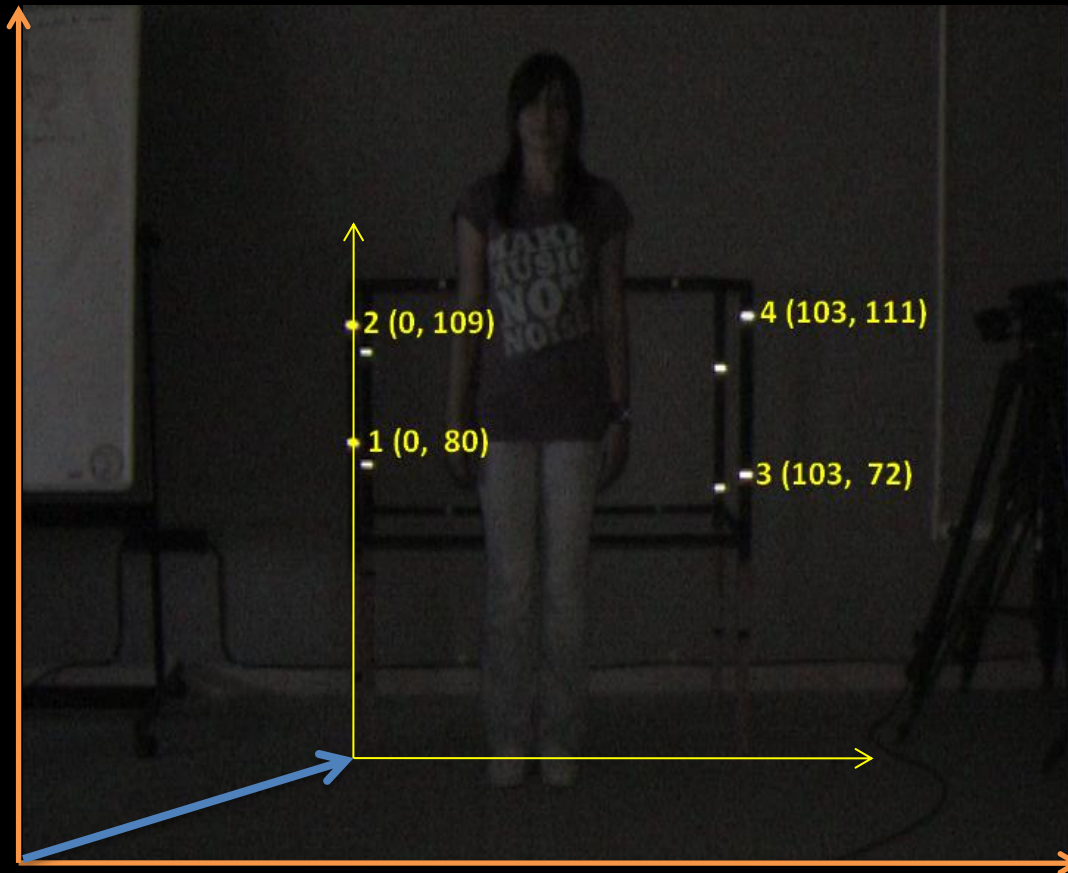
2.Basamak : Dönme. Ölçeklendirilmiş sistemde (x' , y') iki sistemin referans eksenlerini paralel yapmak için ölçeklendirilmiş sistemin (θ) açısı kadar döndürülmelidir. Burada döndürülmüş koordinatlar X' ve Y' dür:

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

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 , θ , 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;

$$A = \begin{bmatrix} x_a & -y_a & 1 & 0 \\ y_a & x_a & 0 & 1 \\ x_b & -y_b & 1 & 0 \\ y_b & x_b & 0 & 1 \\ x_c & -y_c & 1 & 0 \\ y_c & x_c & 0 & 1 \end{bmatrix}$$

$$X = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

$$L = \begin{bmatrix} X_A \\ Y_A \\ X_B \\ Y_B \\ X_C \\ Y_C \end{bmatrix}$$

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

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

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
# Calibration frame values
S = np.array([[0, 80],      # 1.
              [0, 109],    # 2.
              [103, 72],    # 3.
              [103, 111]]) # 4.
```

Calib_im.txt

```
225 282
223 361
492 260
491 367
```

```
I = np.loadtxt('calib_im.txt')
```

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

function

