



# BCA 607 Hareket Analizi Sistemleri

## Matlab ile Görüntü İşleme 1



**SERDAR ARITAN**

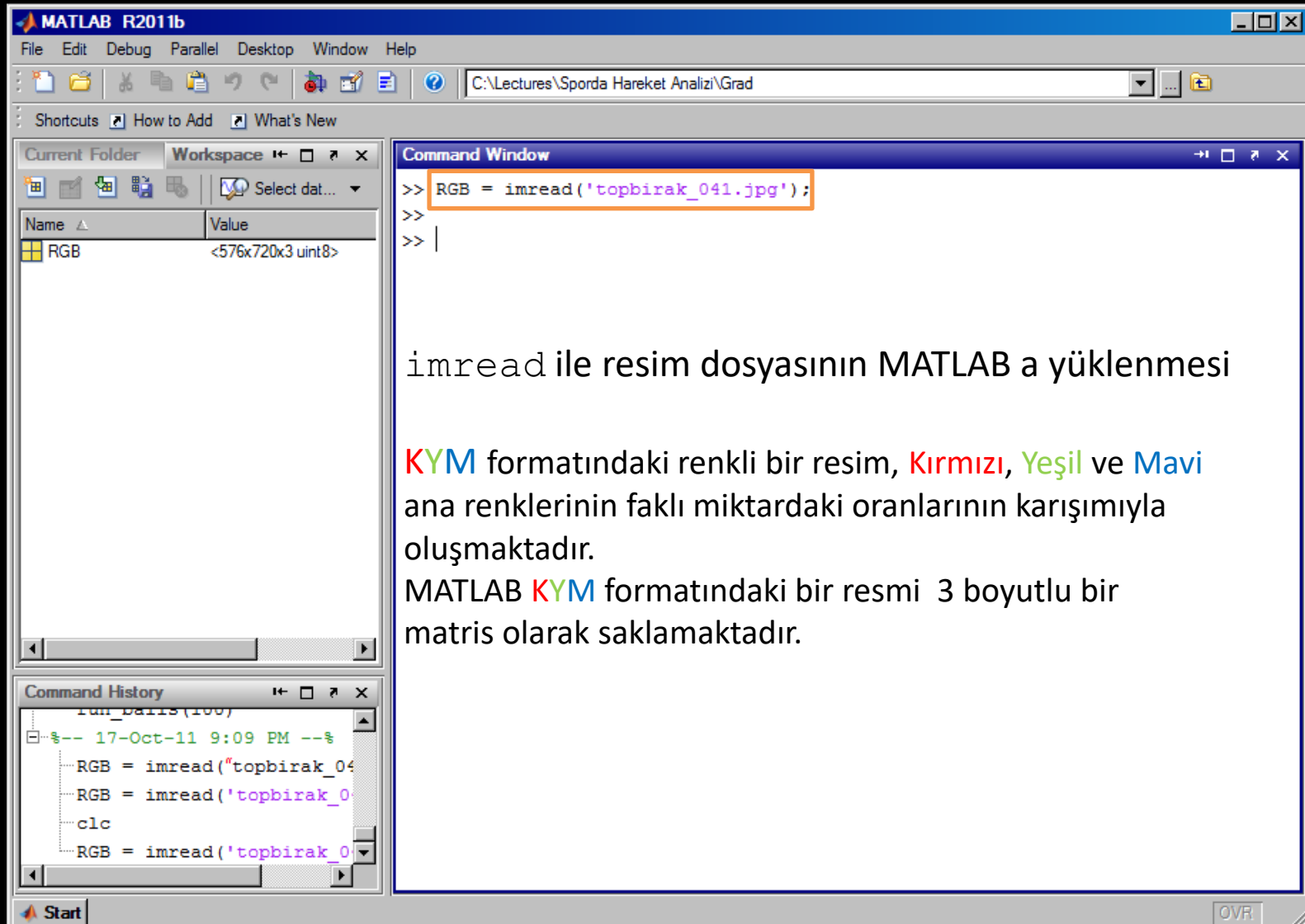
serdar.aritan@hacettepe.edu.tr

Biyomekanik Araştırma Grubu  
www.biomech.hacettepe.edu.tr  
Hacettepe Üniversitesi, Ankara, Türkiye  
www.hacettepe.edu.tr

## Yansıtıcı işaret yakalama



`RGB = imread('topbirak_041.jpg');`



The image shows a screenshot of the MATLAB R2011b interface. The Command Window on the right displays the command `RGB = imread('topbirak_041.jpg');` which has been executed. The Workspace window on the left shows a variable named `RGB` with a value of `<576x720x3 uint8>`. The Command History window at the bottom left shows the command `RGB = imread('topbirak_041.jpg');` being entered.

`imread` ile resim dosyasının MATLAB a yüklenmesi

KYM formatındaki renkli bir resim, Kırmızı, Yeşil ve Mavi ana renklerinin farklı miktardaki oranlarının karışımıyla oluşmaktadır.

MATLAB KYM formatındaki bir resmi 3 boyutlu bir matris olarak saklamaktadır.



$R = \text{RGB}(:, :, 1);$

**MATLAB R2011b**

File Edit View Graphics Debug Parallel Desktop Window Help

C:\Lectures\Sporda Hareket Analizi\Grad

Shortcuts How to Add What's New

**Current Folder** **Workspace** **Command Window** **Command History**

**Workspace**

Name	Value
R	<576x720 uint8>
RGB	<576x720x3 uint8>

**Command Window**

```
>> RGB = imread('topbirak_041.jpg');  
>>  
>> R = RGB(:, :, 1);  
>>
```

**Command History**

```
... RGB = imread('topbirak_041.jpg');  
... RGB = imread('topbirak_041.jpg');  
... clc  
... RGB = imread('topbirak_041.jpg');  
... R = RGB(:, :, 1);
```

Start

Workspace de bulunan resimin 1 katmanını  
(bu durumda Kırmızı bileşeni) R isimli bir  
değişkene atayalım.



R üzerine çift tıklayın

**MATLAB R2011b**

File Edit View Graphics Debug Parallel Desktop Window Help

C:\Lectures\Sporda Hareket Analizi\Grad

Shortcuts How to Add What's New

**Current Folder** **Workspace**

Name Value

R <576x720 uint8>

RGB <576x720x3 uint8>

**Variable Editor - R**

Stack: Base No valid plots for: R(1,1)

R <576x720 uint8>

	1	2	3	4	5	6	7	8
1	48	47	51	50	48	44	46	4
2	48	46	48	43	39	38	46	4
3	47	46	49	46	44	45	47	4
4	47	47	50	49	49	50	48	4
5	49	51	56	54	53	54	53	5
6	50	57	63	60	57	54	55	5
7	56	56	60	62	59	58	57	5
8	58	54	56	61	63	60	60	5
9	60	58	59	63	62	60	60	6
10	61	61	65	64	62	60	60	6
11	63	64	66	64	64	64	66	6
12	64	66	66	63	64	66	70	6
13	64	65	65	60	63	65	66	6

**Command History**

```
RGB = imread('topbirak_041.jpg');  
RGB = imread('topbirak_041.jpg');  
clc  
RGB = imread('topbirak_041.jpg');  
R = RGB(:,:,1);
```

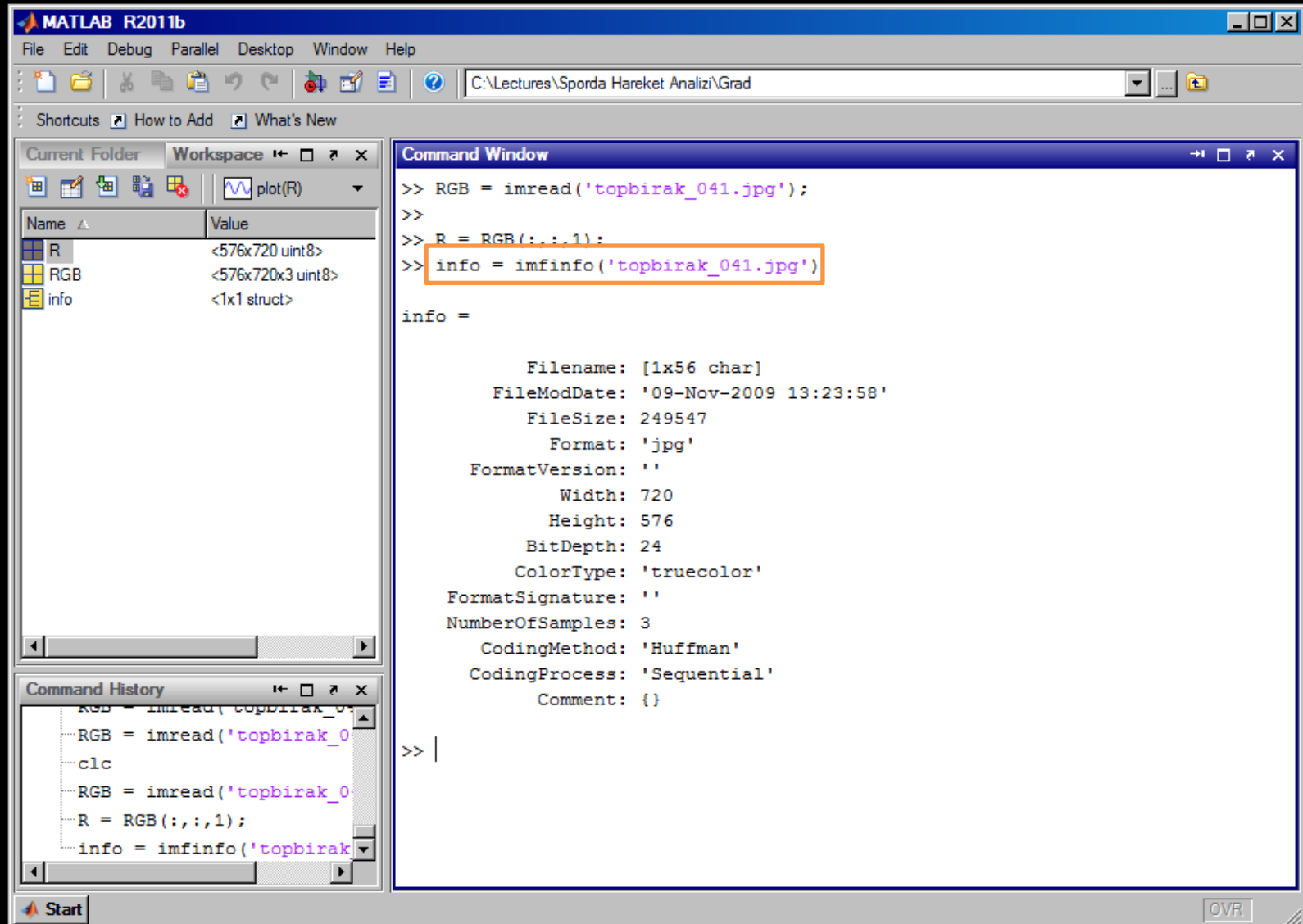
**Command Window**

```
>> RGB = imread('topbirak_041.jpg');  
>>  
>> R = RGB(:,:,1);  
>>
```



Resim hakkında bilgi almak için;

```
info = iminfo('topbirak_041.jpg')
```



The image shows a MATLAB R2011b interface with the following components:

- Current Folder:** C:\Lectures\Sporda Hareket Analizi\Grad
- Workspace:** Contains variables R (576x720 uint8), RGB (576x720x3 uint8), and info (1x1 struct).
- Command Window:** Displays the execution of the following code:

```
>> RGB = imread('topbirak_041.jpg');  
>>  
>> R = RGB(:,:,1);  
>> info = iminfo('topbirak_041.jpg')
```

The line `info = iminfo('topbirak_041.jpg')` is highlighted with an orange box. Below the code, the output for `info` is displayed:

```
info =  
  
    Filename: [1x56 char]  
    FileModDate: '09-Nov-2009 13:23:58'  
    FileSize: 249547  
    Format: 'jpg'  
    FormatVersion: ''  
    Width: 720  
    Height: 576  
    BitDepth: 24  
    ColorType: 'truecolor'  
    FormatSignature: ''  
    NumberOfSamples: 3  
    CodingMethod: 'Huffman'  
    CodingProcess: 'Sequential'  
    Comment: {}
```
- Command History:** Shows the sequence of commands entered in the Command Window.



