BCA 607 Hareket Analizi Sistemleri

Matlab ile Görüntü İşleme 2

De Motu Animalium G.Borelli (1680)

SERDAR ARITAN

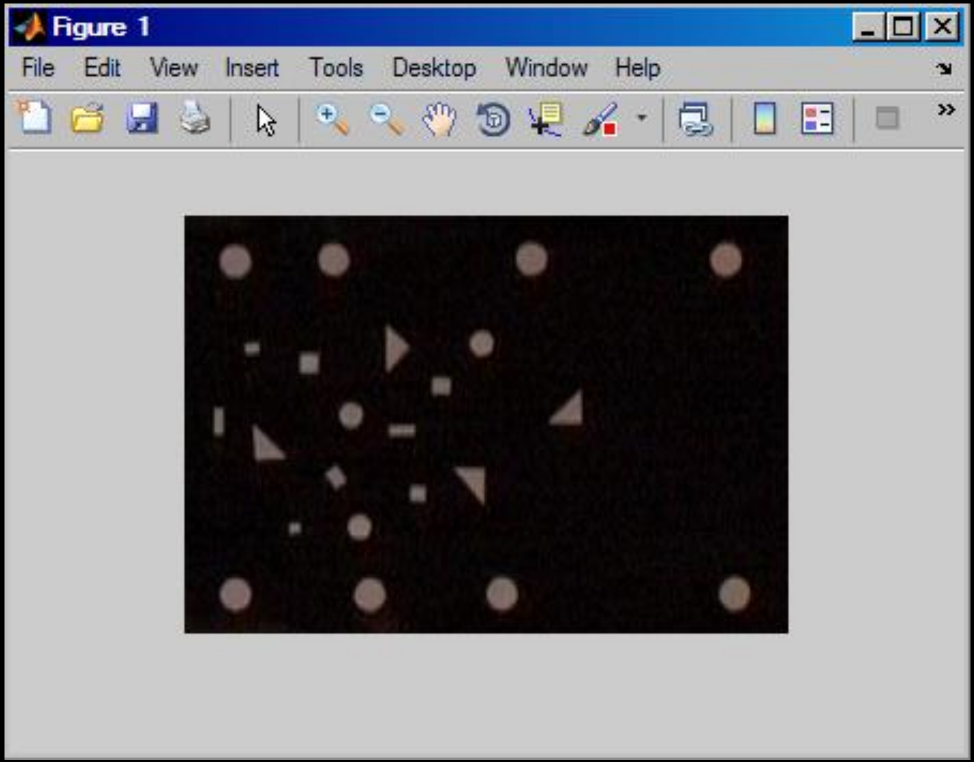
serdar.aritan@hacettepe.edu.tr

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

Yansıtıcı işaret yakalama

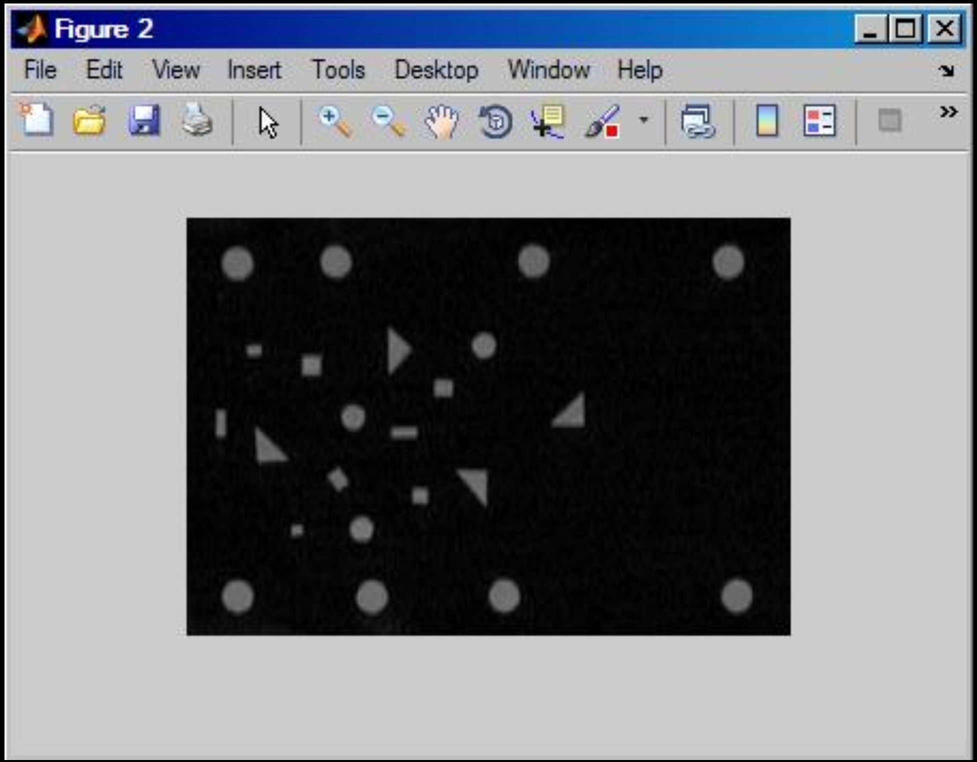
Yansıtıcı işaret yakalama

```
RGB = imread('tnesneler.jpg');
imshow(RGB);
```

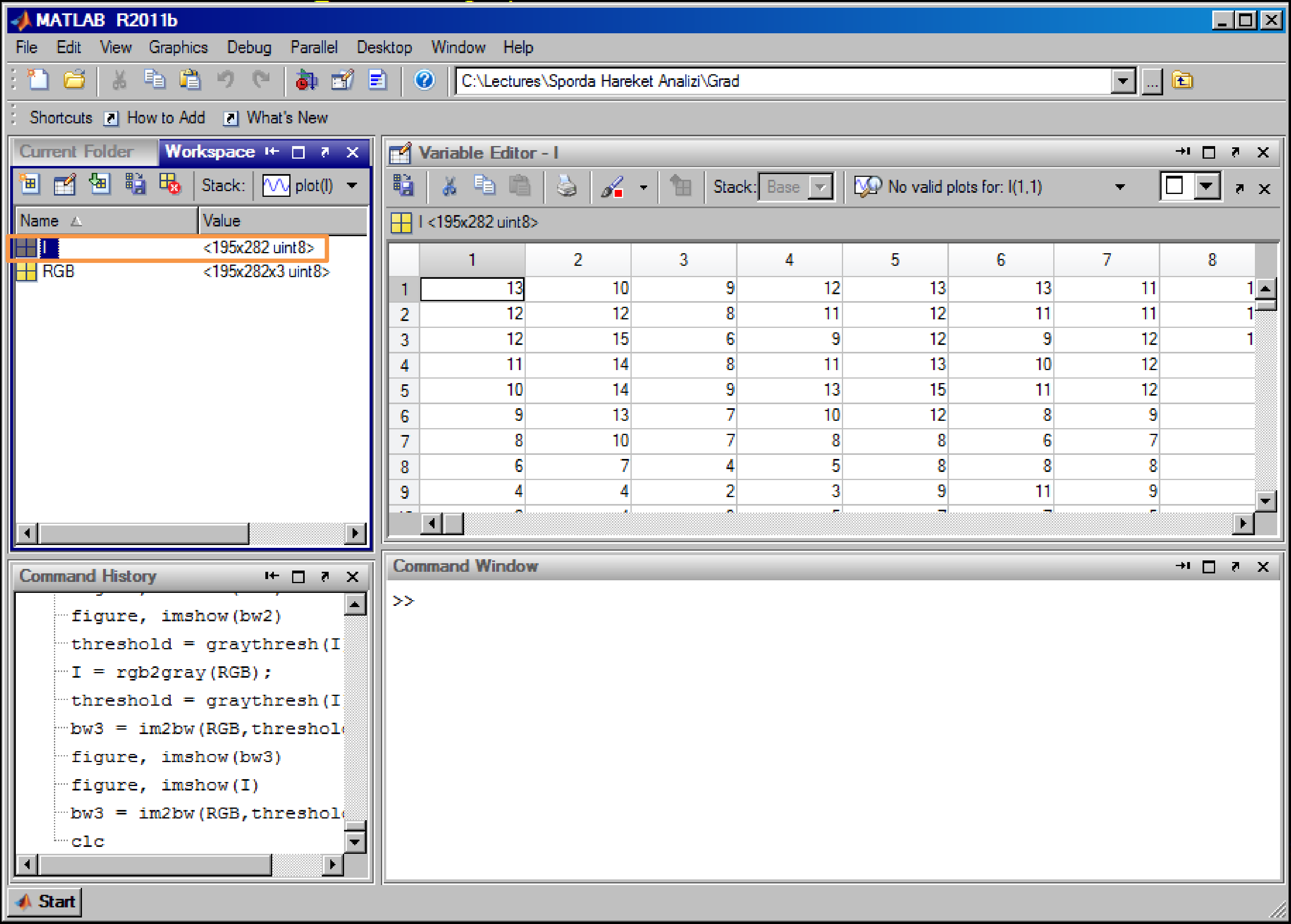


Yansıtıcı işaret yakalama

```
I = rgb2gray(RGB);
figure,imshow(I)
```



Yansıtıcı işaret yakalama


MATLAB R2011b
File Edit View Graphics Debug Parallel Desktop Window Help
Shortcuts How to Add What's New
Current Folder Workspace
Stack: plot(I)
Variable Editor - I
Stack: Base
No valid plots for: I(1,1)
Command History
Command Window
Start

C:\Lectures\Sporda Hareket Analizi\Grad

Name	Value
I	<195x282 uint8>
RGB	<195x282x3 uint8>

I <195x282 uint8>

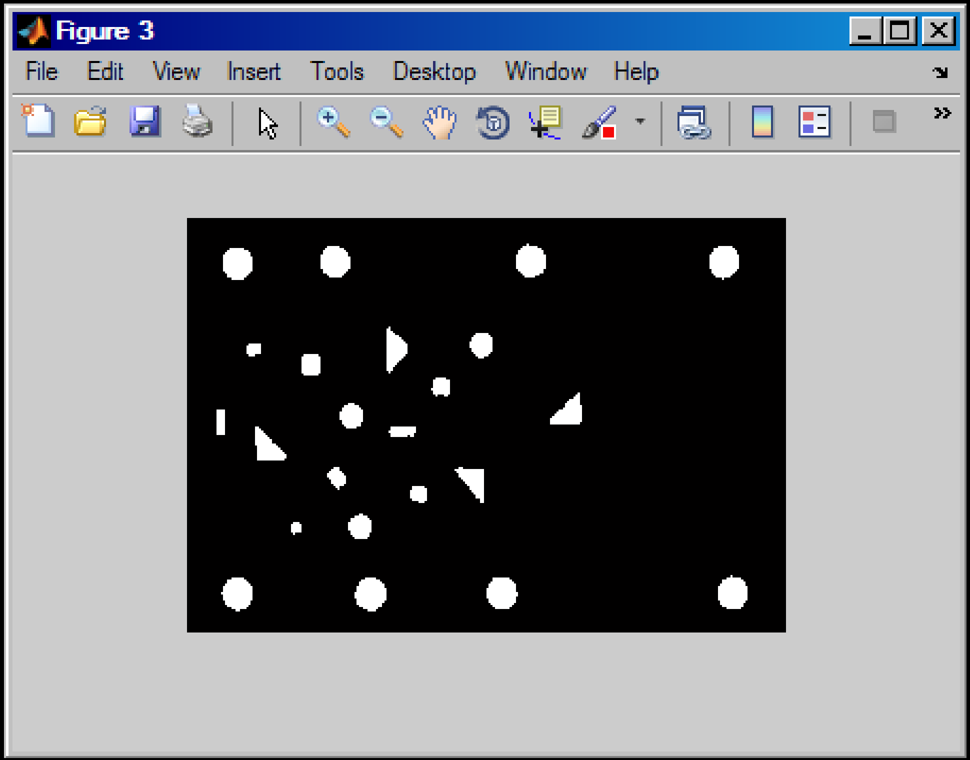
	1	2	3	4	5	6	7	8
1	13	10	9	12	13	13	11	1
2	12	12	8	11	12	11	11	1
3	12	15	6	9	12	9	12	1
4	11	14	8	11	13	10	12	
5	10	14	9	13	15	11	12	
6	9	13	7	10	12	8	9	
7	8	10	7	8	8	6	7	
8	6	7	4	5	8	8	8	
9	4	4	2	3	9	11	9	

```
figure, imshow(bw2)
threshold = graythresh(I
I = rgb2gray(RGB);
threshold = graythresh(I
bw3 = im2bw(RGB,threshol
figure, imshow(bw3)
figure, imshow(I)
bw3 = im2bw(RGB,threshol
clc
```

```
>>
```