```
theta = np.rad2deg(np.arctan(x[1]/x[0]))
```

```
scale = x[0]/(np.cos(theta))
```

```
Tx = x[2]
```

```
Ty = x[3]
```



```
def calculate_conformal(I, S, method = 1):
    """ Conformal Map Transformation
    I : Image Coordinates of Calibration Points
    S : Space Coordinates of Calibration Points
    method : 1. LSM, 2. Psedou Inverse
    returns calibration coefficients"""

    rS, cS = S.shape
    rI, cI = I.shape
    # check the matrix dimensions
    if cS > 3 or cI > 2:
        raise ValueError ('Error in Matrix Dimensions')

    u = I[:,0] # Image coordiantes of calibration frame x values
    v = I[:,1] # Image coordiantes of calibration frame y values

    # Coefficients Matrix
    A1 = np.column_stack((u, -v, np.ones(rI), np.zeros(rI)))
    A2 = np.column_stack((v, u, np.zeros(rI), np.ones(rI)))
    A = np.vstack((A1, A2))
    # Right hand side of the equation
    b = np.reshape(S, -1, 'F') # Fortran vectorization

    # To solve Ax = b there are two approaches
    # 1. linear least squares
    # 2. pseudoinverse
    if method == 1:
        L = (np.linalg.lstsq(A, b, rcond=None))[0]
    elif method == 2:
        L = np.dot(np.linalg.pinv(A), b)
    else:
        print("Missing method : use (1) for Least Squares use (2) for Pseudoinverse")
        L = []
    return L
```

```
Ball_drop = np.loadtxt('calib im.txt')
```

```
H = calculate_reconformal(x, ball_drop)
```

```
def calculate_reconformal(x, I):  
    """ Reconstruction of 2D Positions  
        Image Coordinates of Marker Points : I  
        Conformal Coefficients              : P """
```

```
    rI, cI = I.shape
```

```
    u = I[:,0] # Image coordiantes of markers x values
```

```
    v = I[:,1] # Image coordiantes of markers y values
```

```
    # Coefficients Matrix
```

```
    A1 = np.column_stack((u, -v, np.ones(rI), np.zeros(rI)))
```

```
    A2 = np.column_stack((v, u, np.zeros(rI), np.ones(rI)))
```

```
    A = np.vstack((A1, A2))
```

```
    H = A@x
```

```
    return H.reshape(int(len(H)/2), 2, order='F')
```

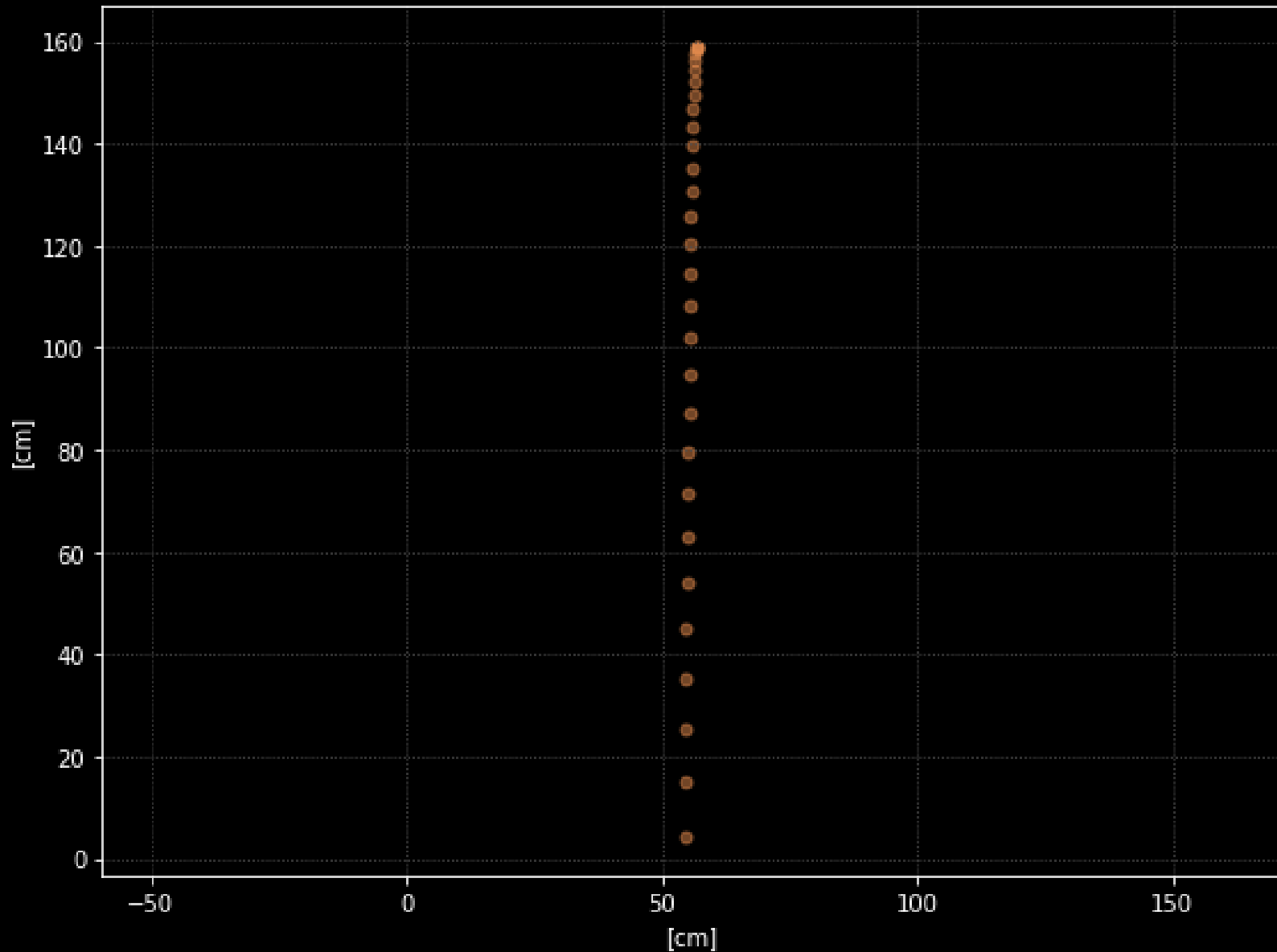
Ball_drop.txt

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