## Resim göstermek için imshow

The image shows the MATLAB R2011b interface. The Command Window contains the following code:

```
>> RGB = imread('topbirak_041.jpg');  
>> R = RGB(:,:,1);  
>> imshow(RGB)  
>>
```

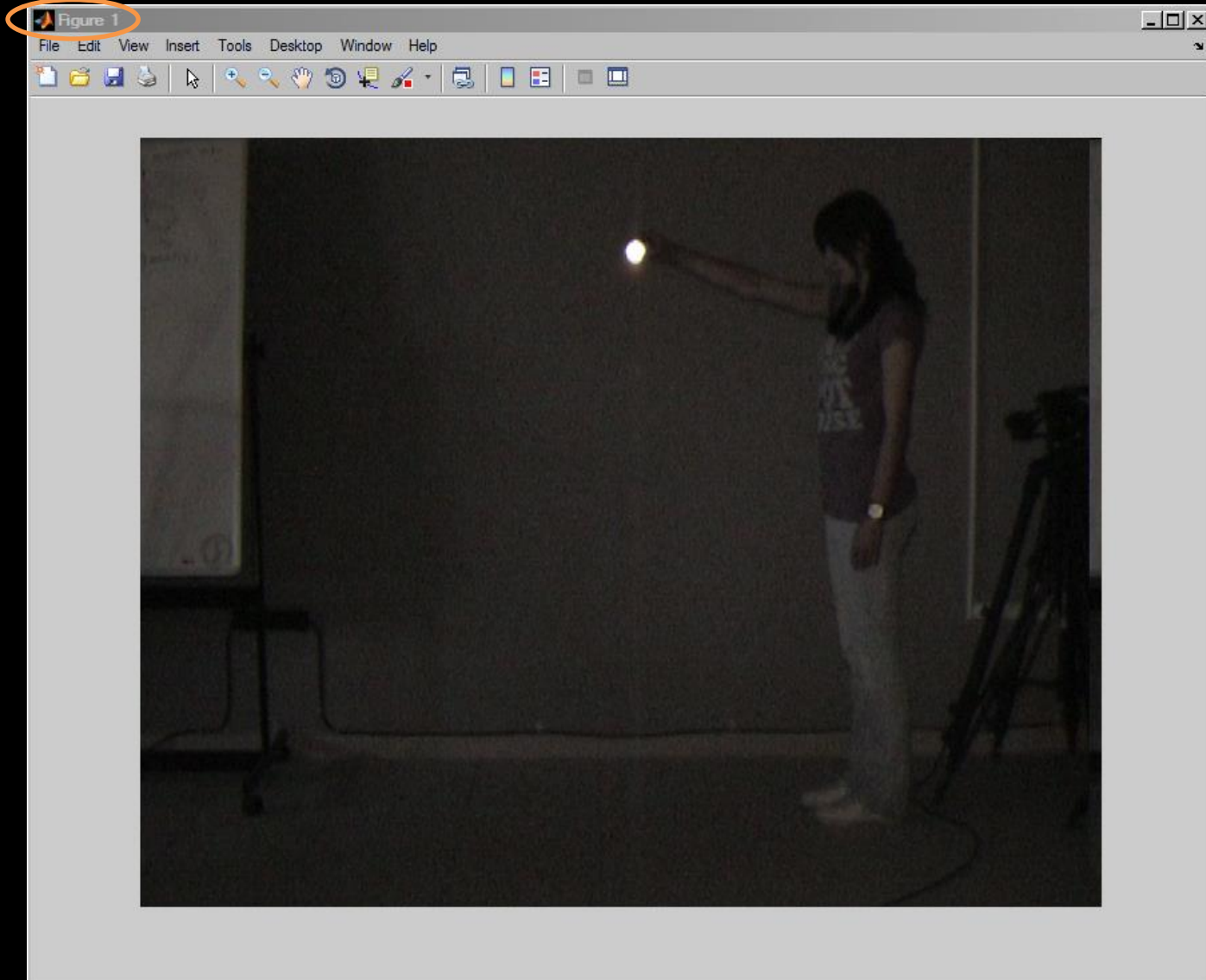
The Workspace window shows the following variables:

Name	Value
R	<576x720 uint8>
RGB	<576x720x3 uint8>

The Command History window shows the following commands:

```
RGB = imread('topbirak_041.jpg');  
clc  
RGB = imread('topbirak_041.jpg');  
R = RGB(:,:,1);  
imshow(RGB)
```

Yeni bir figure penceresinde resim gözükür





## Rgb2gray fonksiyonu

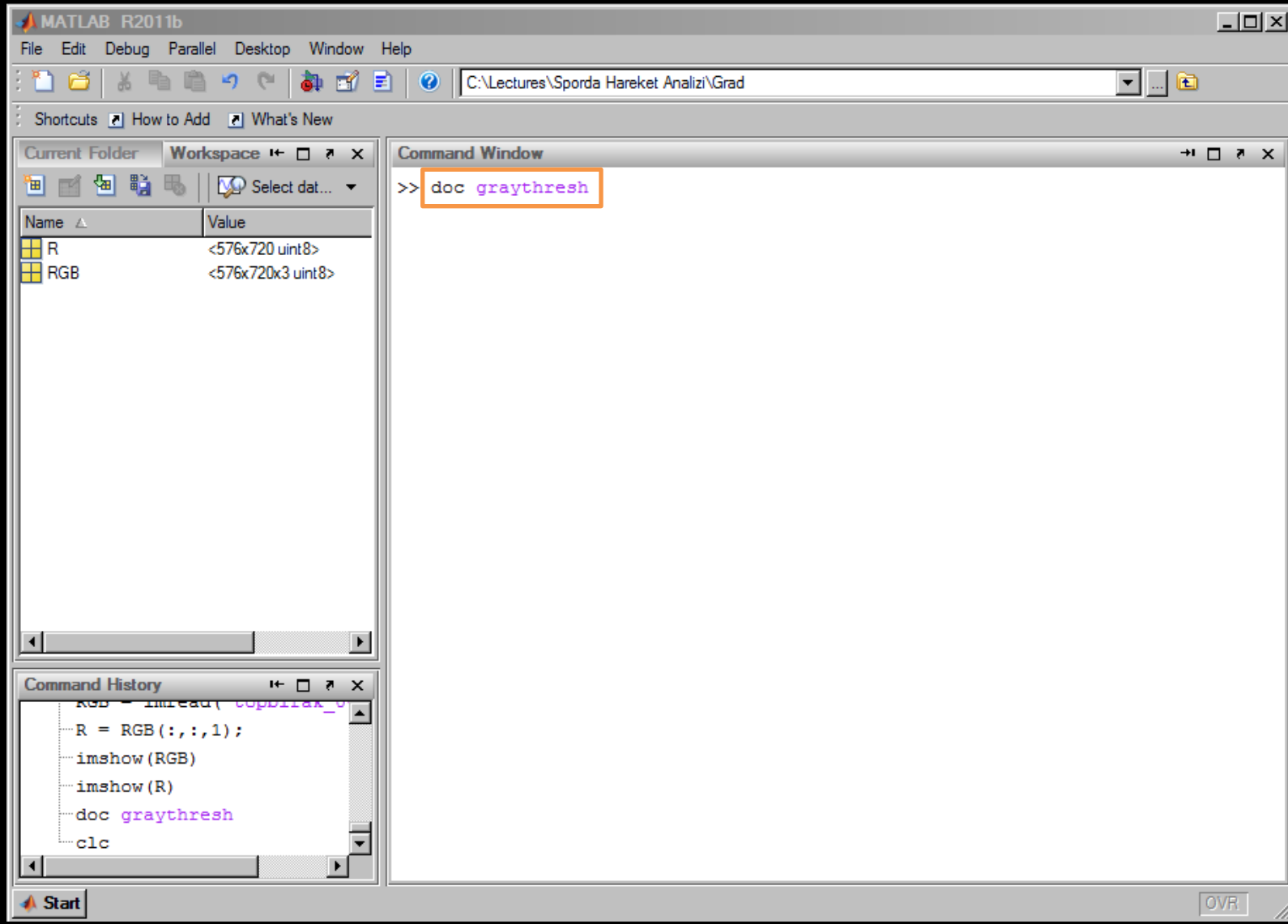
Convert the RGB colormap to a grayscale colormap and redisplay the image.

```
I = rgb2gray(RGB) ;  
imshowpair(I, RGB, 'montage')
```

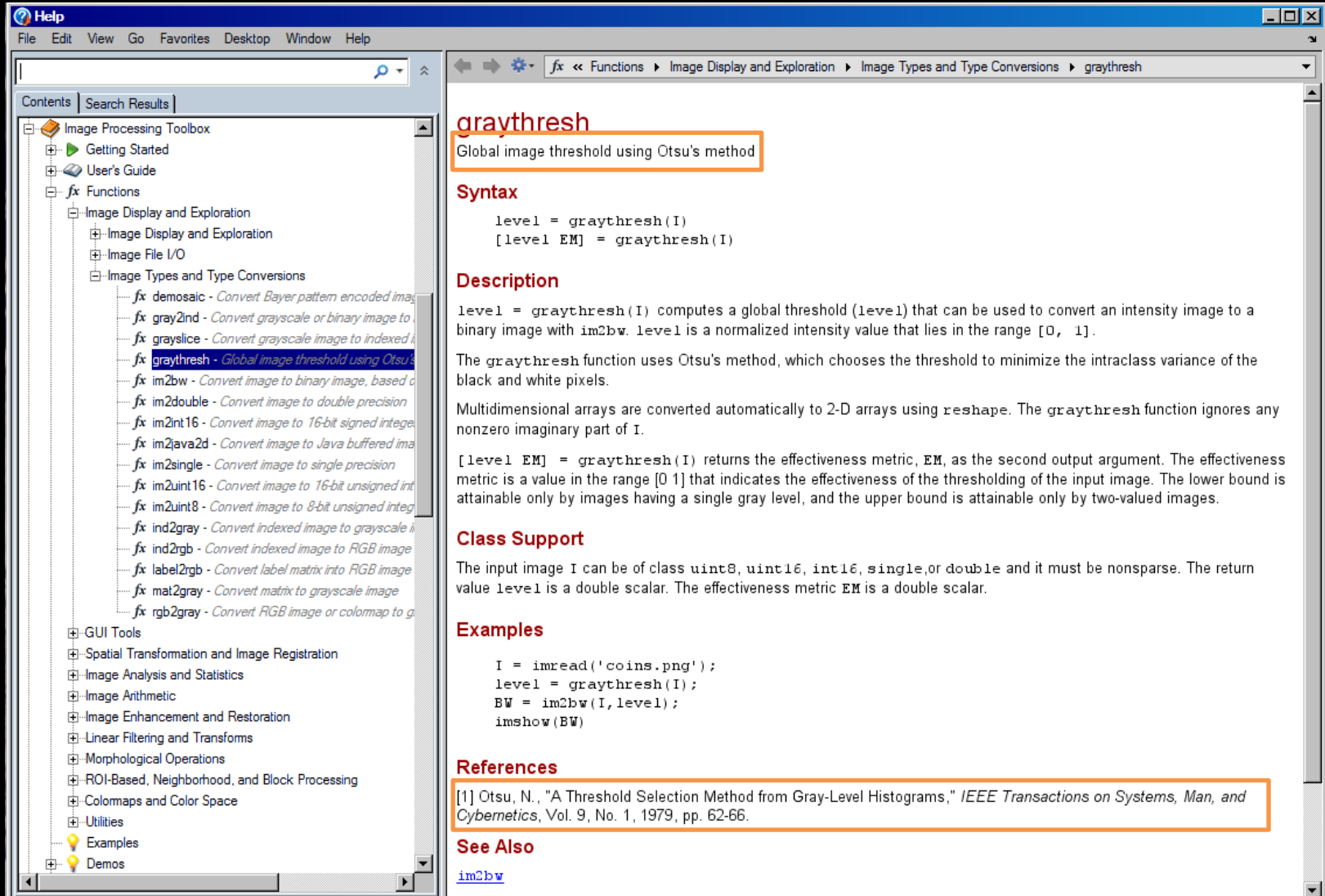




## Bir matlab fonksiyonu hakkında bilgi almak : doc



## graythresh fonksiyonu hakkında bilgi



**graythresh**  
Global image threshold using Otsu's method

**Syntax**

```
level = graythresh(I)  
[level EM] = graythresh(I)
```

**Description**

`level = graythresh(I)` computes a global threshold (`level`) that can be used to convert an intensity image to a binary image with `im2bw`. `level` is a normalized intensity value that lies in the range `[0, 1]`.

The `graythresh` function uses Otsu's method, which chooses the threshold to minimize the intraclass variance of the black and white pixels.

Multidimensional arrays are converted automatically to 2-D arrays using `reshape`. The `graythresh` function ignores any nonzero imaginary part of `I`.

`[level EM] = graythresh(I)` returns the effectiveness metric, `EM`, as the second output argument. The effectiveness metric is a value in the range `[0 1]` that indicates the effectiveness of the thresholding of the input image. The lower bound is attainable only by images having a single gray level, and the upper bound is attainable only by two-valued images.

**Class Support**

The input image `I` can be of class `uint8`, `uint16`, `int16`, `single`, or `double` and it must be nonsparse. The return value `level` is a double scalar. The effectiveness metric `EM` is a double scalar.