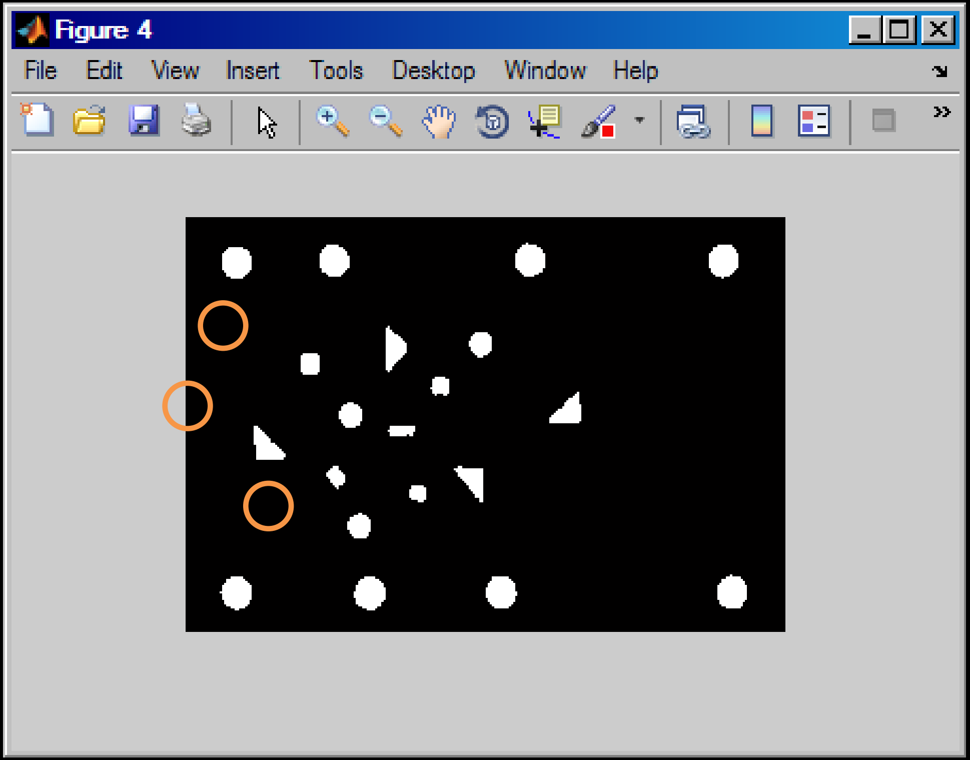
Yansıtıcı işaret yakalama

```
threshold = graythresh(I);
bw = im2bw(I,threshold);
figure,imshow(bw)
```

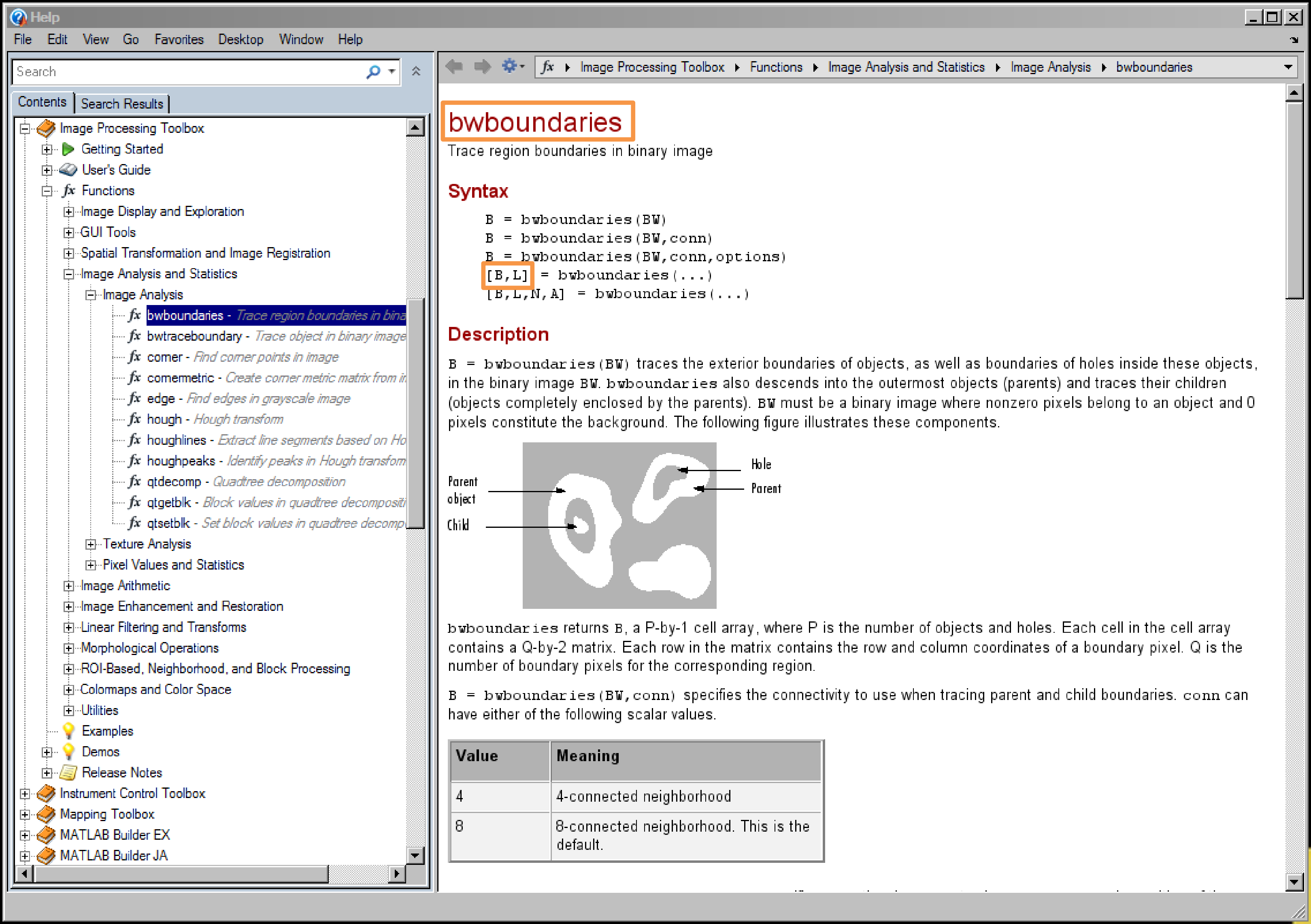


Yansıtıcı işaret yakalama

```
% Alanı 50 pikselden az olan
% nesneleri resimden sil
bw = bwareaopen(bw,50);
figure, imshow(bw)
```



Yansıtıcı işaret yakalama


Help
File Edit View Go Favorites Desktop Window Help
Search
Contents Search Results
Image Processing Toolbox
Getting Started
User's Guide
Functions
Image Display and Exploration
GUI Tools
Spatial Transformation and Image Registration
Image Analysis and Statistics
Image Analysis
bwboundaries - Trace region boundaries in bina
bwtraceboundary - Trace object in binary image
corner - Find corner points in image
cornermetric - Create corner metric matrix from i
edge - Find edges in grayscale image
hough - Hough transform
houghlines - Extract line segments based on Ho
houghpeaks - Identify peaks in Hough transform
qtdecomp - Quadtree decomposition
qtgetblk - Block values in quadtree decompositi
qtsetblk - Set block values in quadtree decomp
Texture Analysis
Pixel Values and Statistics
Image Arithmetic
Image Enhancement and Restoration
Linear Filtering and Transforms
Morphological Operations
ROI-Based, Neighborhood, and Block Processing
Colormaps and Color Space
Utilities
Examples
Demos
Release Notes
Instrument Control Toolbox
Mapping Toolbox
MATLAB Builder EX
MATLAB Builder JA
Image Processing Toolbox ▸ Functions ▸ Image Analysis and Statistics ▸ Image Analysis ▸ bwboundaries
Parent object
Child
Hole
Parent

bwboundaries

Trace region boundaries in binary image

Syntax

```
B = bwboundaries(BW)
B = bwboundaries(BW,conn)
B = bwboundaries(BW,conn,options)
[B,L] = bwboundaries(...)
[B,L,N,A] = bwboundaries(...)
```

Description

`B = bwboundaries(BW)` traces the exterior boundaries of objects, as well as boundaries of holes inside these objects, in the binary image `BW`. `bwboundaries` also descends into the outermost objects (parents) and traces their children (objects completely enclosed by the parents). `BW` must be a binary image where nonzero pixels belong to an object and 0 pixels constitute the background. The following figure illustrates these components.

`bwboundaries` returns `B`, a P-by-1 cell array, where P is the number of objects and holes. Each cell in the cell array contains a Q-by-2 matrix. Each row in the matrix contains the row and column coordinates of a boundary pixel. Q is the number of boundary pixels for the corresponding region.

`B = bwboundaries(BW,conn)` specifies the connectivity to use when tracing parent and child boundaries. `conn` can have either of the following scalar values.

Value	Meaning
4	4-connected neighborhood
8	8-connected neighborhood. This is the default.

Yansıtıcı işaret yakalama

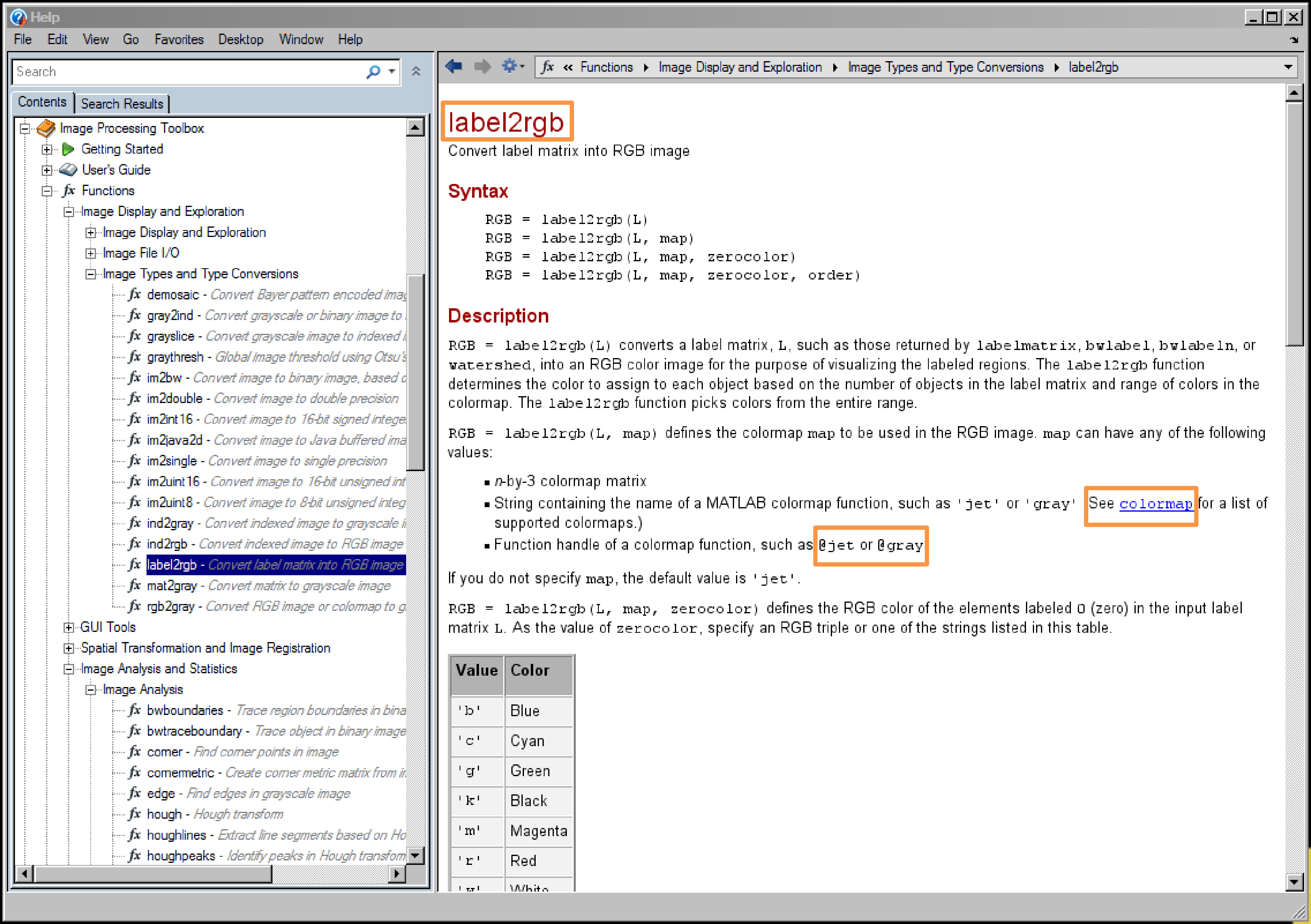

Help
File Edit View Go Favorites Desktop Window Help
Search
Contents
Search Results
Functions » Image Display and Exploration » Image Types and Type Conversions » label2rgb