```
import matplotlib.pyplot as plt

fig, ax = plt.subplots() # a figure with a single Axes
ax.scatter(H[:,0], H[:,1], s=25, alpha=0.5)
ax.set_xlabel('[cm]')
ax.set_ylabel('[cm]')
ax.axis('equal')
ax.grid(True, linestyle=':')
fig.tight_layout()
plt.show()
```



Kinematik Analiz

Sonlu Farklar Analizi : Merkezden Fark metodu

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

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

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



```
import matplotlib.pyplot as plt

def calc_velocity(H, dt):
    vx = [(x2-x1)/(2*dt) for x1, x2 in zip(H[:-3,0], H[2:,0])]
    vy = [(y2-y1)/(2*dt) for y1, y2 in zip(H[:-3,1], H[2:,1])]

    return np.asarray([vx, vy]).T

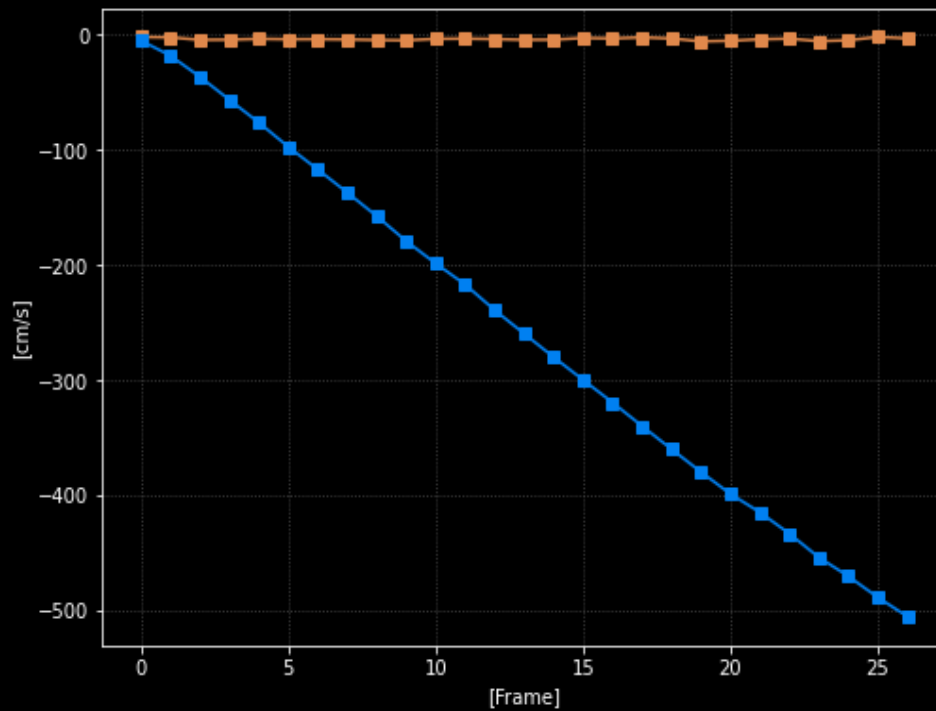
V = calc_velocity(H, 1/50)
A = calc_velocity(V, 1/50)