**Examples**

```
I = imread('coins.png');  
level = graythresh(I);  
BW = im2bw(I, level);  
imshow(BW)
```

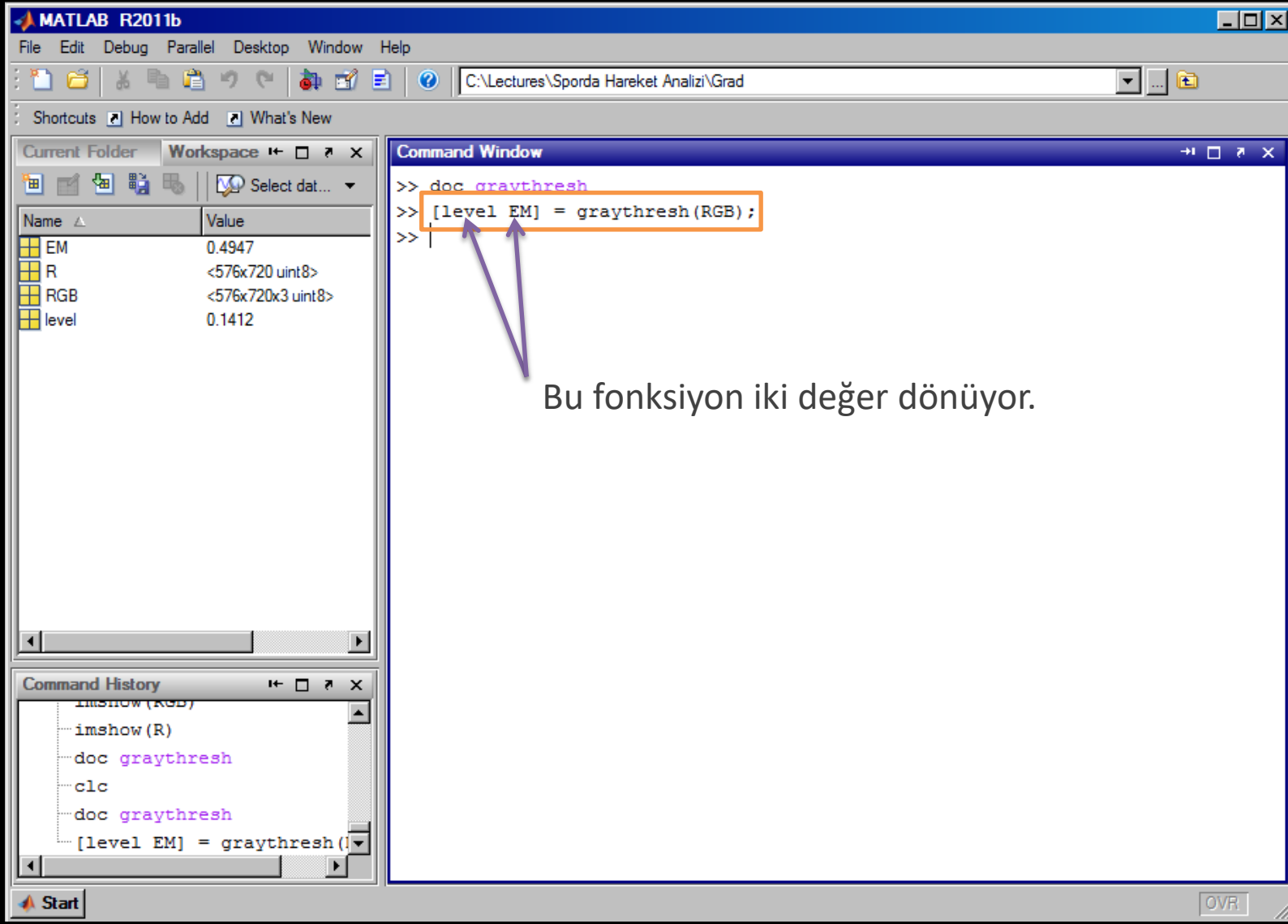
**References**

[1] Otsu, N., "A Threshold Selection Method from Gray-Level Histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 9, No. 1, 1979, pp. 62-66.

**See Also**

[im2bw](#)

## graythresh fonksiyonunun kullanımı



Command Window

```
>> doc graythresh  
>> [level EM] = graythresh(RGB);  
>>
```

Bu fonksiyon iki değer dönüyor.

Workspace

Name	Value
EM	0.4947
R	<576x720 uint8>
RGB	<576x720x3 uint8>
level	0.1412

Command History

```
imshow(RGB)  
imshow(R)  
doc graythresh  
clc  
doc graythresh  
[level EM] = graythresh(RGB);
```



## im2bw fonksiyonu hakkında bilgi

The image shows the MATLAB R2011b interface. The Command Window on the right contains the following code:

```
>> doc graythresh  
>> [level EM] = graythresh(RGB);  
>> doc im2bw  
>>
```

The Workspace window on the left shows the following variables:

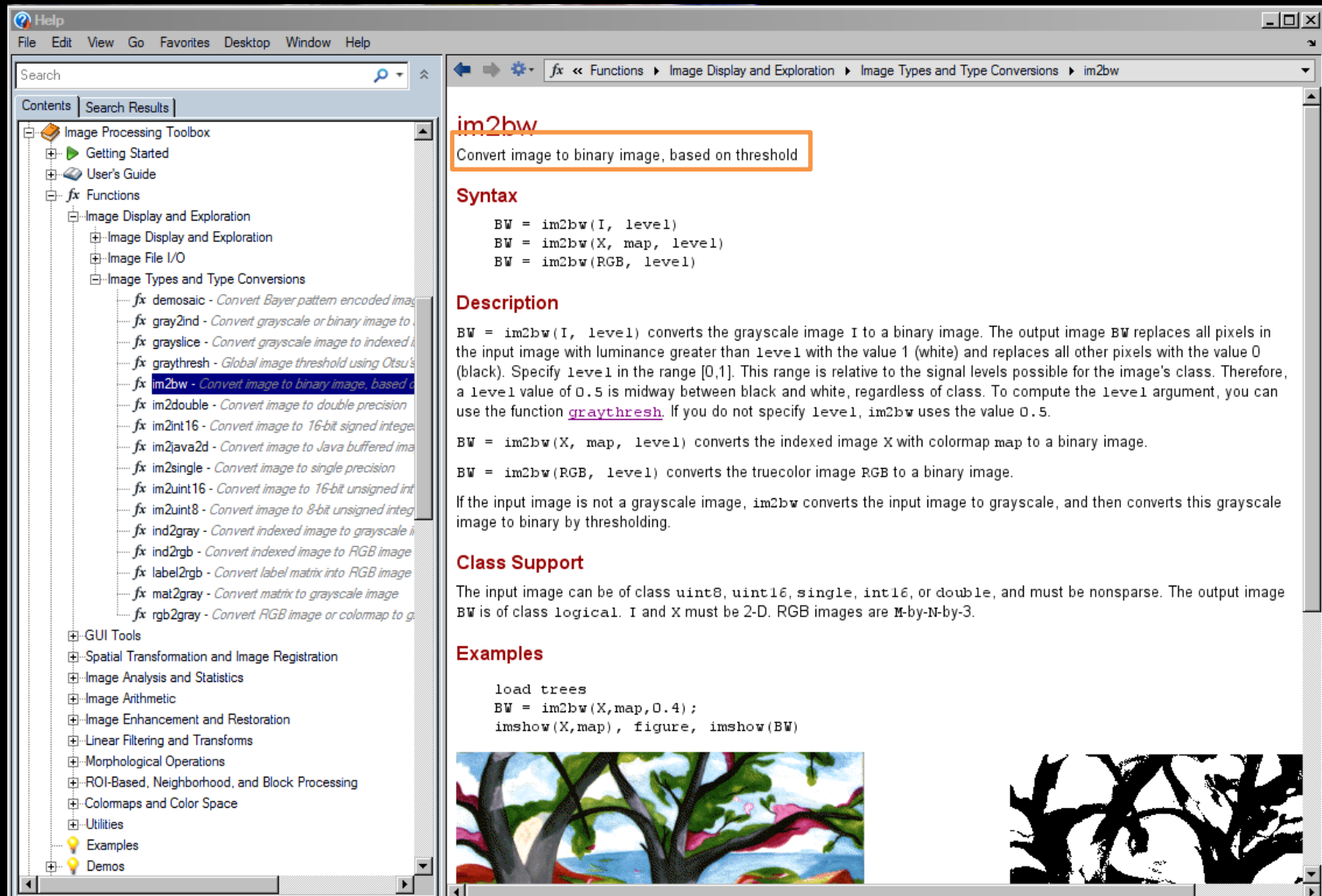
Name	Value
EM	0.4947
R	<576x720 uint8>
RGB	<576x720x3 uint8>
level	0.1412

The Command History window at the bottom left shows the following commands:

```
imshow(R)  
doc graythresh  
clc  
doc graythresh  
[level EM] = graythresh(R)  
doc im2bw
```



## im2bw fonksiyonu hakkında bilgi



The image shows a screenshot of the MATLAB Help window. The left pane displays the 'Contents' tree with 'Image Processing Toolbox' expanded, showing 'Image Types and Type Conversions' and 'im2bw'. The right pane shows the 'im2bw' function documentation, which includes the following sections:

### im2bw

Convert image to binary image, based on threshold

#### Syntax

```
BW = im2bw(I, level)
BW = im2bw(X, map, level)
BW = im2bw(RGB, level)
```

#### Description

**BW = im2bw(I, level)** converts the grayscale image I to a binary image. The output image BW replaces all pixels in the input image with luminance greater than level with the value 1 (white) and replaces all other pixels with the value 0 (black). Specify level in the range [0,1]. This range is relative to the signal levels possible for the image's class. Therefore, a level value of 0.5 is midway between black and white, regardless of class. To compute the level argument, you can use the function [graythresh](#). If you do not specify level, im2bw uses the value 0.5.

**BW = im2bw(X, map, level)** converts the indexed image X with colormap map to a binary image.

**BW = im2bw(RGB, level)** converts the truecolor image RGB to a binary image.


If the input image is not a grayscale image, im2bw converts the input image to grayscale, and then converts this grayscale image to binary by thresholding.

#### Class Support

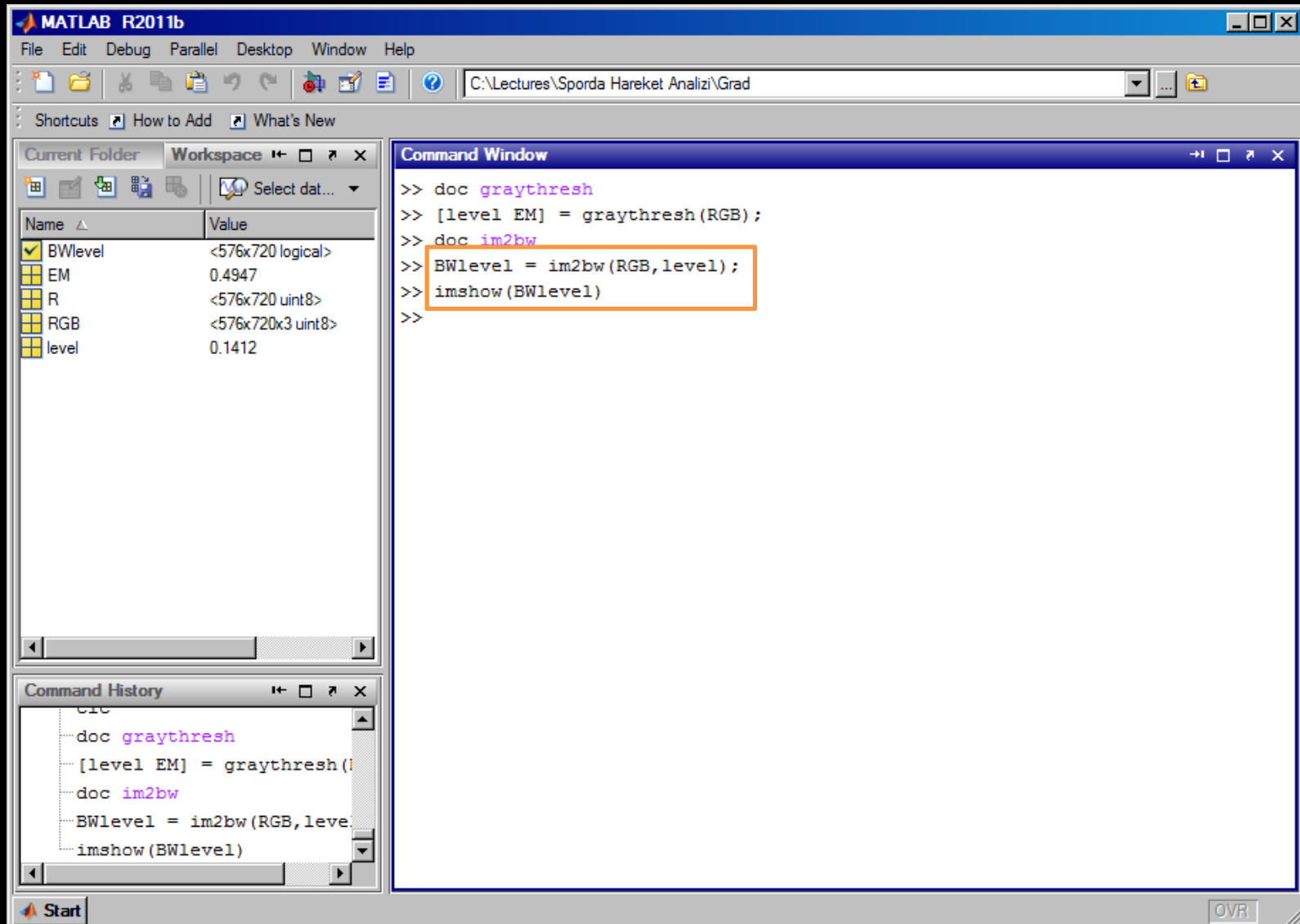
The input image can be of class `uint8`, `uint16`, `single`, `int16`, or `double`, and must be nonsparse. The output image BW is of class `logical`. I and X must be 2-D. RGB images are M-by-N-by-3.