Image Processing Toolbox
- Getting Started
- User's Guide
- Functions
  - Image Display and Exploration
    - Image Display and Exploration
    - Image File I/O
    - Image Types and Type Conversions
      - demosaic - *Convert Bayer pattern encoded imag*
      - gray2ind - *Convert grayscale or binary image to*
      - grayslice - *Convert grayscale image to indexed i*
      - graythresh - *Global image threshold using Otsu's*
      - im2bw - *Convert image to binary image, based o*
      - im2double - *Convert image to double precision*
      - im2int16 - *Convert image to 16-bit signed intege*
      - im2java2d - *Convert image to Java buffered ima*
      - im2single - *Convert image to single precision*
      - im2uint16 - *Convert image to 16-bit unsigned int*
      - im2uint8 - *Convert image to 8-bit unsigned integ*
      - ind2gray - *Convert indexed image to grayscale i*
      - ind2rgb - *Convert indexed image to RGB image*
      - label2rgb - *Convert label matrix into RGB image*
      - mat2gray - *Convert matrix to grayscale image*
      - rgb2gray - *Convert RGB image or colormap to g*
  - GUI Tools
  - Spatial Transformation and Image Registration
  - Image Analysis and Statistics
    - Image Analysis
      - bwboundaries - *Trace region boundaries in bina*
      - bwtraceboundary - *Trace object in binary image*
      - corner - *Find corner points in image*
      - cornermetric - *Create corner metric matrix from i*
      - edge - *Find edges in grayscale image*
      - hough - *Hough transform*
      - houghlines - *Extract line segments based on Ho*
      - houghpeaks - *Identify peaks in Hough transform*

label2rgb

Convert label matrix into RGB image

Syntax

```
RGB = label2rgb(L)
RGB = label2rgb(L, map)
RGB = label2rgb(L, map, zerocolor)
RGB = label2rgb(L, map, zerocolor, order)
```

Description

`RGB = label2rgb(L)` converts a label matrix, `L`, such as those returned by `labelmatrix`, `bwlabel`, `bwlabeln`, or `watershed`, into an RGB color image for the purpose of visualizing the labeled regions. The `label2rgb` function determines the color to assign to each object based on the number of objects in the label matrix and range of colors in the colormap. The `label2rgb` function picks colors from the entire range.

`RGB = label2rgb(L, map)` defines the colormap `map` to be used in the RGB image. `map` can have any of the following values:

- *n*-by-3 colormap matrix
- String containing the name of a MATLAB colormap function, such as `'jet'` or `'gray'` See colormap for a list of supported colormaps.)
- Function handle of a colormap function, such as `@jet` or `@gray`

If you do not specify `map`, the default value is `'jet'`.

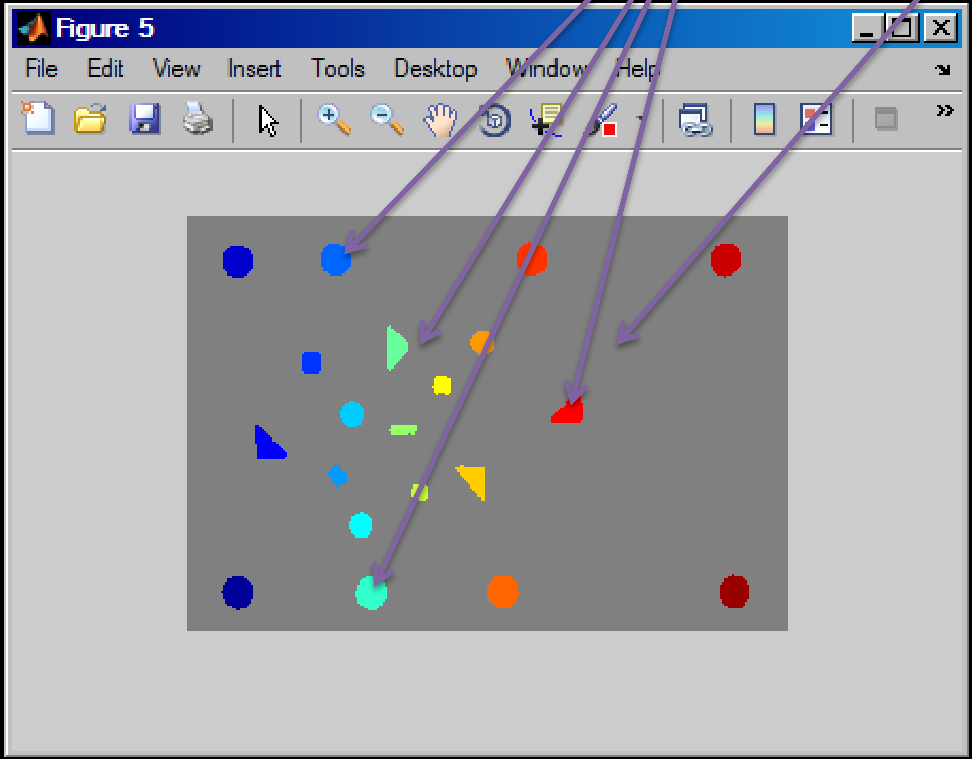
`RGB = label2rgb(L, map, zerocolor)` defines the RGB color of the elements labeled `0` (zero) in the input label matrix `L`. As the value of `zerocolor`, specify an RGB triple or one of the strings listed in this table.

Value	Color
'b'	Blue
'c'	Cyan
'g'	Green
'k'	Black
'm'	Magenta
'r'	Red
'w'	White

Yansıtıcı işaret yakalama

```
[B,L]=bwboundaries(bw,'noholes');
LRGB=label2rgb(L, @jet, [.5 .5 .5]);
figure,imshow(LRGB)
```

Taban için renk bilgisi : [gri]

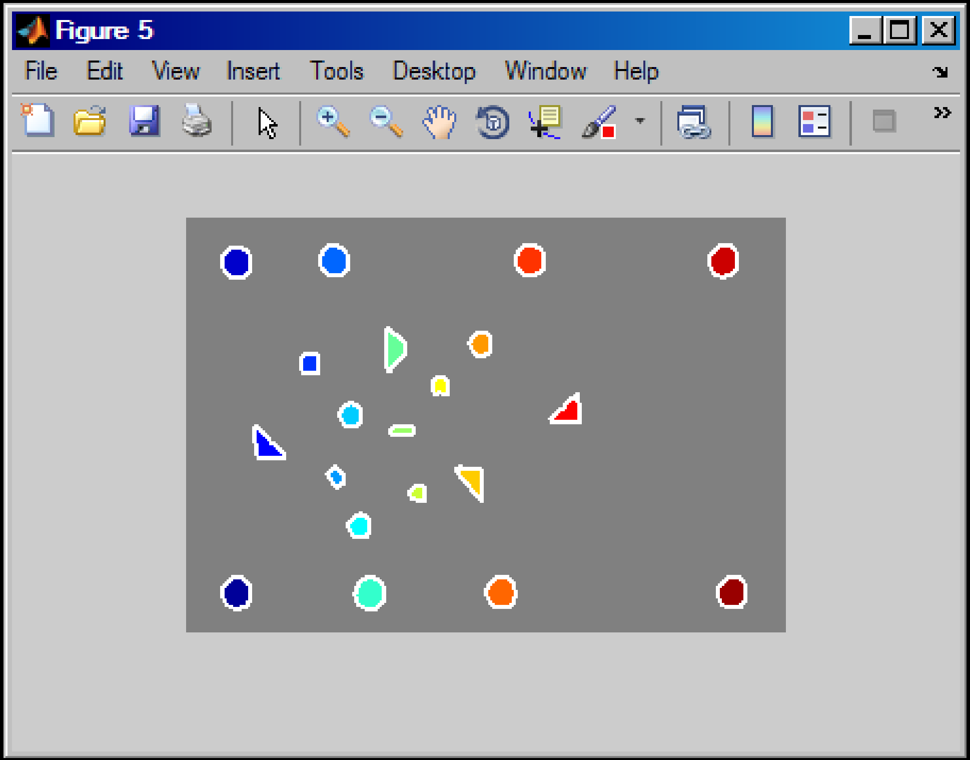


Yansıtıcı işaret yakalama

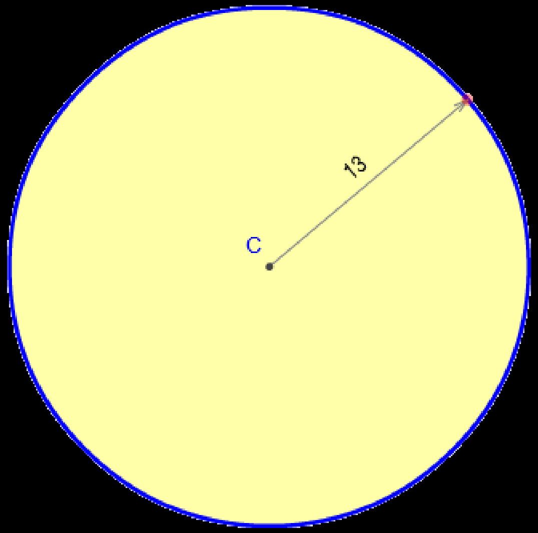
= 1 : 20

beyaz çizgi
ve kalınlığı 2

```
hold on
for k = 1:length(B)
 boundary = B{k};
 plot(boundary(:,2),boundary(:,1),'w', 'LineWidth',2)
end
```