fig, ax = plt.subplots() # a figure with a single Axes
ax.plot(V[0])
ax.plot(V[1])
ax.set_xlabel('[Frame]')
ax.set_ylabel('[cm/s]')

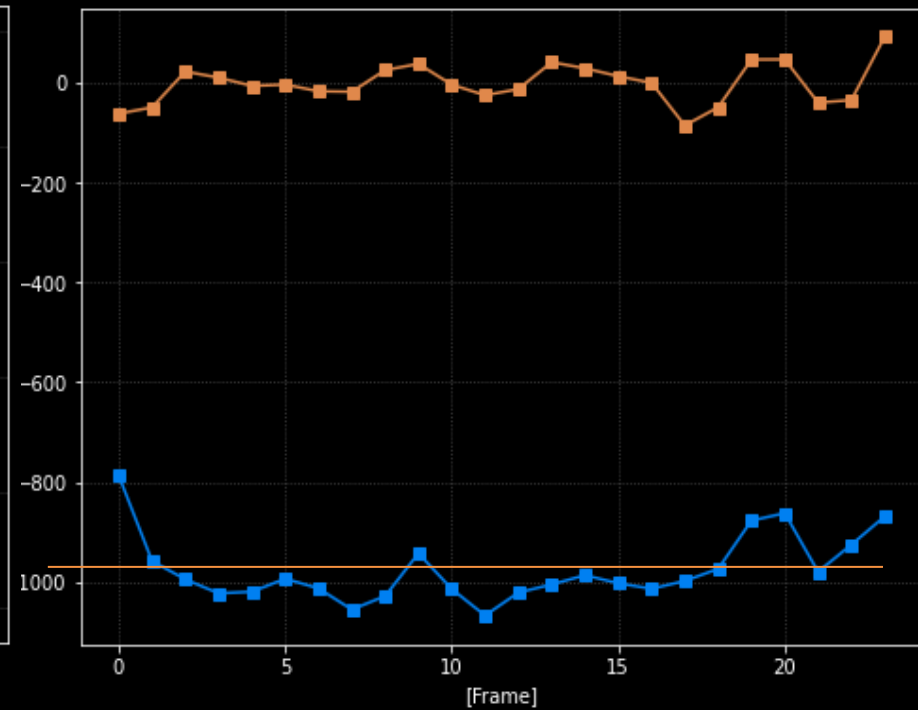
ax.grid(True, linestyle=':')