#### Examples

```
load trees
BW = im2bw(X,map,0.4);
imshow(X,map), figure, imshow(BW)
```



## im2bw fonksiyonunun level değişkeniyle kullanımı



The image shows the MATLAB R2011b interface. The Command Window displays the following code:

```
>> doc graythresh
>> [level EM] = graythresh(RGB);
>> doc im2bw
>> BWlevel = im2bw(RGB,level);
>> imshow(BWlevel)
>>
```

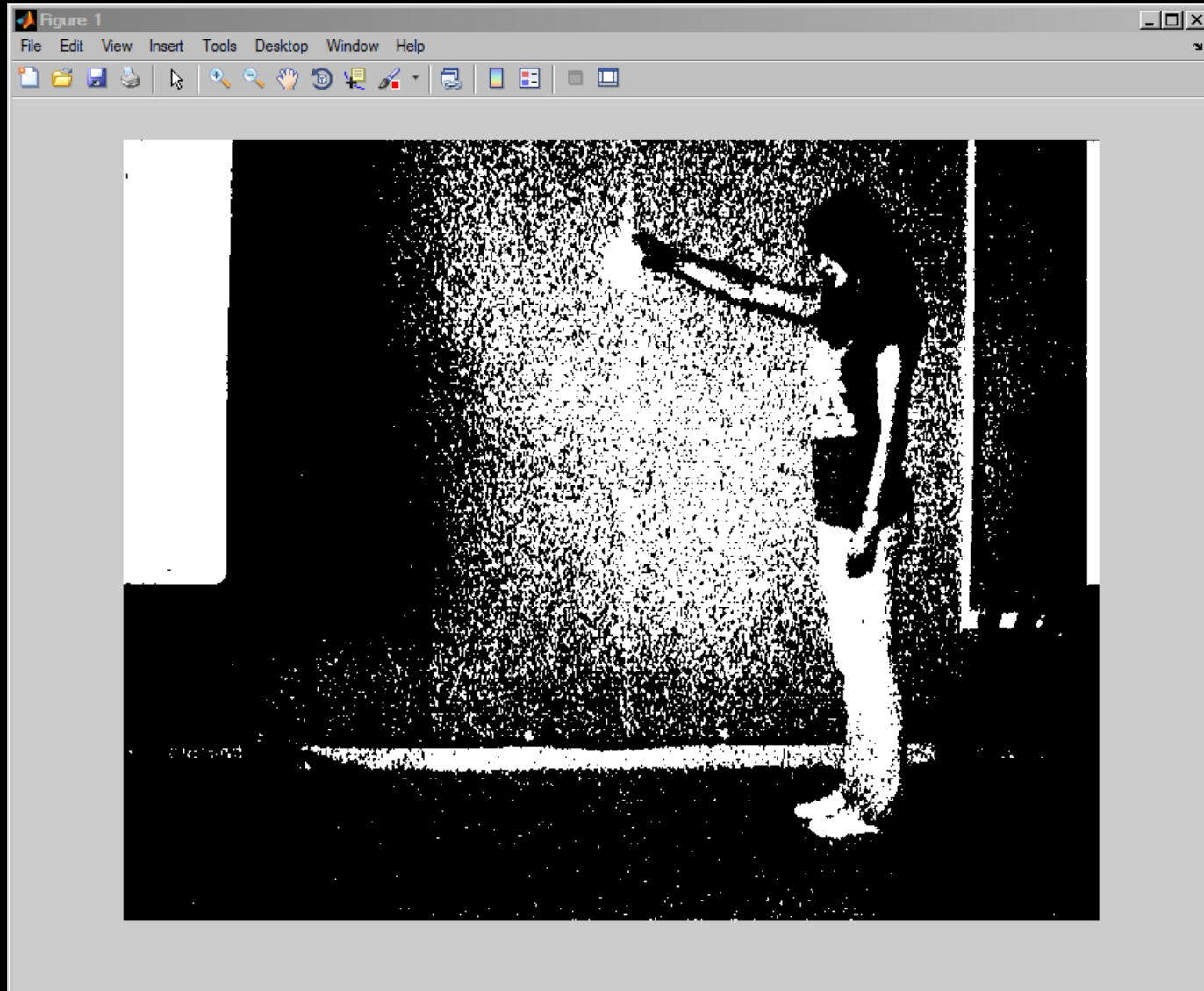
The line `BWlevel = im2bw(RGB,level);` is highlighted with an orange box. The Workspace window shows the following variables:

Name	Value
BWlevel	<576x720 logical>
EM	0.4947
R	<576x720 uint8>
RGB	<576x720x3 uint8>
level	0.1412

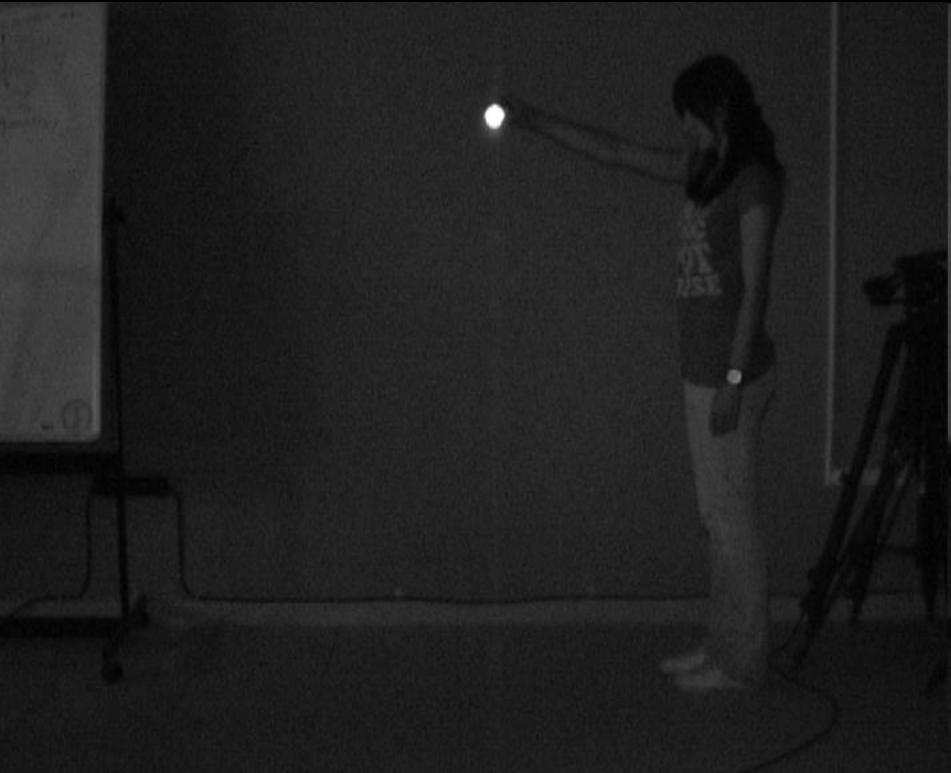
The Command History window shows the following commands:

```
C:\>
>> doc graythresh
>> [level EM] = graythresh(RGB);
>> doc im2bw
>> BWlevel = im2bw(RGB,level);
>> imshow(BWlevel)
```

level değişkeniyle çok fazla bilgi veya gürültü



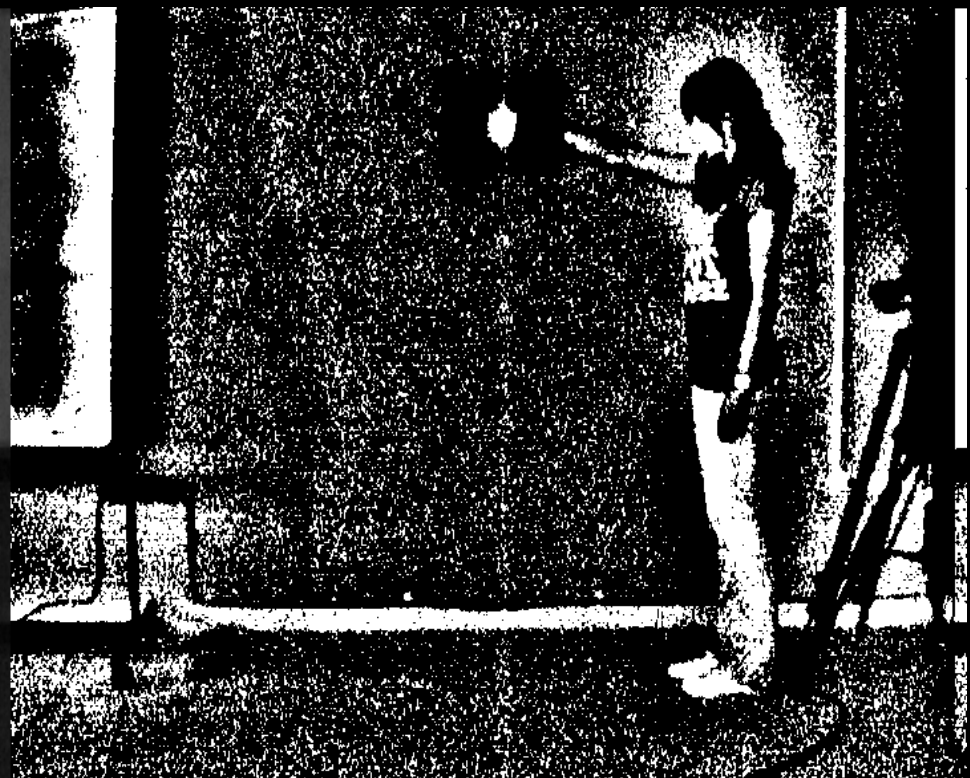
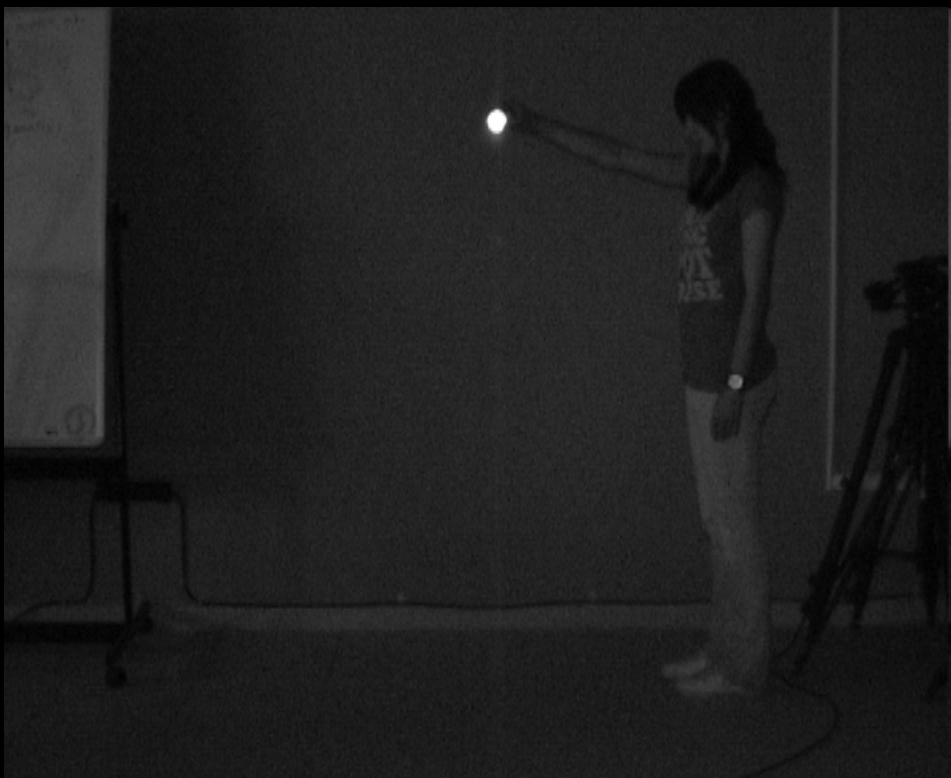
level değişkeniyle çok fazla bilgi veya gürültü