Dairenin Geometrik Özellikleri



Çevre = 2.π.r = 2 x 3.1415 x 13 = 81.7

Çevre = π.d = 3.1415 x 26 = 81.7

Çevre = √4.π.Alan = √4 x 3.1415 x 530.9 = 81.7

Alan = π.r^2 = 3.1415 x (13)^2 = 530.9

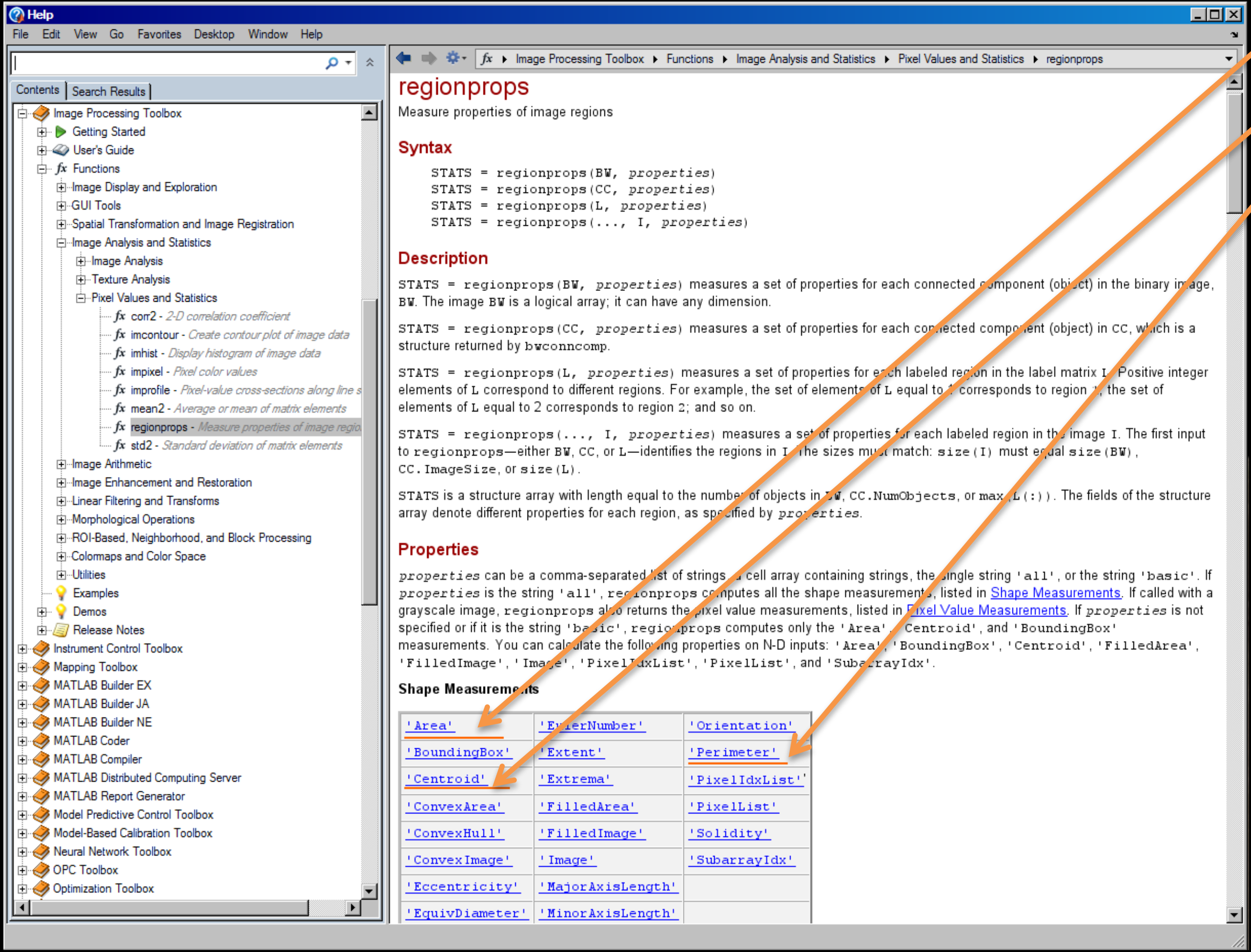
Alan = (Çevre)^2 / 4.π = (81.7)^2 / 4 x 3.1415 = 530.9

Yuvarlaklık Katsayısı = 4.π.Alan / (Çevre)^2 = 1

Yuvarlaklık Katsayısı ≈ 1 → Daha Yuvarlak

Yansıtıcı işaret yakalama

```
stats = regionprops(L, ...
                   'Area', ...
                   'Centroid', ...
                   'Perimeter');
```


Help
File Edit View Go Favorites Desktop Window Help
Contents
Search Results
Image Processing Toolbox
Getting Started
User's Guide
Functions
Image Display and Exploration
GUI Tools
Spatial Transformation and Image Registration
Image Analysis and Statistics
Image Analysis
Texture Analysis
Pixel Values and Statistics
corr2 - 2-D correlation coefficient
imcontour - Create contour plot of image data
imhist - Display histogram of image data
impixel - Pixel color values
improfile - Pixel-value cross-sections along line s
mean2 - Average or mean of matrix elements
regionprops - Measure properties of image regio
std2 - Standard deviation of matrix elements
Image Arithmetic
Image Enhancement and Restoration
Linear Filtering and Transforms
Morphological Operations
ROI-Based, Neighborhood, and Block Processing
Colormaps and Color Space
Utilities
Examples
Demos
Release Notes
Instrument Control Toolbox
Mapping Toolbox
MATLAB Builder EX
MATLAB Builder JA
MATLAB Builder NE
MATLAB Coder
MATLAB Compiler
MATLAB Distributed Computing Server
MATLAB Report Generator
Model Predictive Control Toolbox
Model-Based Calibration Toolbox
Neural Network Toolbox
OPC Toolbox
Optimization Toolbox
Image Processing Toolbox ▸ Functions ▸ Image Analysis and Statistics ▸ Pixel Values and Statistics ▸ regionprops

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'	'Extent'	'Perimeter'
'Centroid'	'Extrema'	'PixelIdxList'
'ConvexArea'	'FilledArea'	'PixelList'
'ConvexHull'	'FilledImage'	'Solidity'
'ConvexImage'	'Image'	'SubarrayIdx'
'Eccentricity'	'MajorAxisLength'	
'EquivDiameter'	'MinorAxisLength'	

Yansıtıcı işaret yakalama

0.95 <= Yuvarlaklık Katsayısı >= 1.05

```
ratioLow = 0.95;
ratioUp  = 1.05;
```

Yansıtıcı işaret yakalama

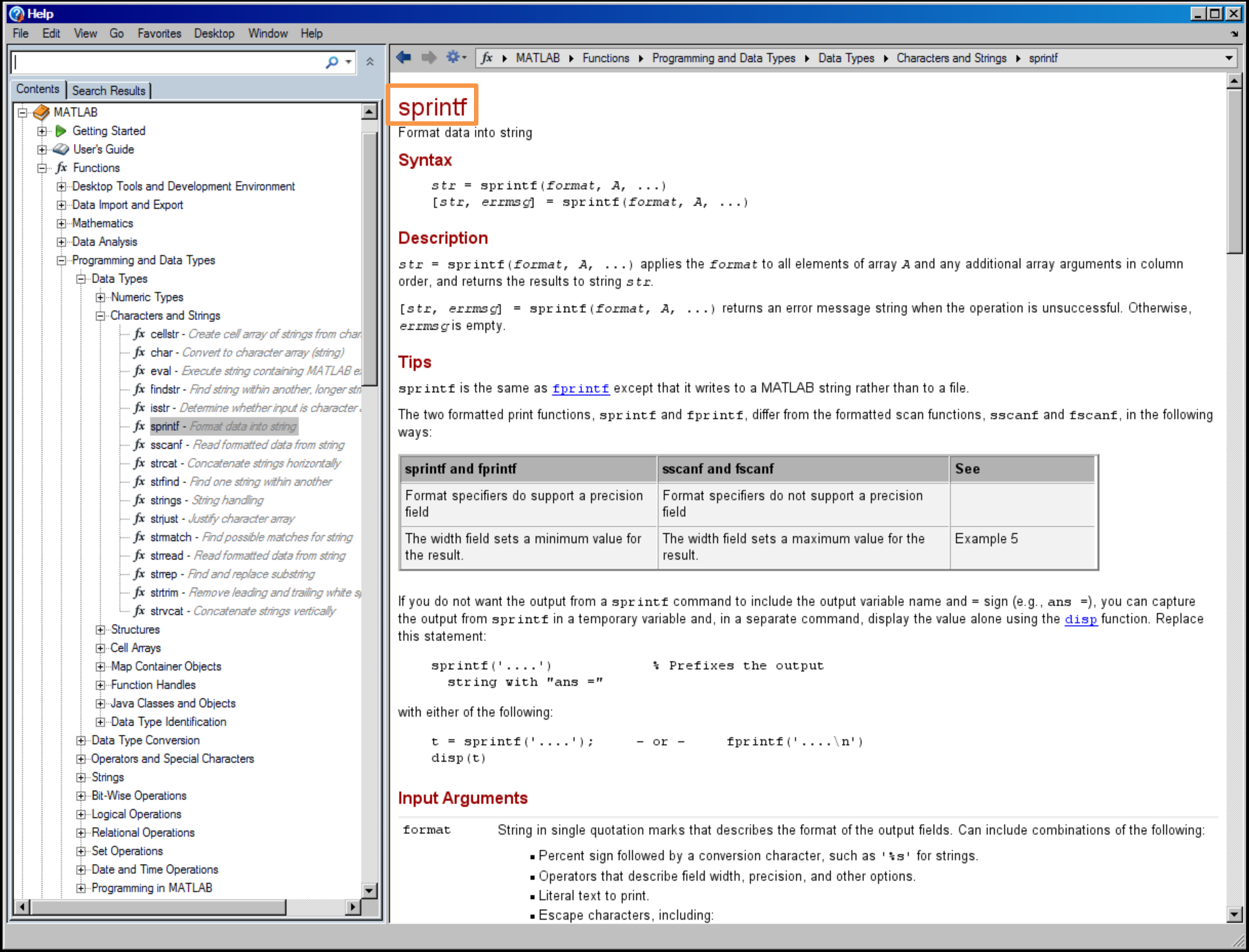
```
for k = 1:length(B)
      % 'k' etiketindeki nesnenin cevresi
      perimeter = stats(k).Perimeter;
      % 'k' etiketindeki nesnenin alanı
      area = stats(k).Area;
      % yuvarlaklık oranını hesapla
      ratio = 4*pi*area / perimeter^2;
      .
      .
      .

end
```

Yansıtıcı işaret yakalama

```
for k = 1:length(B)
      .
      .
      .
      % yuvarliklik oranini formatli
      % olarak degiskene yaz
      ratio_string = sprintf('%2.2f', ratio);
      .
      .
      .
end
```

Yansıtıcı işaret yakalama


Help
File Edit View Go Favorites Desktop Window Help
Contents | Search Results
MATLAB
Getting Started
User's Guide
Functions
Desktop Tools and Development Environment
Data Import and Export
Mathematics
Data Analysis
Programming and Data Types
Data Types
Numeric Types
Characters and Strings
cellstr - Create cell array of strings from char
char - Convert to character array (string)
eval - Execute string containing MATLAB e
findstr - Find string within another, longer str
isstr - Determine whether input is character
sprintf - Format data into string
sscanf - Read formatted data from string
strcat - Concatenate strings horizontally
strfind - Find one string within another
strings - String handling
strjust - Justify character array
strmatch - Find possible matches for string
strread - Read formatted data from string
strrep - Find and replace substring
strtrim - Remove leading and trailing white s
strvcat - Concatenate strings vertically
Structures
Cell Arrays
Map Container Objects
Function Handles
Java Classes and Objects
Data Type Identification
Data Type Conversion
Operators and Special Characters
Strings
Bit-Wise Operations
Logical Operations
Relational Operations
Set Operations
Date and Time Operations
Programming in MATLAB
MATLAB ▸ Functions ▸ Programming and Data Types ▸ Data Types ▸ Characters and Strings ▸ sprintf

sprintf

Format data into string

Syntax

```
str = sprintf(format, A, ...)
[str, errmsg] = sprintf(format, A, ...)
```

Description

`str = sprintf(format, A, ...)` applies the `format` to all elements of array `A` and any additional array arguments in column order, and returns the results to string `str`.

`[str, errmsg] = sprintf(format, A, ...)` returns an error message string when the operation is unsuccessful. Otherwise, `errmsg` is empty.

Tips

`sprintf` is the same as `fprintf` except that it writes to a MATLAB string rather than to a file.