plt.show()
```

Kinematik Analiz

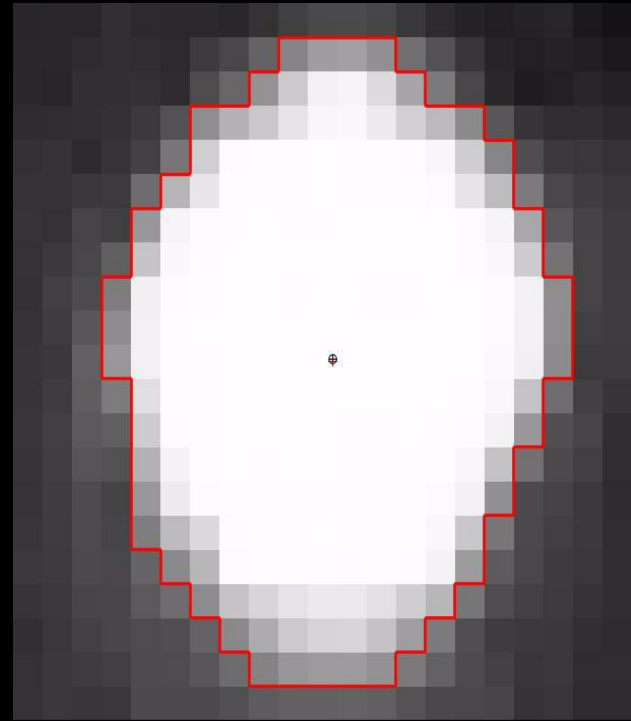
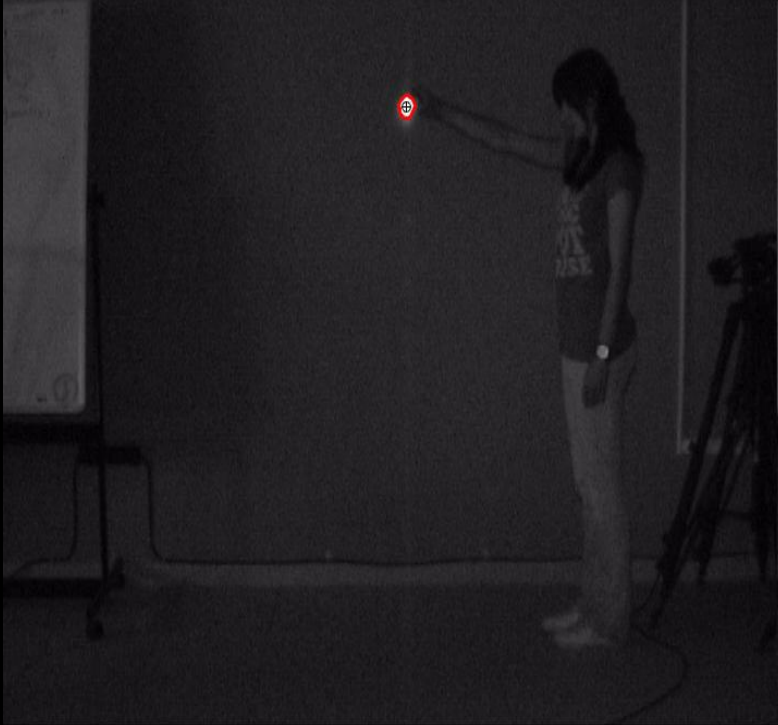
Hız



İvme

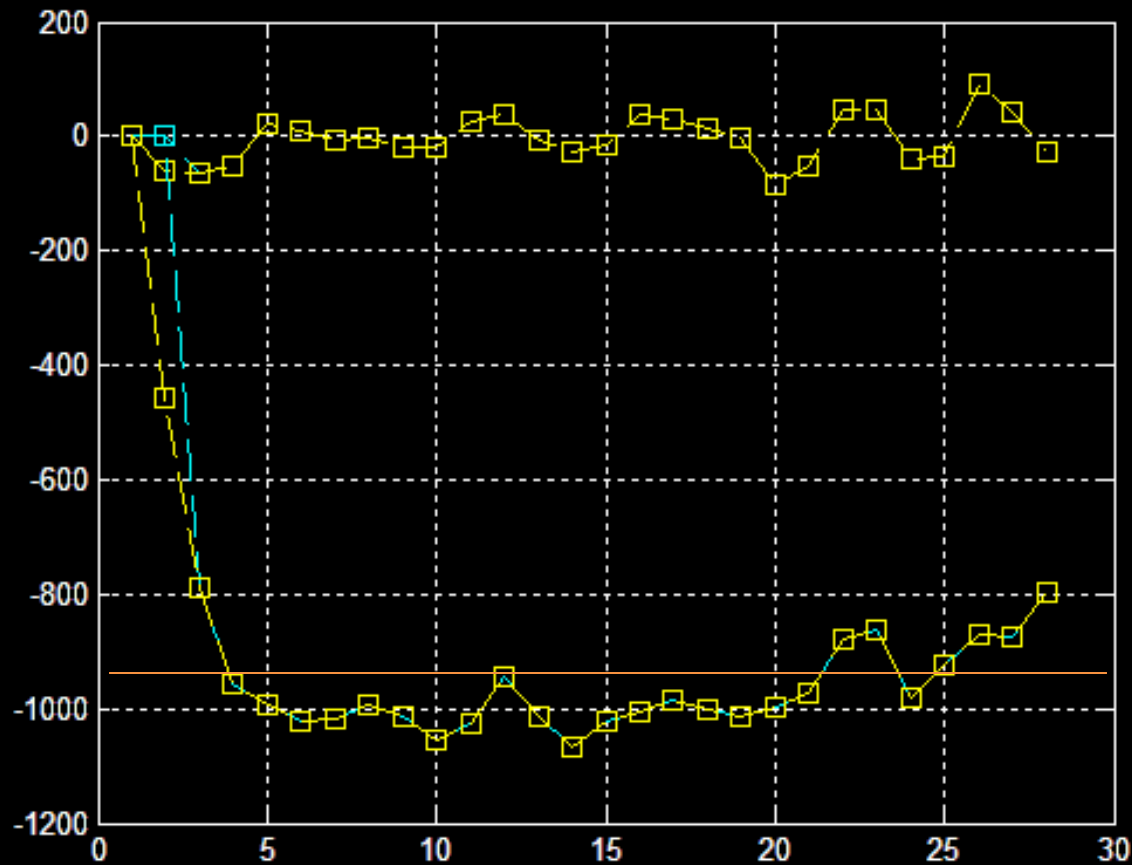


Kinematik Analiz