Adaptthresh fonksiyonunun kullanımı

Adaptive image threshold using local first-order statistics

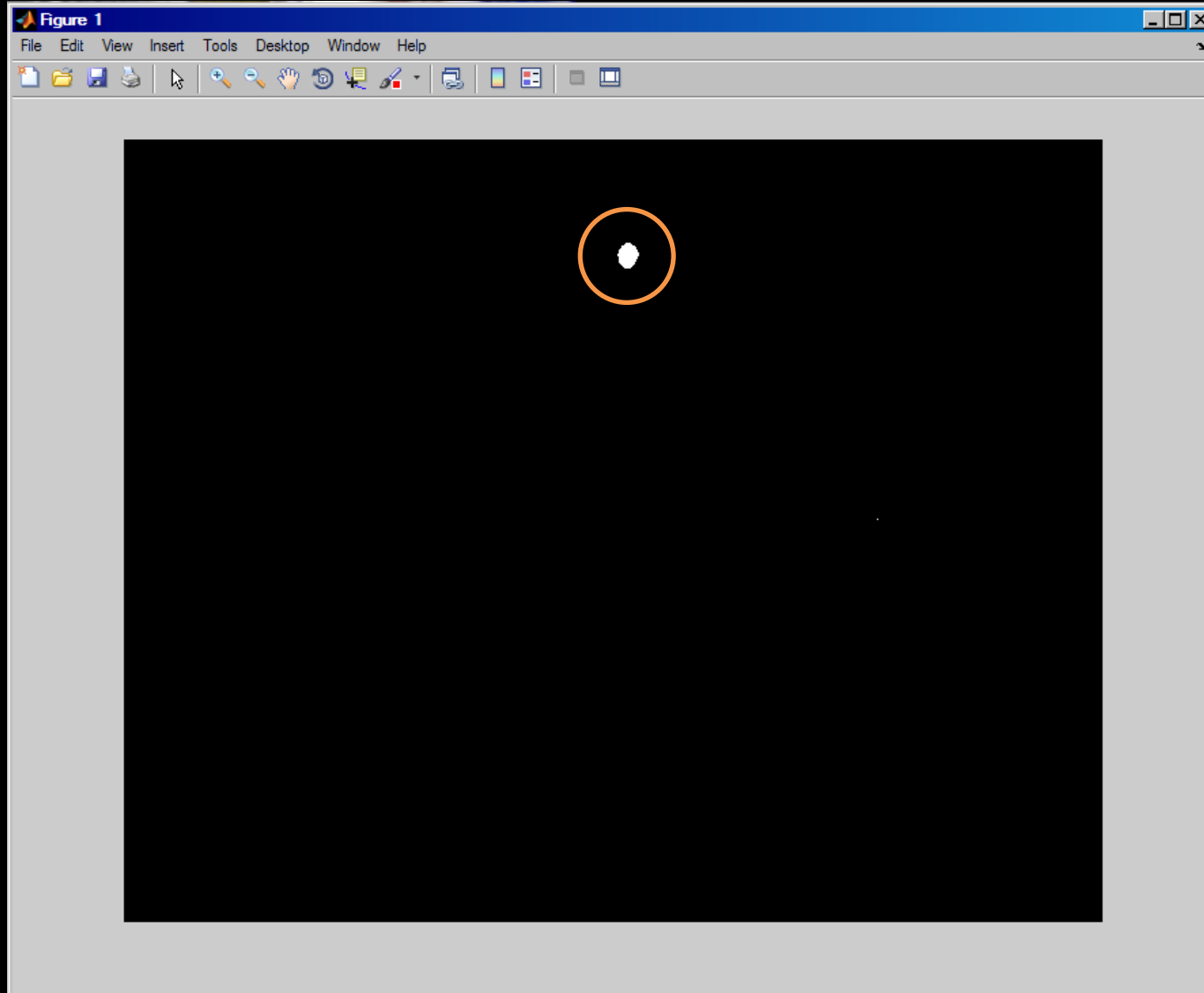






Try to find `a threshold` value  
which can be used to convert an  
intensity image to a binary image with  
`imbinarize`.

Yeni Eşik değışkeniyle kabul edilebilir bilgi ve az gürültü





## Yansıtıcı işaretin merkezinin bulunması : Etiketleme

The image shows the MATLAB R2011b interface. The Command Window on the right contains the command `doc bwlabel`, which is highlighted with an orange box. The Workspace on the left shows several variables: `BWem` (576x720 logical), `BWlevel` (576x720 logical), `EM` (0.4947), `L` (576x720 double), `RGB` (576x720x3 uint8), `ans` (576x1 double), and `level` (0.1412). The Command History at the bottom left shows the following commands: `max(bwlabel(BWem), [], 2)`, `L = bwlabel(BWem);`, `clc`, `L = bwlabel(BWem);`, `clc`, and `doc bwlabel`.

Start

OVR

## bwlabel fonksiyonu hakkında bilgi

Help

File Edit View Go Favorites Desktop Window Help

max

Contents Search Results

Image Processing Toolbox

Getting Started

User's Guide

fx Functions

Image Display and Exploration

GUI Tools

Spatial Transformation and Image Registration

Image Analysis and Statistics

Image Arithmetic

Image Enhancement and Restoration

Linear Filtering and Transforms

Morphological Operations

Intensity and Binary Images

Binary Images

fx applylut - Neighborhood operations on binary image

fx bwarea - Area of objects in binary image

fx bwareaopen - Morphologically open binary image

fx bwconncomp - Find connected components in binary image

fx bwconvhull - Generate convex hull image from binary image

fx bwdist - Distance transform of binary image

fx bwdistgeodesic - Geodesic distance transform of binary image

fx bweuler - Euler number of binary image

fx bwhitmiss - Binary hit-miss operation

fx bwlabel - Label connected components in 2-D binary image

fx bwlabeln - Label connected components in N-D binary image

fx bwmorph - Morphological operations on binary image

fx bwpack - Pack binary image

fx bwperim - Find perimeter of objects in binary image

fx bwselect - Select objects in binary image

fx bwulterode - Ultimate erosion

fx bwunpack - Unpack binary image

fx graydist - Gray-weighted distance transform of binary image

fx imtophat - Top-hat filtering

fx makelut - Create lookup table for use with applylut

Structuring Element Creation and Manipulation

ROI-Based, Neighborhood, and Block Processing

fx Image Processing Toolbox Functions Morphological Operations Binary Images bwlabel

### bwlabel

Label connected components in 2-D binary image

#### Syntax

```
L = bwlabel(BW, n)
[L, num] = bwlabel(BW, n)
```

#### Description

**L = bwlabel(BW, n)** returns a matrix **L**, of the same size as **BW**, containing labels for the connected objects in **BW**. The variable **n** can have a value of either 4 or 8, where 4 specifies 4-connected objects and 8 specifies 8-connected objects. If the argument is omitted, it defaults to 8.

The elements of **L** are integer values greater than or equal to 0. The pixels labeled 0 are the background. The pixels labeled 1 make up one object; the pixels labeled 2 make up a second object; and so on.

**[L, num] = bwlabel(BW, n)** returns in **num** the number of connected objects found in **BW**.

#### Tips

**Using bwlabel, bwlabeln, bwconncomp, and regionprops**

The functions **bwlabel**, **bwlabeln**, and **bwconncomp** all compute connected components for binary images. **bwconncomp** replaces the use of **bwlabel** and **bwlabeln**. It uses significantly less memory and is sometimes faster than the other functions.

	Input Dimension	Output Form	Memory Use	Connectivity
<b>bwlabel</b>	2-D	Double-precision label matrix	High	4 or 8
<b>bwlabeln</b>	N-D	Double-precision label matrix	High	Any
<b>bwconncomp</b>	N-D	CC struct	Low	Any

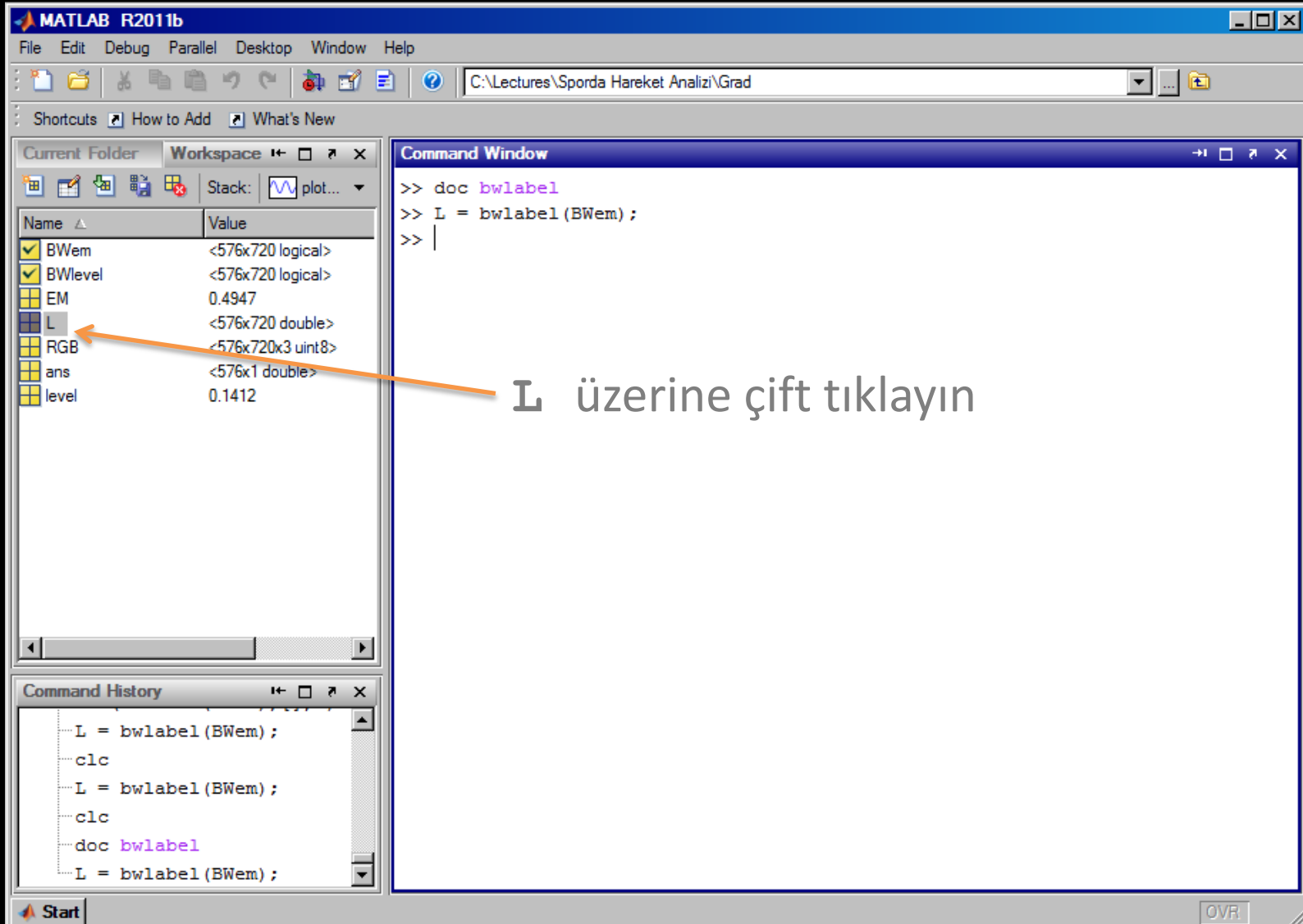
To extract features from a binary image using **regionprops** with default connectivity, just pass **BW** directly into **regionprops**, i.e., **regionprops(BW)**.

To compute a label matrix having a more memory-efficient data type (e.g., **uint8** versus **double**), use the **labelmatrix** function on the output of **bwconncomp**. For more information, see the reference page for each function.

#### Using find with bwlabel

You can use the MATLAB **find** function in conjunction with **bwlabel** to return vectors of indices for the pixels that make

## bwlabel fonksiyonunun kullanımı



Current Folder    Workspace    Stack: plot...