The two formatted print functions, `sprintf` and `fprintf`, differ from the formatted scan functions, `sscanf` and `fscanf`, in the following ways:

sprintf and fprintf	sscanf and fscanf	See
Format specifiers do support a precision field	Format specifiers do not support a precision field	
The width field sets a minimum value for the result.	The width field sets a maximum value for the result.	Example 5

If you do not want the output from a `sprintf` command to include the output variable name and = sign (e.g., `ans =`), you can capture the output from `sprintf` in a temporary variable and, in a separate command, display the value alone using the `disp` function. Replace this statement:

```
sprintf('....')             % Prefixes the output
  string with "ans ="
```

with either of the following:

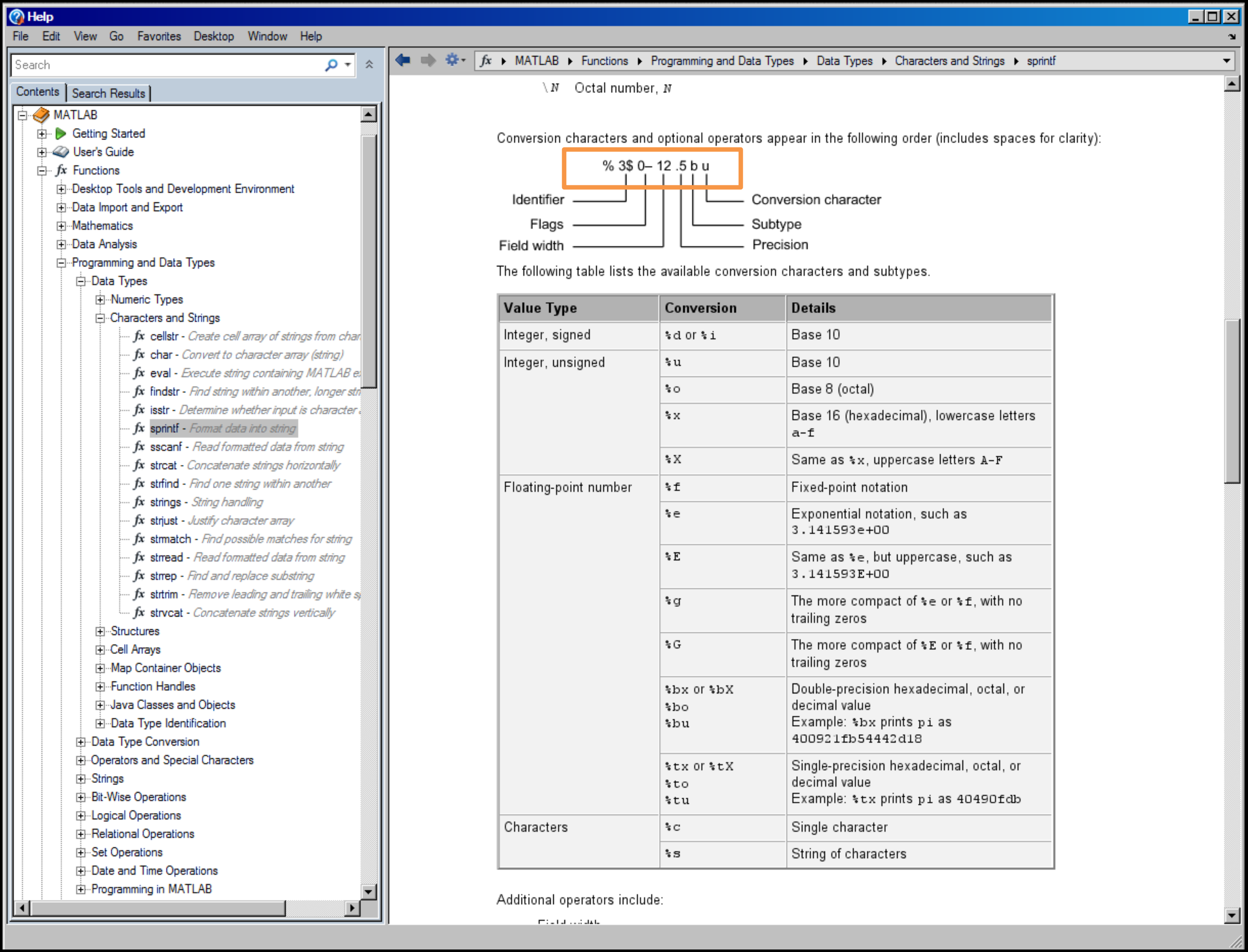
```
t = sprintf('....');     - or -     fprintf('....\n')
disp(t)
```

Input Arguments

`format` String in single quotation marks that describes the format of the output fields. Can include combinations of the following:

- Percent sign followed by a conversion character, such as `'%s'` for strings.
- Operators that describe field width, precision, and other options.
- Literal text to print.
- Escape characters, including:

Yansıtıcı işaret yakalama


Help
File Edit View Go Favorites Desktop Window Help
Search
Contents Search Results
MATLAB
Getting Started
User's Guide
Functions
Desktop Tools and Development Environment
Data Import and Export
Mathematics
Data Analysis
Programming and Data Types
Data Types
Numeric Types
Characters and Strings
cellstr - Create cell array of strings from char
char - Convert to character array (string)
eval - Execute string containing MATLAB e
findstr - Find string within another, longer str
isstr - Determine whether input is character
sprintf - Format data into string
sscanf - Read formatted data from string
strcat - Concatenate strings horizontally
strfind - Find one string within another
strings - String handling
strjust - Justify character array
strmatch - Find possible matches for string
strread - Read formatted data from string
strrep - Find and replace substring
strtrim - Remove leading and trailing white s
strvcat - Concatenate strings vertically
Structures
Cell Arrays
Map Container Objects
Function Handles
Java Classes and Objects
Data Type Identification
Data Type Conversion
Operators and Special Characters
Strings
Bit-Wise Operations
Logical Operations
Relational Operations
Set Operations
Date and Time Operations
Programming in MATLAB
MATLAB ▸ Functions ▸ Programming and Data Types ▸ Data Types ▸ Characters and Strings ▸ sprintf
% 3$ 0– 12 .5 b u
Identifier
Flags
Field width
Conversion character
Subtype
Precision

\N Octal number, N

Conversion characters and optional operators appear in the following order (includes spaces for clarity):

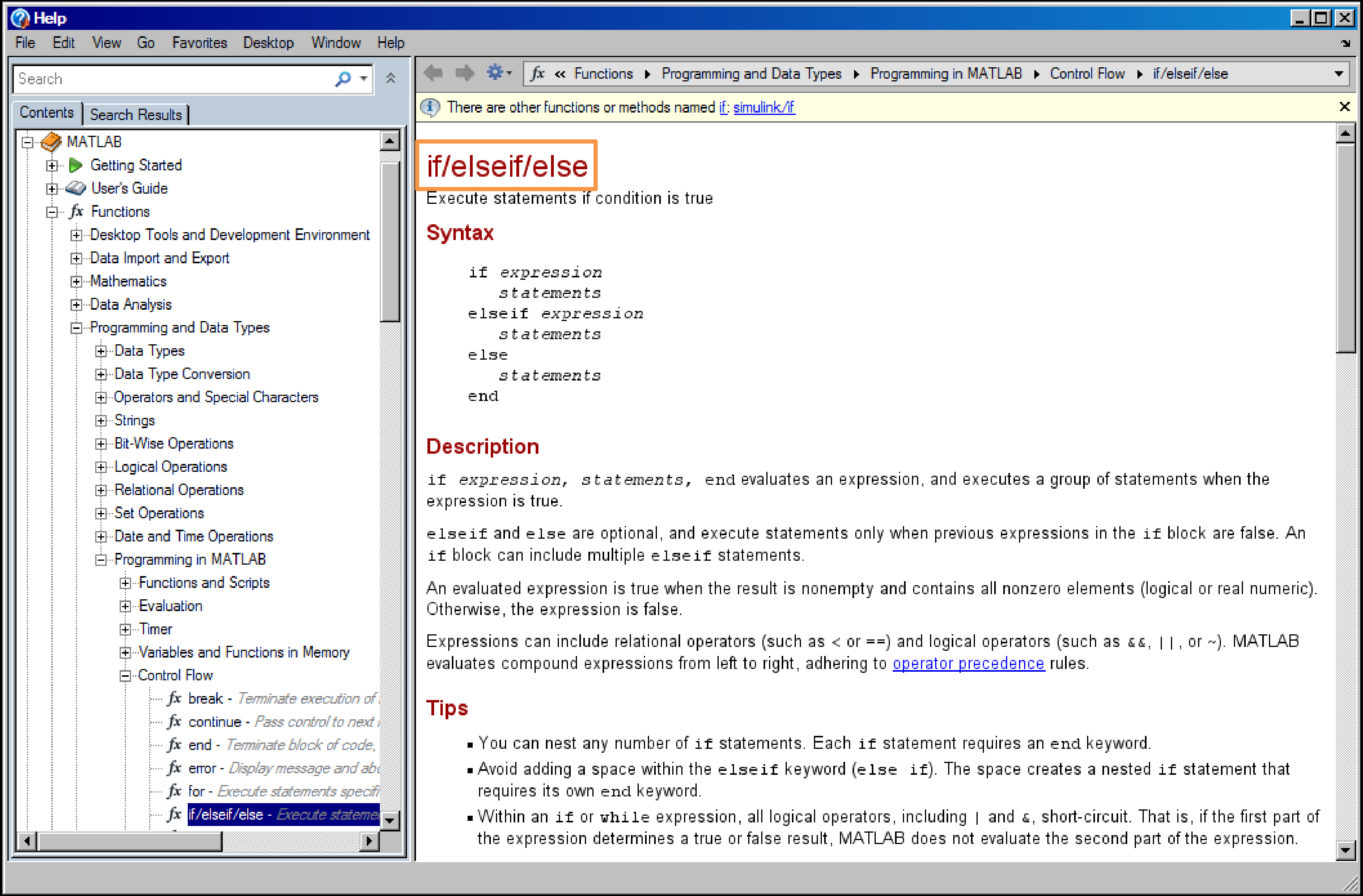
The following table lists the available conversion characters and subtypes.

Value Type	Conversion	Details
Integer, signed	`%d` or `%i`	Base 10
Integer, unsigned	`%u`	Base 10
	`%o`	Base 8 (octal)
	`%x`	Base 16 (hexadecimal), lowercase letters `a-f`
	`%X`	Same as `%x`, uppercase letters `A-F`
Floating-point number	`%f`	Fixed-point notation
	`%e`	Exponential notation, such as `3.141593e+00`
	`%E`	Same as `%e`, but uppercase, such as `3.141593E+00`
	`%g`	The more compact of `%e` or `%f`, with no trailing zeros
	`%G`	The more compact of `%E` or `%f`, with no trailing zeros
	`%bx` or `%bX` `%bo` `%bu`	Double-precision hexadecimal, octal, or decimal value Example: `%bx` prints `pi` as `400921fb54442d18`
	`%tx` or `%tX` `%to` `%tu`	Single-precision hexadecimal, octal, or decimal value Example: `%tx` prints `pi` as `40490fdb`
Characters	`%c`	Single character
	`%s`	String of characters

Additional operators include:

Yansıtıcı işaret yakalama

```
for k = 1:length(B)
      .
      .
      .
  % yuvarlaklık kriterine uyan nesnelerin ortasına
  % siyah yuvarlak isaret yerlestir
  if (ratio >= ratioLow) && (ratio <= ratioUp)
    centroid = stats(k).Centroid;
    plot(centroid(1), centroid(2),'ko');
    text(centroid(1) - 15, centroid(2) + 5,
      ratio_string,...
      'Color','y', ...
      'FontSize',14, ...
      'FontWeight','bold');
  end
end
```


Help
File Edit View Go Favorites Desktop Window Help
Search
Contents | Search Results
MATLAB
Getting Started
User's Guide
Functions
Desktop Tools and Development Environment
Data Import and Export
Mathematics
Data Analysis
Programming and Data Types
Data Types
Data Type Conversion
Operators and Special Characters
Strings
Bit-Wise Operations
Logical Operations
Relational Operations
Set Operations
Date and Time Operations
Programming in MATLAB
Functions and Scripts
Evaluation
Timer
Variables and Functions in Memory
Control Flow
break - Terminate execution of
continue - Pass control to next
end - Terminate block of code,
error - Display message and ab
for - Execute statements specifi
if/elseif/else - Execute stateme
« Functions ▸ Programming and Data Types ▸ Programming in MATLAB ▸ Control Flow ▸ if/elseif/else

There are other functions or methods named if: simulink/if

if/elseif/else

Execute statements if condition is true

Syntax

```
if expression
   statements
elseif expression
   statements
else
   statements
end
```

Description

`if expression, statements, end` evaluates an expression, and executes a group of statements when the expression is true.