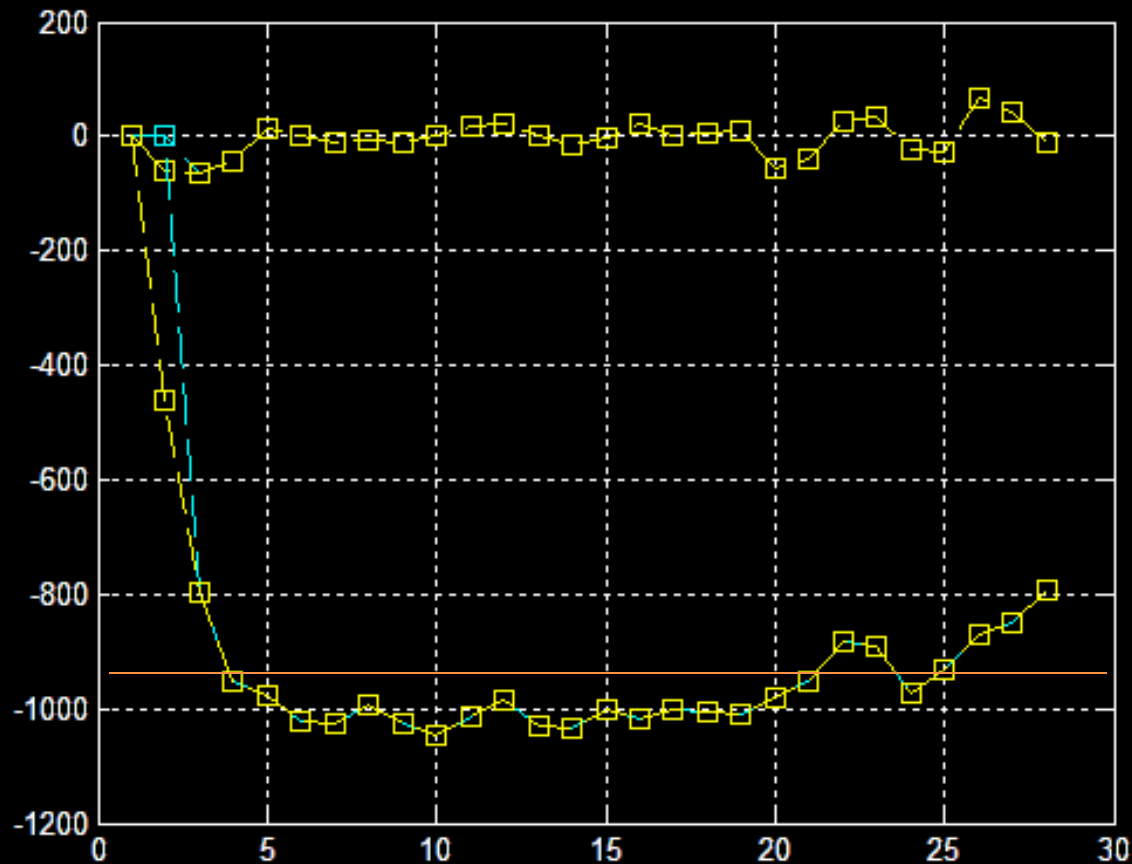
Kinematik Analiz

Kütle Merkezi



Kinematik Analiz

Ağırlıklı Kütle Merkezi



Ev Ödevi 4 Kalibrasyon

Geriye Salto hareketinde bulduğunuz eklem işaret koordinatlarını cm çeviriniz