Name	Value
BWem	<576x720 logical>
BWlevel	<576x720 logical>
EM	0.4947
L	<576x720 double>
RGB	<576x720x3 uint8>
ans	<576x1 double>
level	0.1412

Command Window

```
>> doc bwlabel  
>> L = bwlabel(BWem);  
>>
```

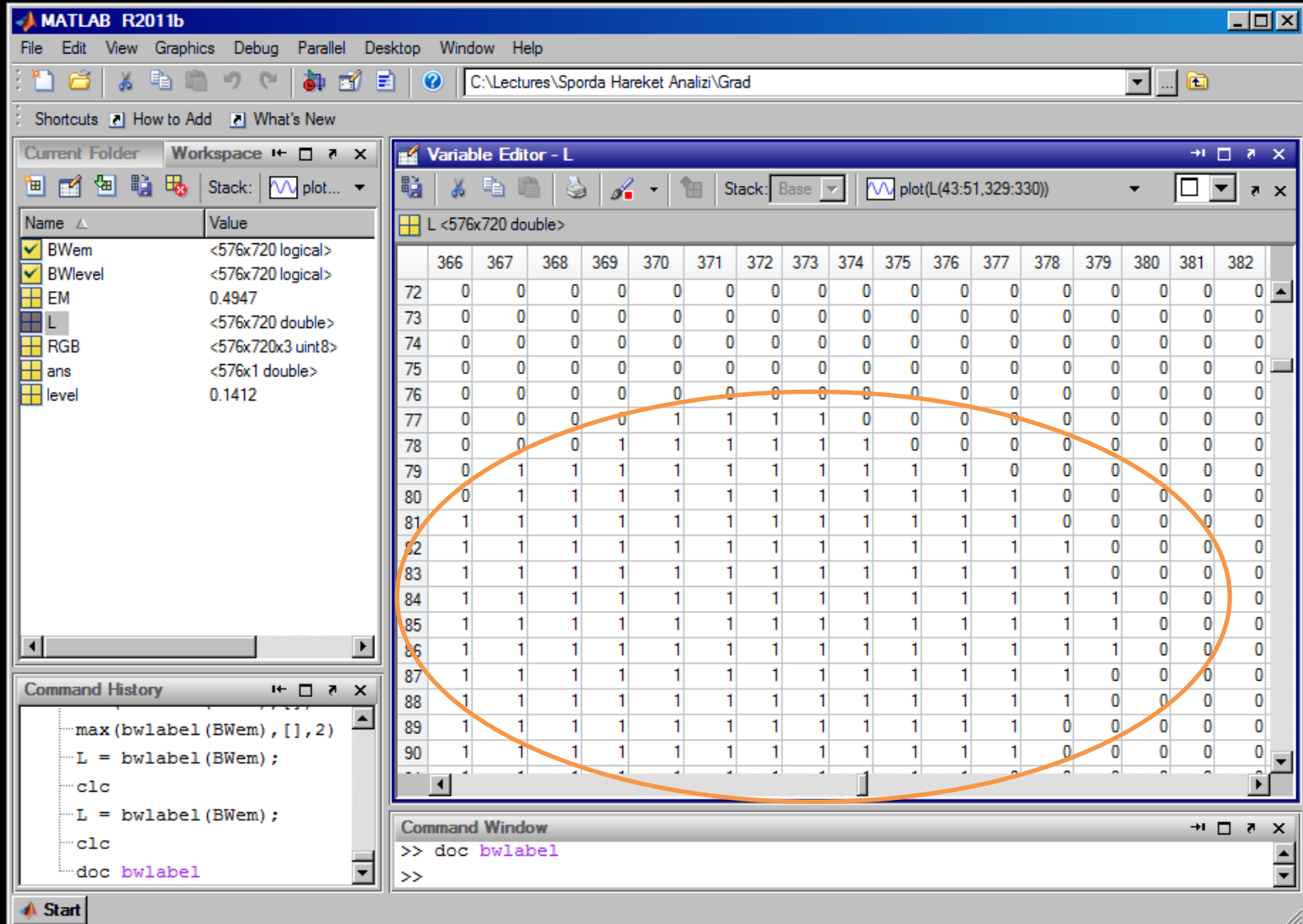
Command History

```
L = bwlabel(BWem);  
clc  
L = bwlabel(BWem);  
clc  
doc bwlabel  
L = bwlabel(BWem);
```

L üzerine çift tıklayın



## bwlabel fonksiyonunun çıktısı





## regionprops fonksiyonu hakkında bilgi

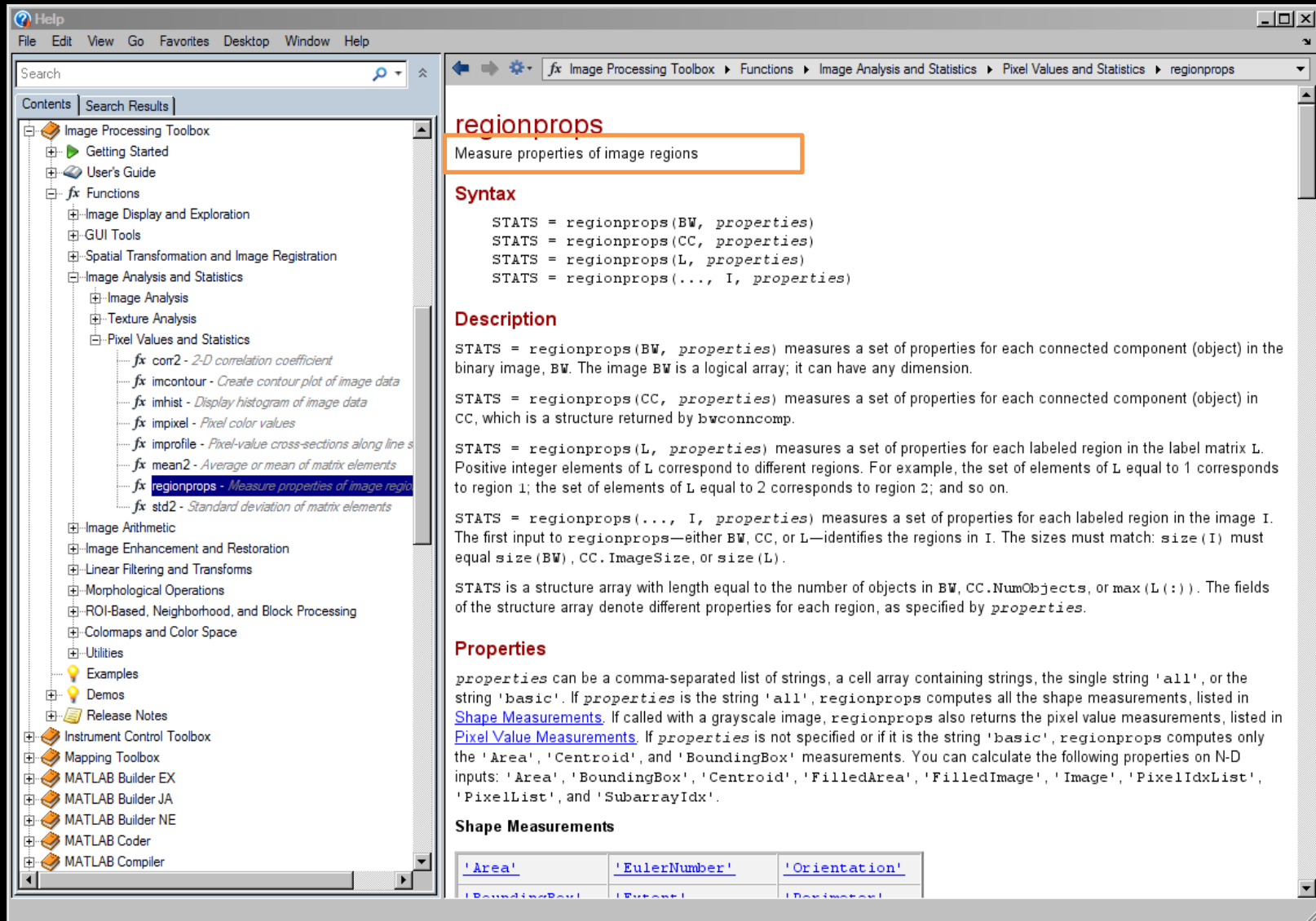
The image shows the MATLAB R2011b interface. The workspace window displays the following variables:

Name	Value
BWem	<576x720 logical>
BWlevel	<576x720 logical>
EM	0.4947
L	<576x720 double>
RGB	<576x720x3 uint8>
level	0.1412

The Command Window shows the command `doc regionprops` entered. The Command History window shows the following commands:

```
clc  
doc bwlabel  
L = bwlabel(BWem);  
cg = regionprops(L, 'cen'  
clc  
doc regionprops
```

## regionprops fonksiyonu hakkında bilgi



The screenshot shows the MATLAB Help window for the `regionprops` function. The left pane shows the 'Contents' tree with 'regionprops' selected under 'Pixel Values and Statistics'. The right pane displays the function's description, syntax, and properties.

### regionprops

Measure properties of image regions

#### Syntax

```
STATS = regionprops(BW, properties)
STATS = regionprops(CC, properties)
STATS = regionprops(L, properties)
STATS = regionprops(..., I, properties)
```

#### Description

`STATS = regionprops(BW, properties)` measures a set of properties for each connected component (object) in the binary image `BW`. The image `BW` is a logical array; it can have any dimension.

`STATS = regionprops(CC, properties)` measures a set of properties for each connected component (object) in `CC`, which is a structure returned by `bwconncomp`.

`STATS = regionprops(L, properties)` measures a set of properties for each labeled region in the label matrix `L`. Positive integer elements of `L` correspond to different regions. For example, the set of elements of `L` equal to 1 corresponds to region 1; the set of elements of `L` equal to 2 corresponds to region 2; and so on.

`STATS = regionprops(..., I, properties)` measures a set of properties for each labeled region in the image `I`. The first input to `regionprops`—either `BW`, `CC`, or `L`—identifies the regions in `I`. The sizes must match: `size(I)` must equal `size(BW)`, `CC.ImageSize`, or `size(L)`.

`STATS` is a structure array with length equal to the number of objects in `BW`, `CC.NumObjects`, or `max(L(:))`. The fields of the structure array denote different properties for each region, as specified by `properties`.

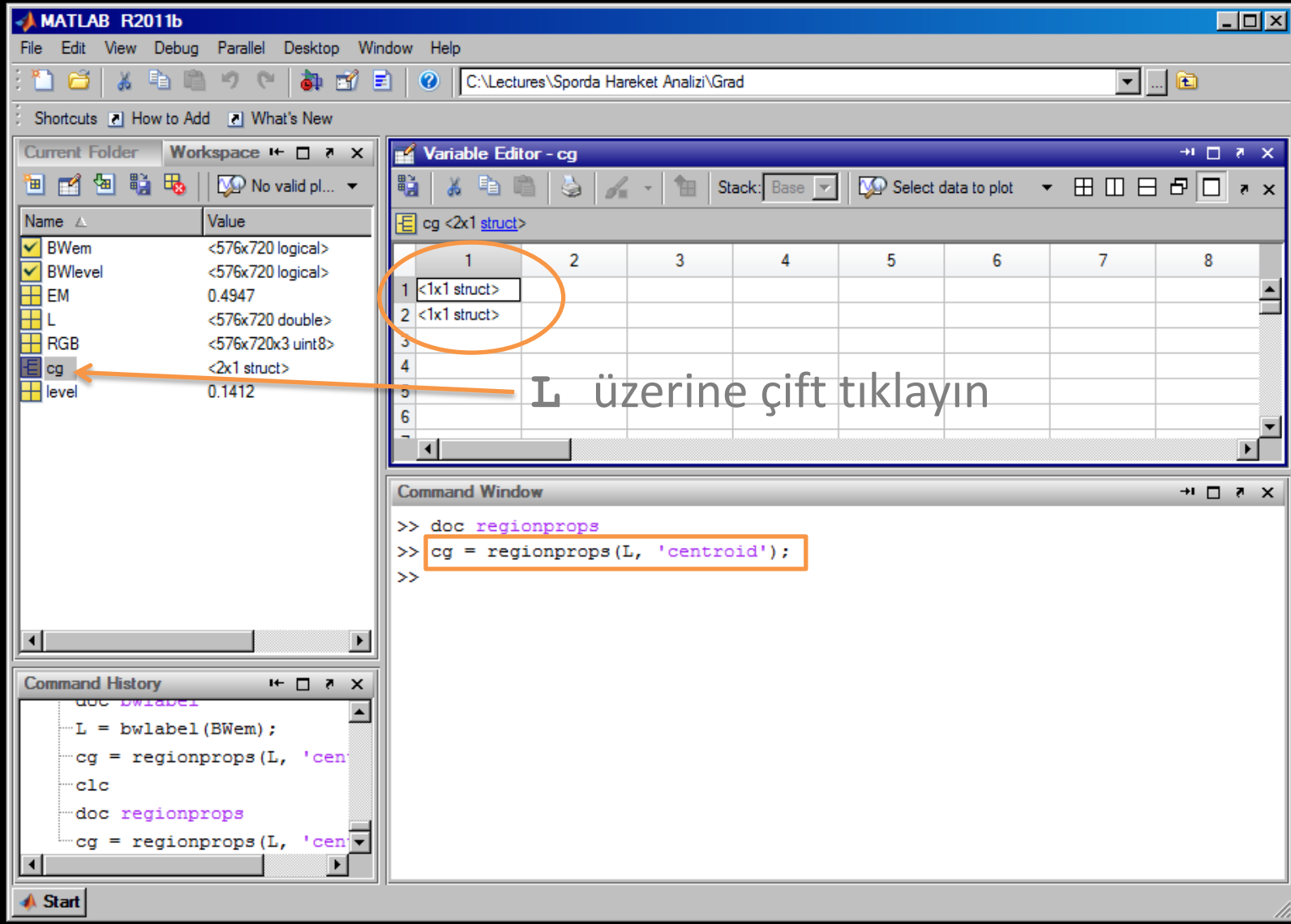
#### Properties

`properties` can be a comma-separated list of strings, a cell array containing strings, the single string 'all', or the string 'basic'. If `properties` is the string 'all', `regionprops` computes all the shape measurements, listed in [Shape Measurements](#). If called with a grayscale image, `regionprops` also returns the pixel value measurements, listed in [Pixel Value Measurements](#). If `properties` is not specified or if it is the string 'basic', `regionprops` computes only the 'Area', 'Centroid', and 'BoundingBox' measurements. You can calculate the following properties on N-D inputs: 'Area', 'BoundingBox', 'Centroid', 'FilledArea', 'FilledImage', 'Image', 'PixelIdxList', 'PixelList', and 'SubarrayIdx'.