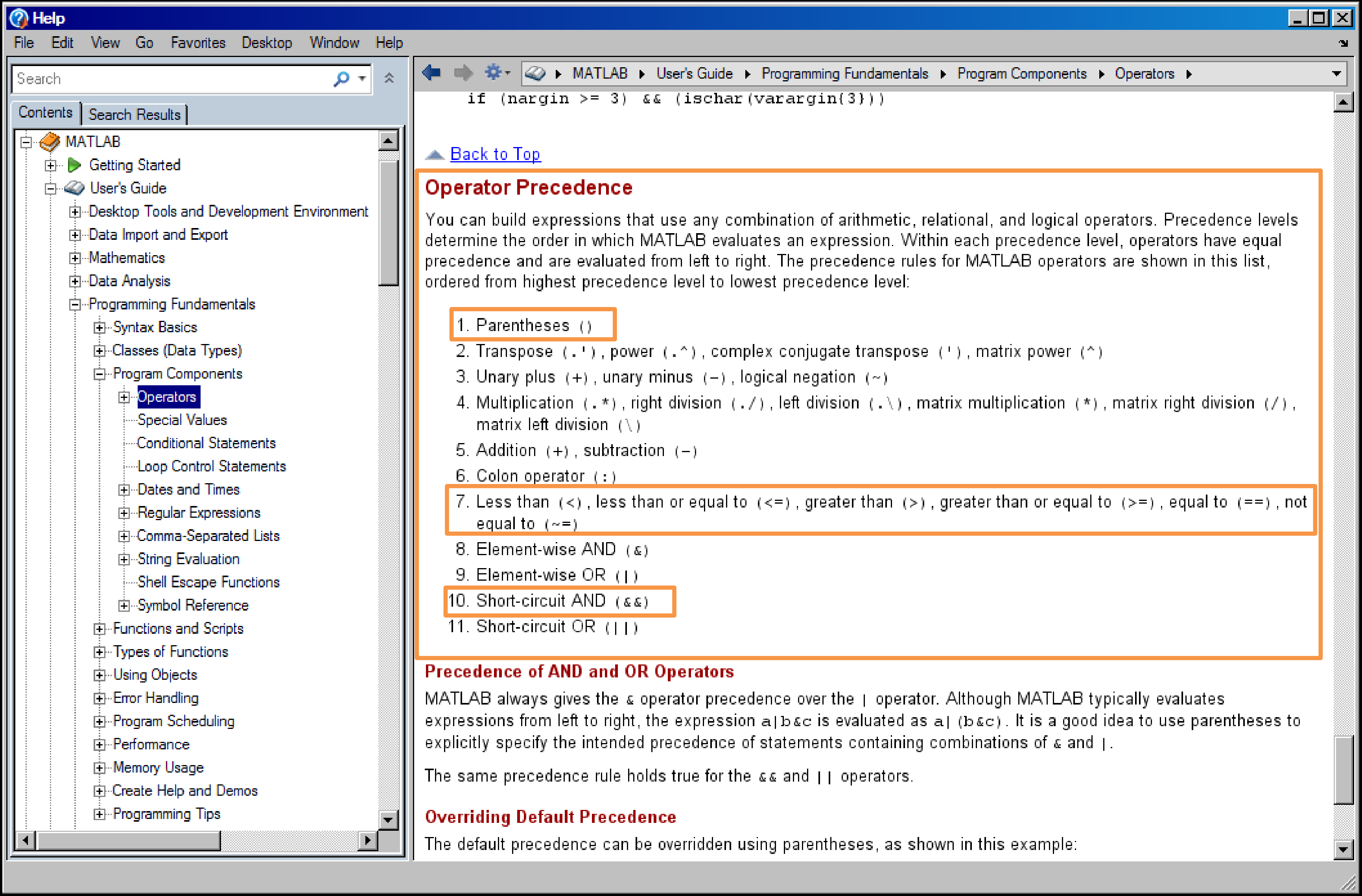
`elseif` and `else` are optional, and execute statements only when previous expressions in the `if` block are false. An `if` block can include multiple `elseif` statements.

An evaluated expression is true when the result is nonempty and contains all nonzero elements (logical or real numeric). Otherwise, the expression is false.

Expressions can include relational operators (such as < or ==) and logical operators (such as `&&`, `||`, or `~`). MATLAB evaluates compound expressions from left to right, adhering to operator precedence rules.

Tips

- You can nest any number of `if` statements. Each `if` statement requires an `end` keyword.
- Avoid adding a space within the `elseif` keyword (`else if`). The space creates a nested `if` statement that requires its own `end` keyword.
- Within an `if` or `while` expression, all logical operators, including `|` and `&`, short-circuit. That is, if the first part of the expression determines a true or false result, MATLAB does not evaluate the second part of the expression.



```
if (nargin >= 3) && (ischar(varargin{3}))
```

Back to Top

Operator Precedence

You can build expressions that use any combination of arithmetic, relational, and logical operators. Precedence levels determine the order in which MATLAB evaluates an expression. Within each precedence level, operators have equal precedence and are evaluated from left to right. The precedence rules for MATLAB operators are shown in this list, ordered from highest precedence level to lowest precedence level:

1. Parentheses `()`
2. Transpose `(.')`, power `(.^)`, complex conjugate transpose `(')`, matrix power `(^)`
3. Unary plus `(+)`, unary minus `(-)`, logical negation `(~)`
4. Multiplication `(.*)`, right division `(./)`, left division `(.\)`, matrix multiplication `(*)`, matrix right division `(/)`, matrix left division `(\)`
5. Addition `(+)`, subtraction `(-)`
6. Colon operator `(:)`
7. Less than `(<)`, less than or equal to `(<=)`, greater than `(>)`, greater than or equal to `(>=)`, equal to `(==)`, not equal to `(~=)`
8. Element-wise AND `(&)`
9. Element-wise OR `(|)`
10. Short-circuit AND `(&&)`
11. Short-circuit OR `(||)`

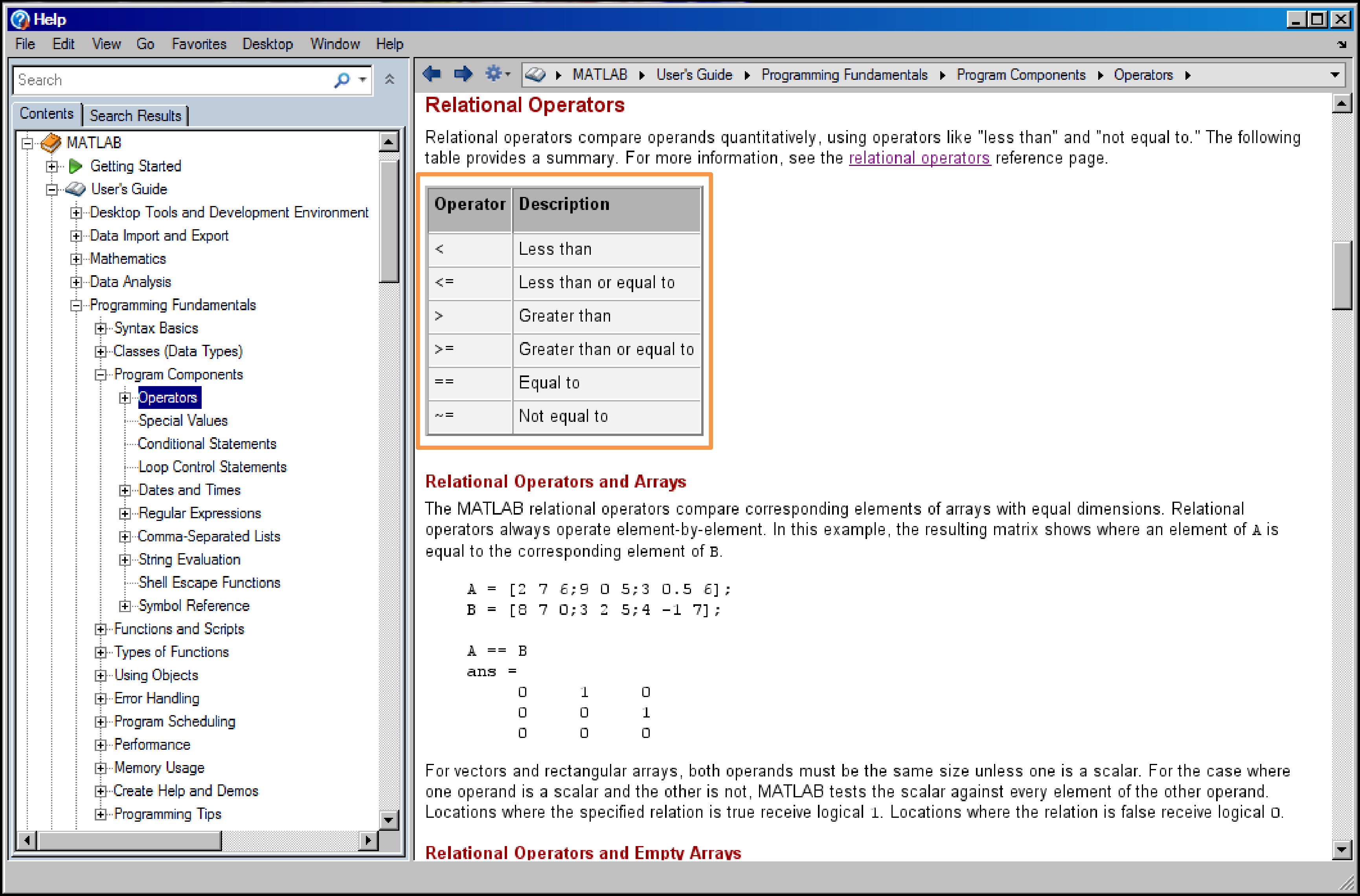
Precedence of AND and OR Operators

MATLAB always gives the `&` operator precedence over the `|` operator. Although MATLAB typically evaluates expressions from left to right, the expression `a|b&c` is evaluated as `a|(b&c)`. It is a good idea to use parentheses to explicitly specify the intended precedence of statements containing combinations of `&` and `|`.

The same precedence rule holds true for the `&&` and `||` operators.

Overriding Default Precedence

The default precedence can be overridden using parentheses, as shown in this example:



Relational Operators

Relational operators compare operands quantitatively, using operators like "less than" and "not equal to." The following table provides a summary. For more information, see the relational operators reference page.

Operator	Description
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
==	Equal to
~=	Not equal to

Relational Operators and Arrays

The MATLAB relational operators compare corresponding elements of arrays with equal dimensions. Relational operators always operate element-by-element. In this example, the resulting matrix shows where an element of A is equal to the corresponding element of B.

```
A = [2 7 6;9 0 5;3 0.5 6];
B = [8 7 0;3 2 5;4 -1 7];

A == B
ans =
     0     1     0
     0     0     1
     0     0     0
```

For vectors and rectangular arrays, both operands must be the same size unless one is a scalar. For the case where one operand is a scalar and the other is not, MATLAB tests the scalar against every element of the other operand. Locations where the specified relation is true receive logical 1. Locations where the relation is false receive logical 0.

Relational Operators and Empty Arrays

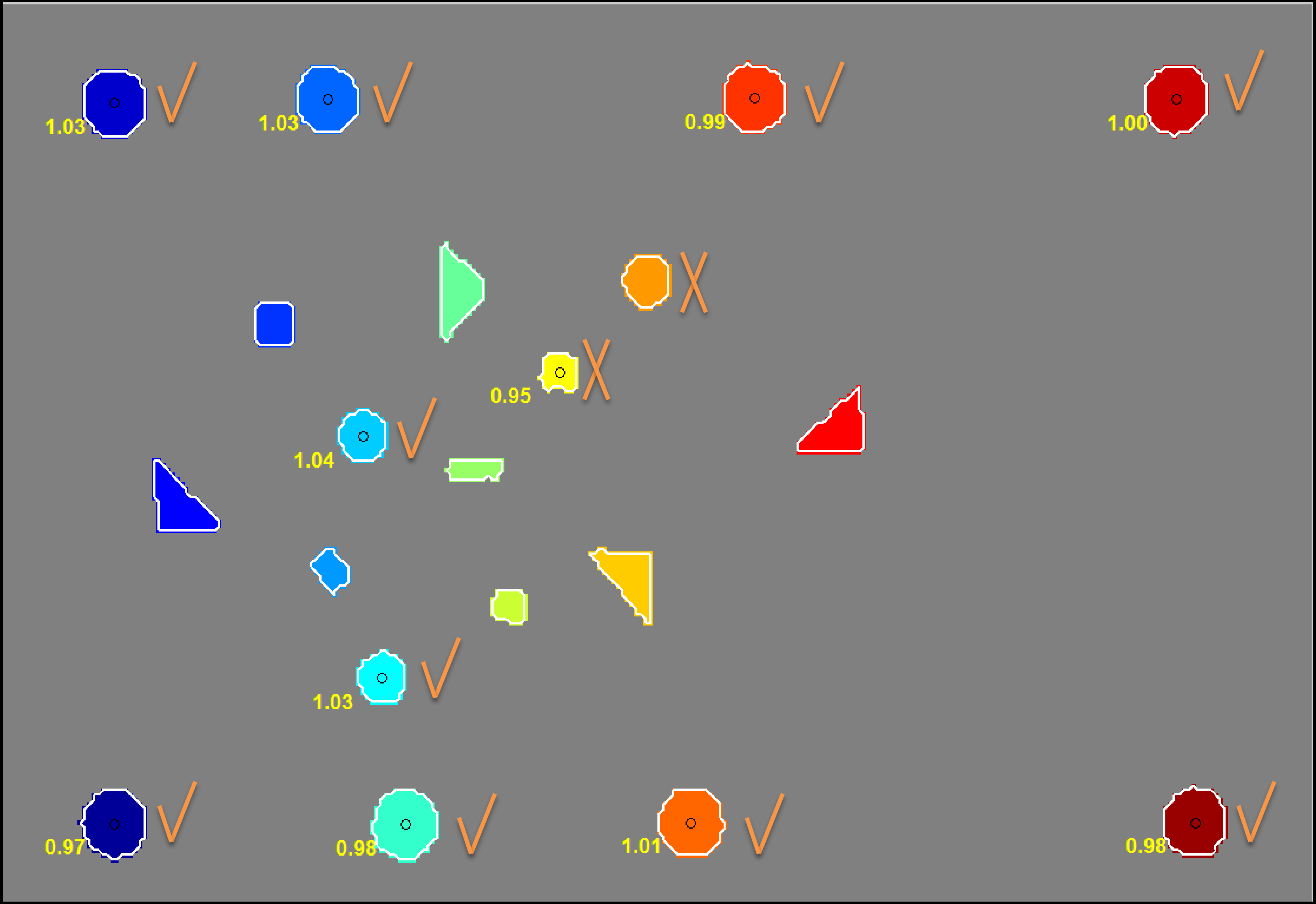
Yansıtıcı işaret yakalama

1

7

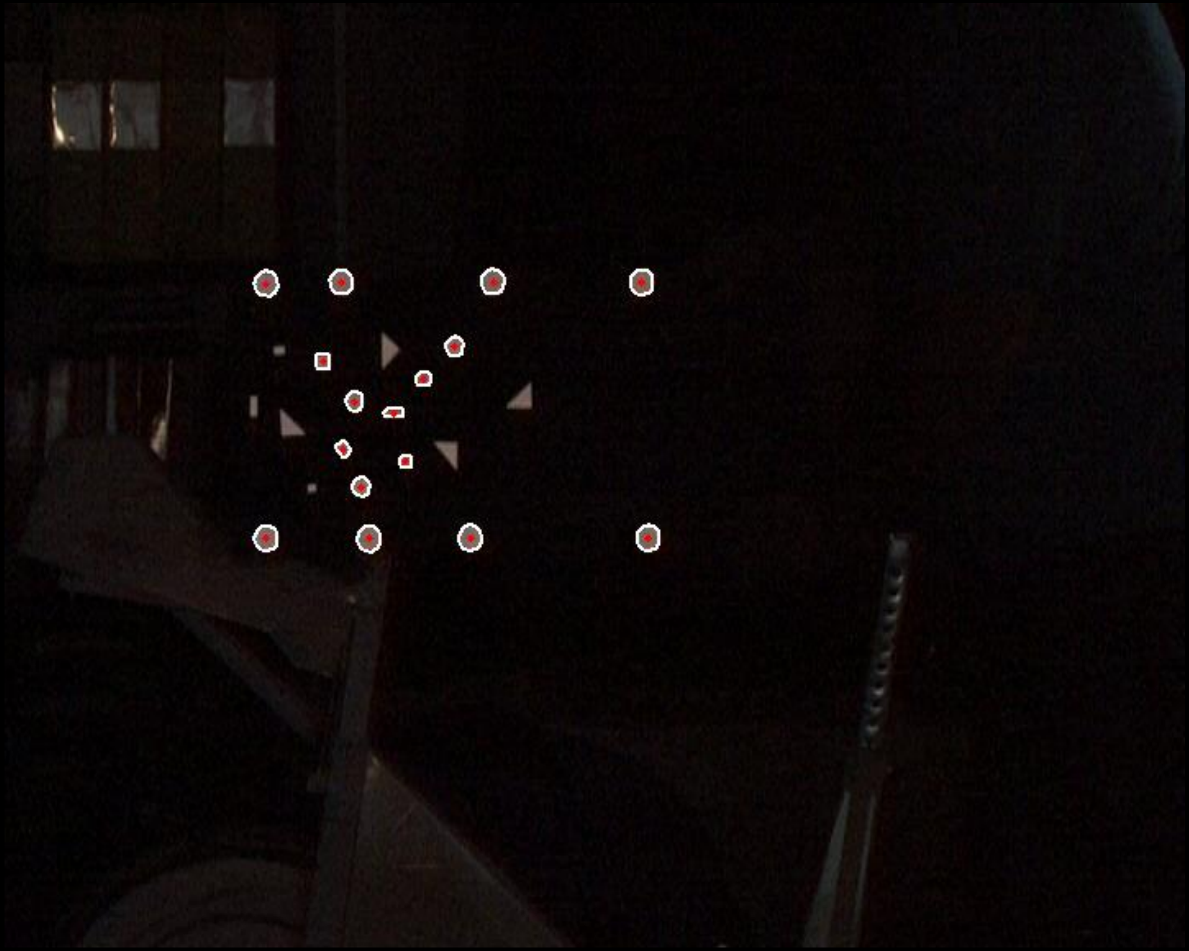
10

```
for k = 1:length(B)
      .
      .
      .
  % yuvarlaklik kriterine uyan nesnelerin ortasına
  % siyah yuvarlak isaret yerlestir
  if (ratio >= ratioLow && ratio <= ratioUp) !
     centroid = stats(k).Centroid;
     plot(centroid(1), centroid(2),'ko');
     text(centroid(1) - 15, centroid(2) + 5,
        ratio_string,...
        'Color','y', ...
        'FontSize',14, ...
        'FontWeight','bold');
  end
end
```

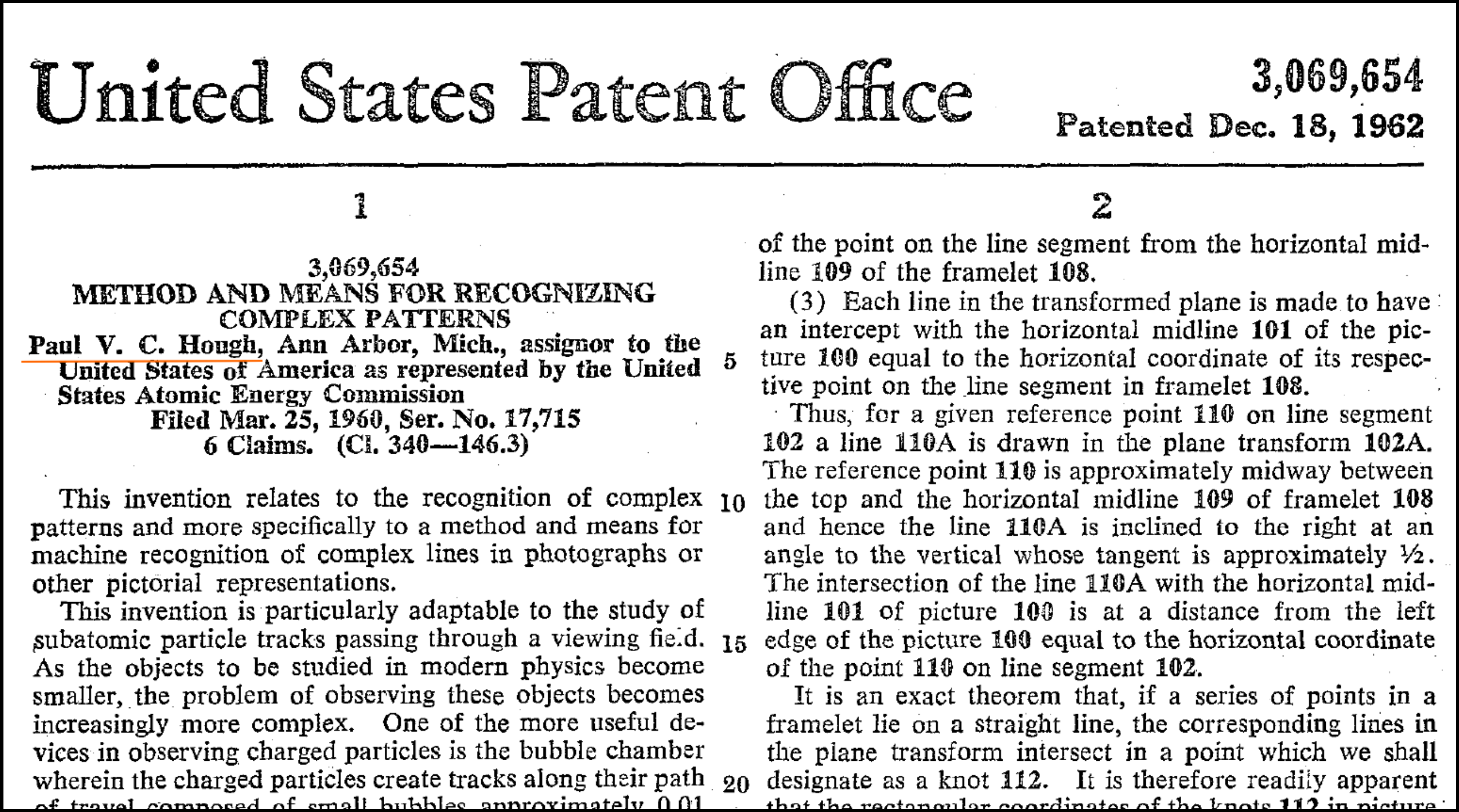

1.03
1.03
0.99
1.00
0.95
1.04
1.03
0.97
0.98
1.01
0.98

Sınıf Çalışması

Hareket sırasındaki yuvarlak yansıtıcıları bulan bir program yazınız.



Çember Bulmayı Nasıl Geliştirebiliriz ?



United States Patent Office

3,069,654
Patented Dec. 18, 1962

1

3,069,654
METHOD AND MEANS FOR RECOGNIZING COMPLEX PATTERNS
Paul V. C. Hough, Ann Arbor, Mich., assignor to the United States of America as represented by the United States Atomic Energy Commission
Filed Mar. 25, 1960, Ser. No. 17,715
6 Claims. (Cl. 340—146.3)

This invention relates to the recognition of complex patterns and more specifically to a method and means for machine recognition of complex lines in photographs or other pictorial representations.