#### Shape Measurements

'Area'	'EulerNumber'	'Orientation'
'BoundingBox'	'EulerNumber'	'Perimeter'

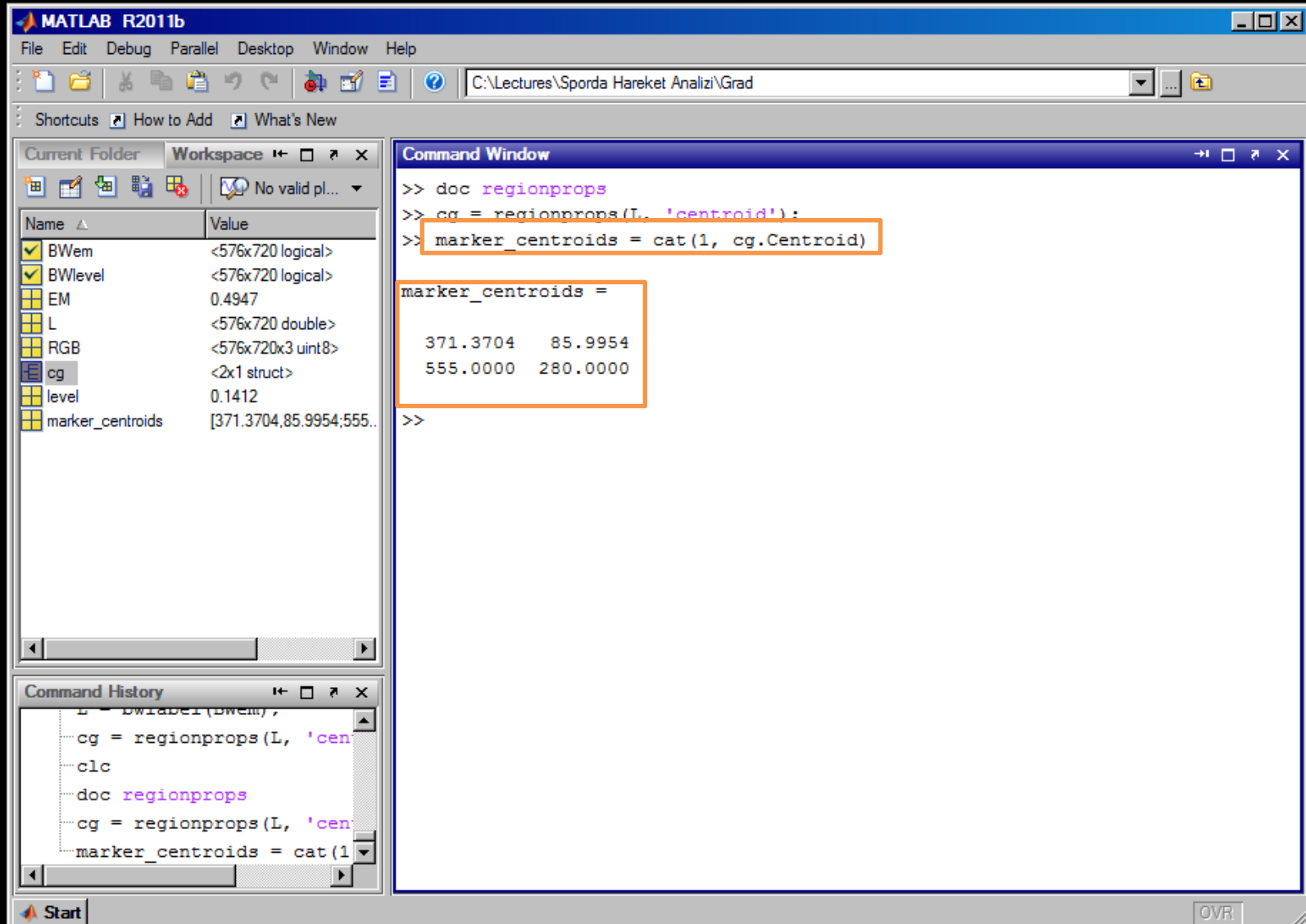
## regionprops fonksiyonunun kullanımı



The image shows the MATLAB R2011b interface with the following components:

- Workspace:** Lists variables: BWem (<576x720 logical>), BWlevel (<576x720 logical>), EM (0.4947), L (<576x720 double>), RGB (<576x720x3 uint8>), cg (<2x1 struct>), and level (0.1412). The 'cg' variable is highlighted with an orange arrow.
- Variable Editor - cg:** Shows the structure of the 'cg' variable. It is a 2x1 struct. The first element is '1' (1x1 struct) and the second element is '2' (1x1 struct). An orange circle highlights the first element, and an orange arrow points to it with the text "L üzerine çift tıklayın".
- Command Window:** Shows the command: `doc regionprops` and `cg = regionprops(L, 'centroid');`. The second command is highlighted with an orange box.
- Command History:** Shows the previous commands: `doc bwlabel`, `L = bwlabel(BWem);`, `cg = regionprops(L, 'cen`, `clc`, `doc regionprops`, and `cg = regionprops(L, 'cen`.

## Yansıtıcı işaretlerin koordinatlarının bulunması



The image shows the MATLAB R2011b interface. The workspace window displays the following variables:

Name	Value
BWem	<576x720 logical>
BWlevel	<576x720 logical>
EM	0.4947
L	<576x720 double>
RGB	<576x720x3 uint8>
cg	<2x1 struct>
level	0.1412
marker_centroids	[371.3704,85.9954;555.0000,280.0000]

The Command Window shows the following code and output:

```
>> doc regionprops
>> cg = regionprops(L, 'centroid');
>> marker_centroids = cat(1, cg.Centroid)

marker_centroids =

    371.3704    85.9954
    555.0000   280.0000

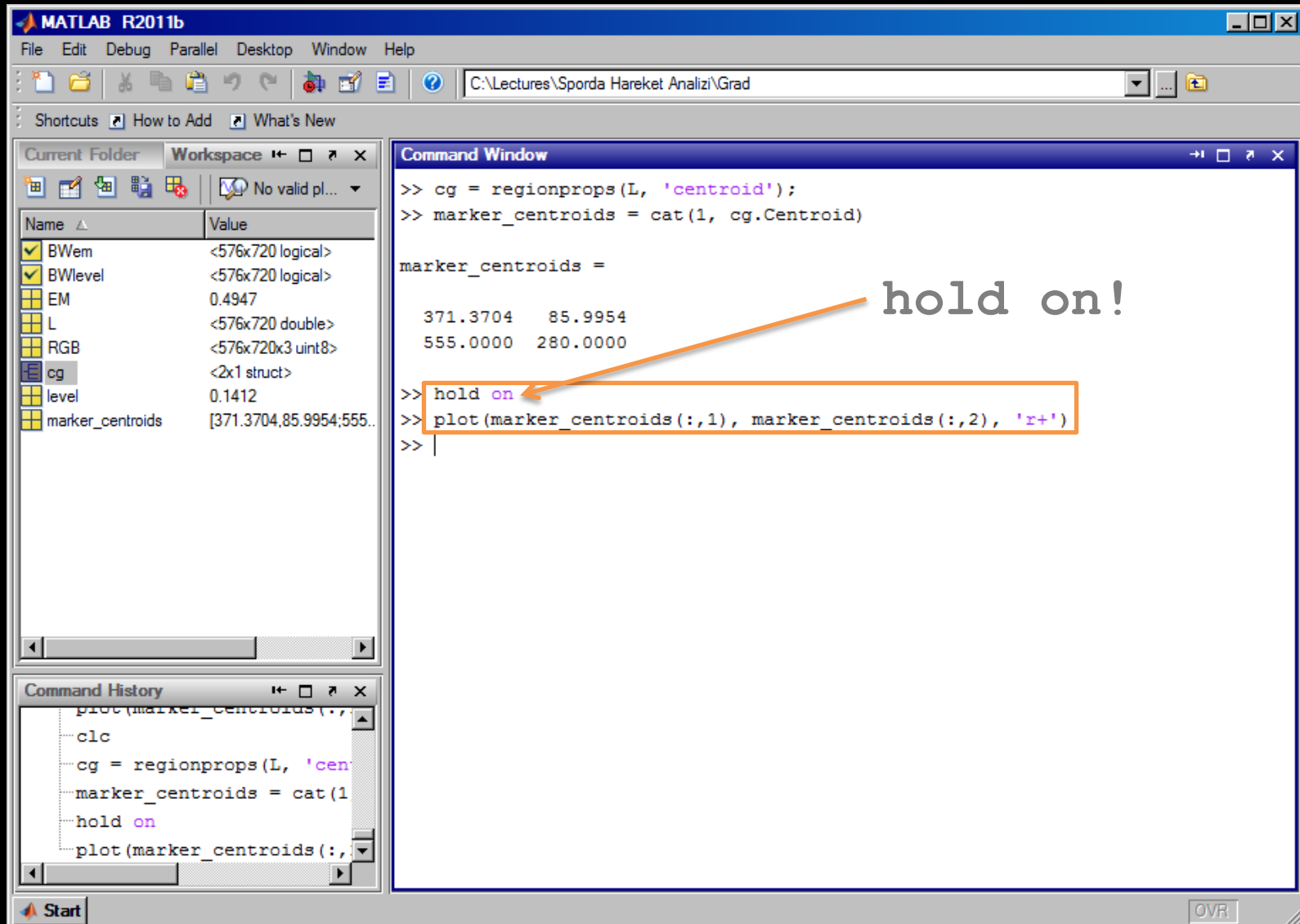
>>
```

The Command History window shows the following commands:

```
L = bwlabel(BWem);
cg = regionprops(L, 'centroid');
clc
doc regionprops
cg = regionprops(L, 'centroid');
marker_centroids = cat(1, cg.Centroid);
```



## Yansıtıcı işaret koordinatlarının grafiklenmesi



**Current Folder**

Name	Value
BWem	<576x720 logical>
BWlevel	<576x720 logical>
EM	0.4947
L	<576x720 double>
RGB	<576x720x3 uint8>
cg	<2x1 struct>
level	0.1412
marker_centroids	[371.3704,85.9954;555.0000,280.0000]

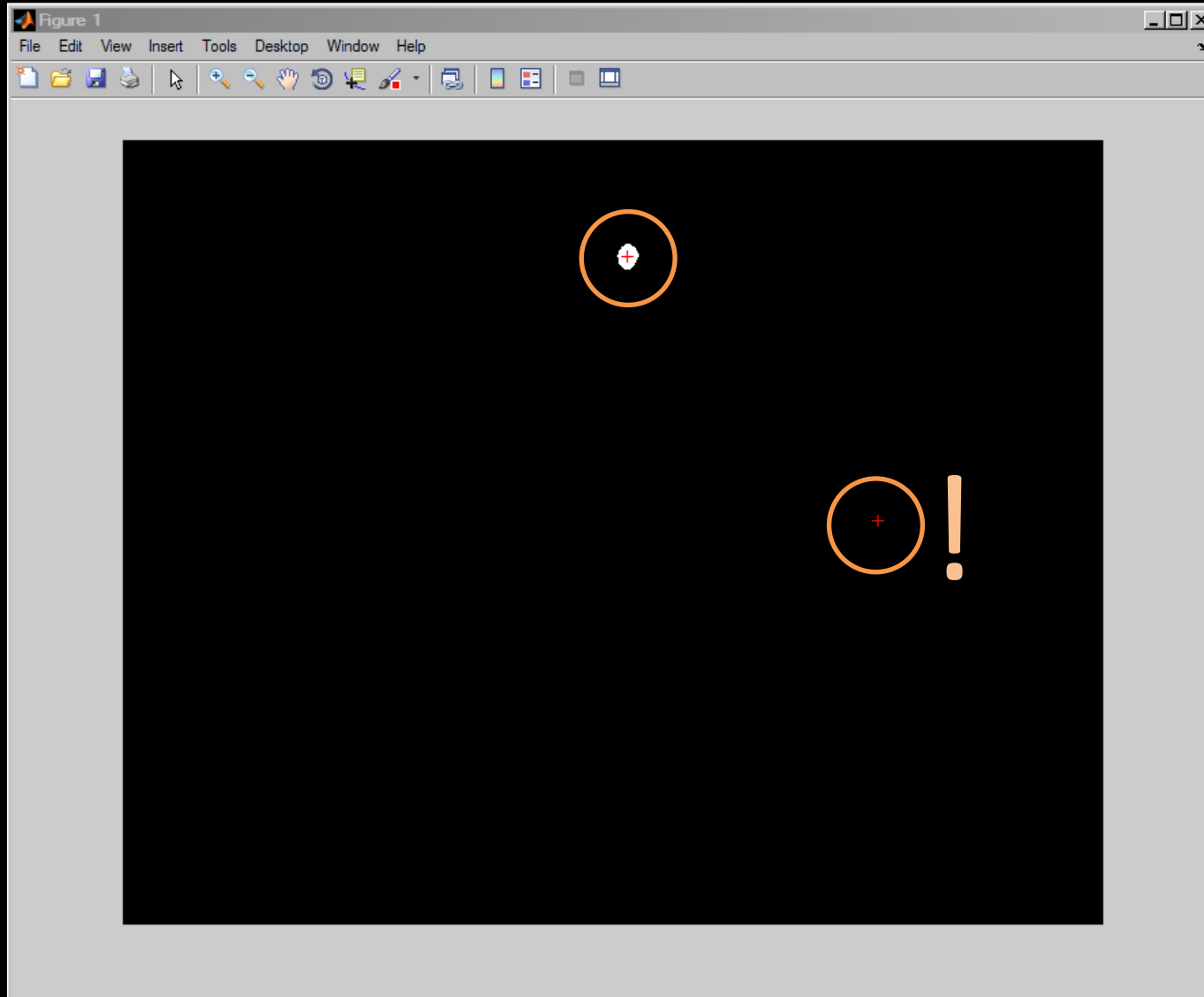
**Command Window**

```
>> cg = regionprops(L, 'centroid');  
>> marker_centroids = cat(1, cg.Centroid)  
  
marker_centroids =  
  
    371.3704    85.9954  
    555.0000   280.0000  
  
>> hold on  
>> plot(marker_centroids(:,1), marker_centroids(:,2), 'r+')  
>>
```