This invention is particularly adaptable to the study of subatomic particle tracks passing through a viewing field. As the objects to be studied in modern physics become smaller, the problem of observing these objects becomes increasingly more complex. One of the more useful devices in observing charged particles is the bubble chamber wherein the charged particles create tracks along their path

2

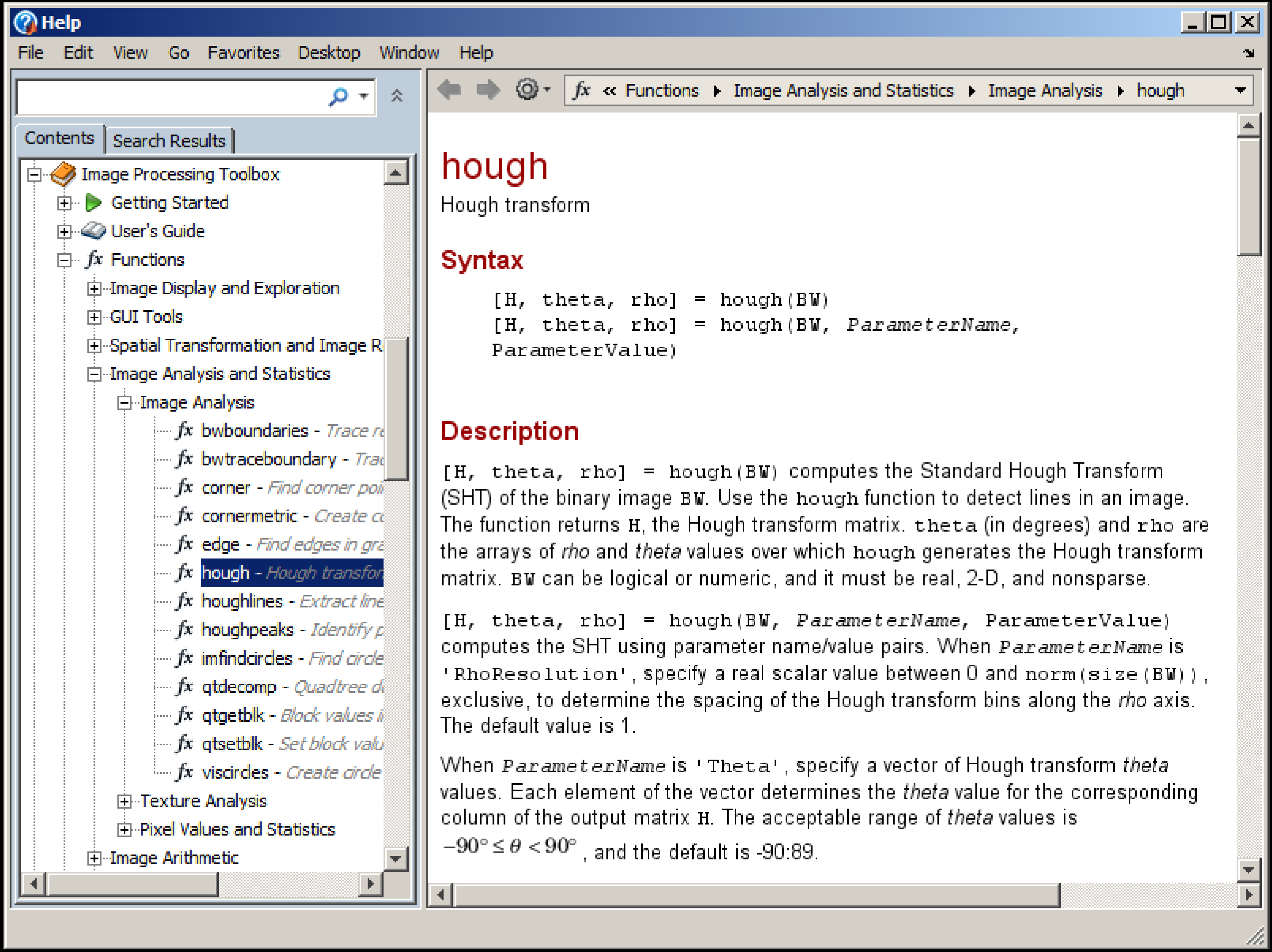
of the point on the line segment from the horizontal midline 109 of the framelet 108.

(3) Each line in the transformed plane is made to have an intercept with the horizontal midline 101 of the picture 100 equal to the horizontal coordinate of its respective point on the line segment in framelet 108.

Thus, for a given reference point 110 on line segment 102 a line 110A is drawn in the plane transform 102A. The reference point 110 is approximately midway between the top and the horizontal midline 109 of framelet 108 and hence the line 110A is inclined to the right at an angle to the vertical whose tangent is approximately ½. The intersection of the line 110A with the horizontal midline 101 of picture 100 is at a distance from the left edge of the picture 100 equal to the horizontal coordinate of the point 110 on line segment 102.

It is an exact theorem that, if a series of points in a framelet lie on a straight line, the corresponding lines in the plane transform intersect in a point which we shall designate as a knot 112. It is therefore readily apparent

MATLAB da Standart Hough Transformation


Help
File Edit View Go Favorites Desktop Window Help
Contents Search Results
Image Processing Toolbox
Getting Started
User's Guide
Functions
Image Display and Exploration
GUI Tools
Spatial Transformation and Image R
Image Analysis and Statistics
Image Analysis
bwboundaries - Trace re
bwtraceboundary - Trac
corner - Find corner poi
cornermetric - Create c
edge - Find edges in gra
hough - Hough transfor
houghlines - Extract line
houghpeaks - Identify p
imfindcircles - Find circle
qtdecomp - Quadtree d
qtgetblk - Block values i
qtsetblk - Set block valu
viscircles - Create circle
Texture Analysis
Pixel Values and Statistics
Image Arithmetic
« Functions ▸ Image Analysis and Statistics ▸ Image Analysis ▸ hough

hough

Hough transform

Syntax

```
[H, theta, rho] = hough(BW)
[H, theta, rho] = hough(BW, ParameterName,
ParameterValue)
```

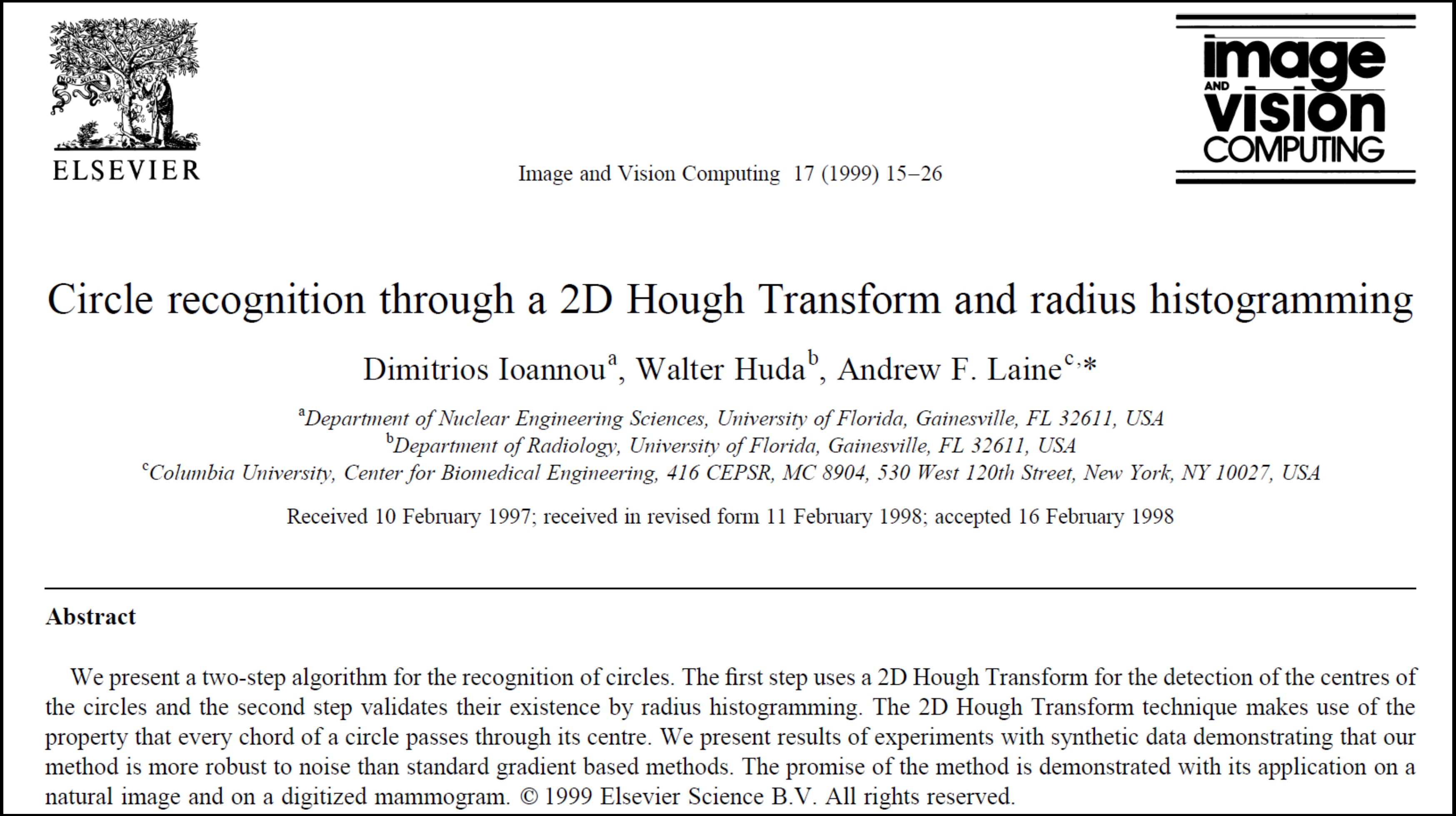
Description

`[H, theta, rho] = hough(BW)` computes the Standard Hough Transform (SHT) of the binary image `BW`. Use the `hough` function to detect lines in an image. The function returns `H`, the Hough transform matrix. `theta` (in degrees) and `rho` are the arrays of *rho* and *theta* values over which `hough` generates the Hough transform matrix. `BW` can be logical or numeric, and it must be real, 2-D, and nonsparse.

`[H, theta, rho] = hough(BW, ParameterName, ParameterValue)` computes the SHT using parameter name/value pairs. When `ParameterName` is `'RhoResolution'`, specify a real scalar value between 0 and `norm(size(BW))`, exclusive, to determine the spacing of the Hough transform bins along the *rho* axis. The default value is 1.

When `ParameterName` is `'Theta'`, specify a vector of Hough transform *theta* values. Each element of the vector determines the *theta* value for the corresponding column of the output matrix `H`. The acceptable range of *theta* values is $-90° \le \theta < 90°$, and the default is -90:89.

İşimize Yarar mı?



ELSEVIER
Image and Vision Computing 17 (1999) 15–26
image AND vision COMPUTING

Circle recognition through a 2D Hough Transform and radius histogramming

Dimitrios Ioannou[a], Walter Huda[b], Andrew F. Laine[c,*]

[a]*Department of Nuclear Engineering Sciences, University of Florida, Gainesville, FL 32611, USA*
[b]*Department of Radiology, University of Florida, Gainesville, FL 32611, USA*
[c]*Columbia University, Center for Biomedical Engineering, 416 CEPSR, MC 8904, 530 West 120th Street, New York, NY 10027, USA*

Received 10 February 1997; received in revised form 11 February 1998; accepted 16 February 1998

Abstract

We present a two-step algorithm for the recognition of circles. The first step uses a 2D Hough Transform for the detection of the centres of the circles and the second step validates their existence by radius histogramming. The 2D Hough Transform technique makes use of the property that every chord of a circle passes through its centre. We present results of experiments with synthetic data demonstrating that our method is more robust to noise than standard gradient based methods. The promise of the method is demonstrated with its application on a natural image and on a digitized mammogram. © 1999 Elsevier Science B.V. All rights reserved.

Ev Ödevi #2

1. Sınıf çalışmasında yazdığımız program
2. Ayrıca Hugh dönüşümünü kullanarak yansıtıcı işaretlerin yerini bulmaya çalışınız.

Ödev Teslim Tarihi: 1 Kasım 2018 Perşembe Saat 10 : 00

Teslim adresi : serdar.aritan@hacettepe.edu.tr

: serdar.aritan@gmail.com

Konu : BCA607 Odev 2 <Öğrenci No>

Öğrendiğimiz MATLAB fonksiyonları:

```
rgb2gray
bwareaopen
bwboundaries
label2rgb
sprintf
if / elseif / else / end
hough
```