**Command History**

```
plot(marker_centroids(:,1), marker_centroids(:,2), 'r+')  
clc  
cg = regionprops(L, 'centroid')  
marker_centroids = cat(1, cg.Centroid)  
hold on  
plot(marker_centroids(:,1), marker_centroids(:,2), 'r+')
```

## Yansıtıcı işaret koordinatlarının grafiklenmesi





## medfilt2 fonksiyonunun kullanımı

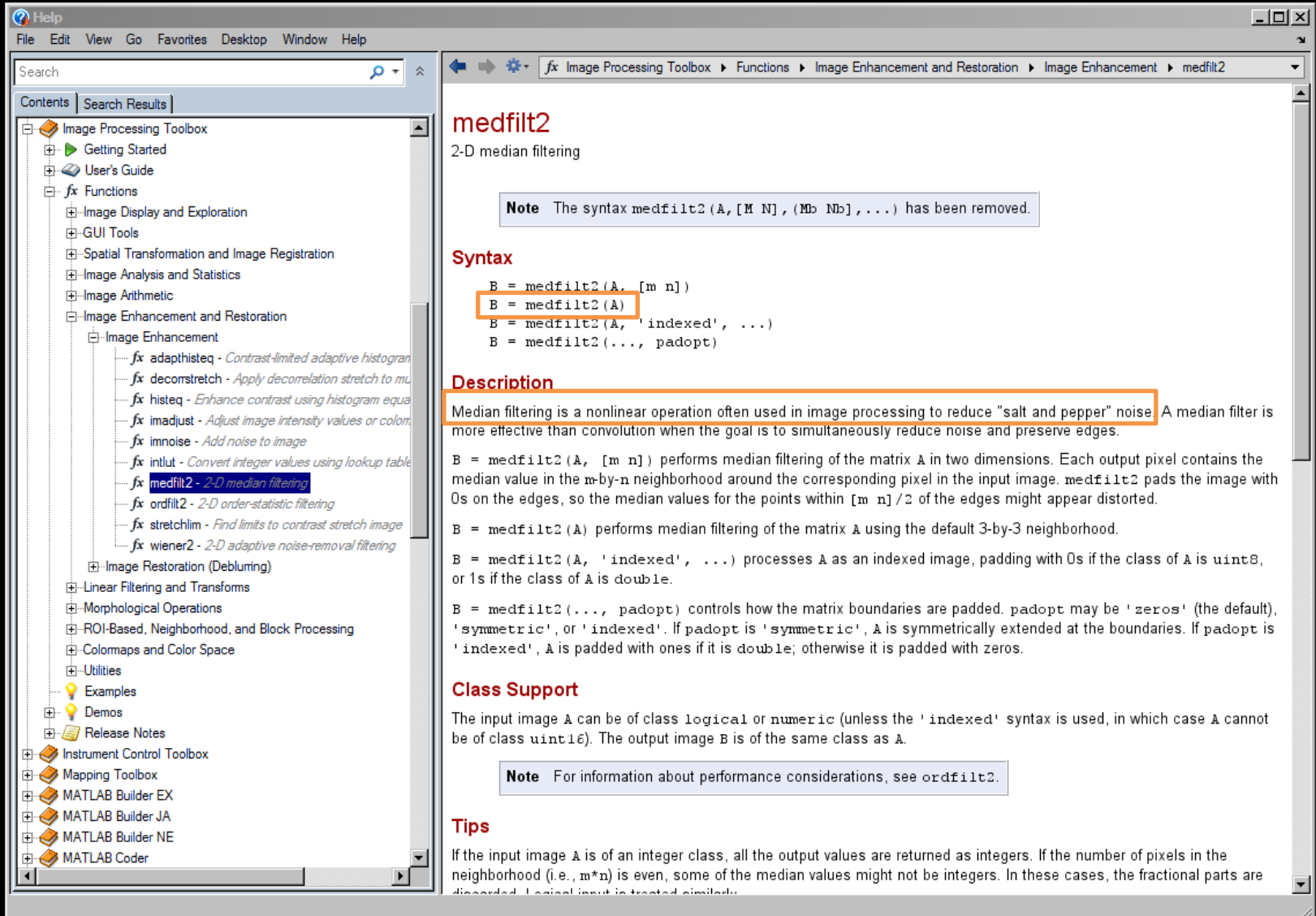
The image shows the MATLAB R2011b interface. The Command Window displays the command `doc medfilt2`, which is highlighted with an orange box. The Workspace window shows a table of variables:

Name	Value
BW	<576x720 logical>
BWem	<576x720 logical>
BWlevel	<576x720 logical>
EM	0.4947
L	<576x720 double>
RGB	<576x720x3 uint8>
cg	<1x1 struct>
level	0.1412
marker_centroids	[371.3907,86.0279]

The Command History window shows the following commands:

```
marker_centroids = cat(1, ...  
imshow(RGB)  
hold on  
plot(marker_centroids(:, ...  
clc  
doc medfilt2
```

## medfilt2 fonksiyonunun kullanımı



**medfilt2**  
2-D median filtering

**Note** The syntax `medfilt2(A, [M N], (Mb Nb), ...)` has been removed.

**Syntax**

```
B = medfilt2(A, [m n])  
B = medfilt2(A)  
B = medfilt2(A, 'indexed', ...)  
B = medfilt2(..., padopt)
```

**Description**

Median filtering is a nonlinear operation often used in image processing to reduce "salt and pepper" noise. A median filter is more effective than convolution when the goal is to simultaneously reduce noise and preserve edges.

`B = medfilt2(A, [m n])` performs median filtering of the matrix `A` in two dimensions. Each output pixel contains the median value in the `m`-by-`n` neighborhood around the corresponding pixel in the input image. `medfilt2` pads the image with 0s on the edges, so the median values for the points within `[m n]/2` of the edges might appear distorted.

`B = medfilt2(A)` performs median filtering of the matrix `A` using the default 3-by-3 neighborhood.

`B = medfilt2(A, 'indexed', ...)` processes `A` as an indexed image, padding with 0s if the class of `A` is `uint8`, or 1s if the class of `A` is `double`.

`B = medfilt2(..., padopt)` controls how the matrix boundaries are padded. `padopt` may be `'zeros'` (the default), `'symmetric'`, or `'indexed'`. If `padopt` is `'symmetric'`, `A` is symmetrically extended at the boundaries. If `padopt` is `'indexed'`, `A` is padded with ones if it is `double`; otherwise it is padded with zeros.

**Class Support**

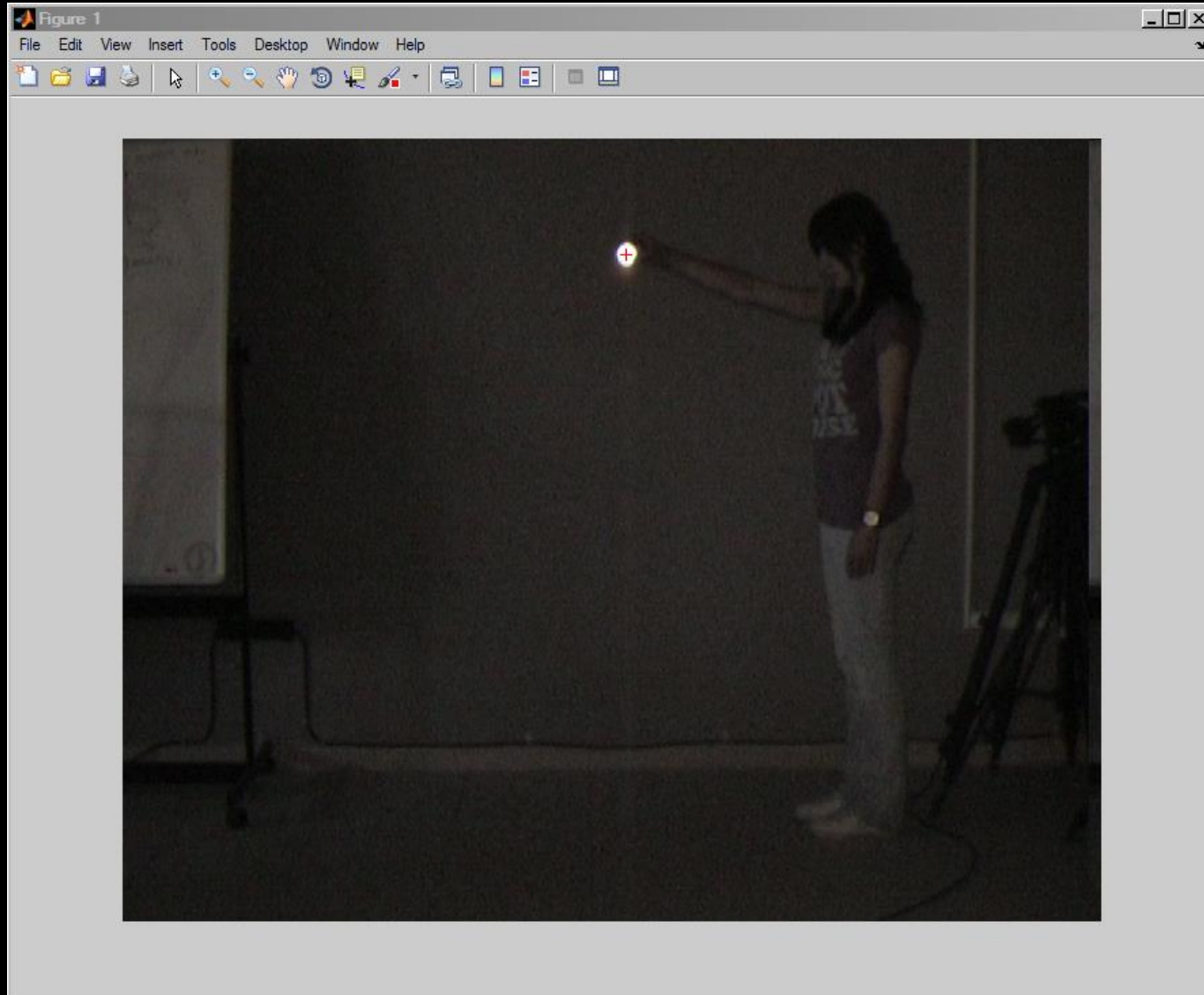
The input image `A` can be of class `logical` or `numeric` (unless the `'indexed'` syntax is used, in which case `A` cannot be of class `uint16`). The output image `B` is of the same class as `A`.

**Note** For information about performance considerations, see `ordfilt2`.

**Tips**

If the input image `A` is of an integer class, all the output values are returned as integers. If the number of pixels in the neighborhood (i.e., `m*n`) is even, some of the median values might not be integers. In these cases, the fractional parts are discarded. Logical input is treated similarly.

## Sonuç





## Öğrendiğimiz MATLAB fonksiyonları:

`imread`

`imshow`

`graythresh`

`im2bw`

`medfilt2`

`bwlabel`

`regionprops`

`plot`

`hold on`



## Ev Ödevi

**Teslim Tarihi : 22 Ekim 2019 Saat 10:00**

- Threshold (eşik) değerini belirlemek için gri değerlerin histogramını çiziniz.
- Resimlerdeki topun merkezini bulan ve resim üzerinde işaretleyen bir program yazınız.
- MATLAB R2016a dan itibaren desteklenen `otsuthresh`, `adaptthresh` fonksiyonlarını da programınızda deneyin.

```
T = otsuthresh(counts)
```

```
[T,EM] = otsuthresh(counts)
```

```
T = adaptthresh(I, 0.4)
```



## Araştırma Makalesi Okuma

NOBUYUKI OTSU, A Threshold Selection Method from Gray-Level Histograms. IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, VOL. SMC-9, NO. 1, JANUARY 1979

