BCA 607 Hareket Analizi Sistemleri

Matlab ile Görüntü İşleme 2

De Motu Animalium G.Borelli (1680)

SERDAR ARITAN

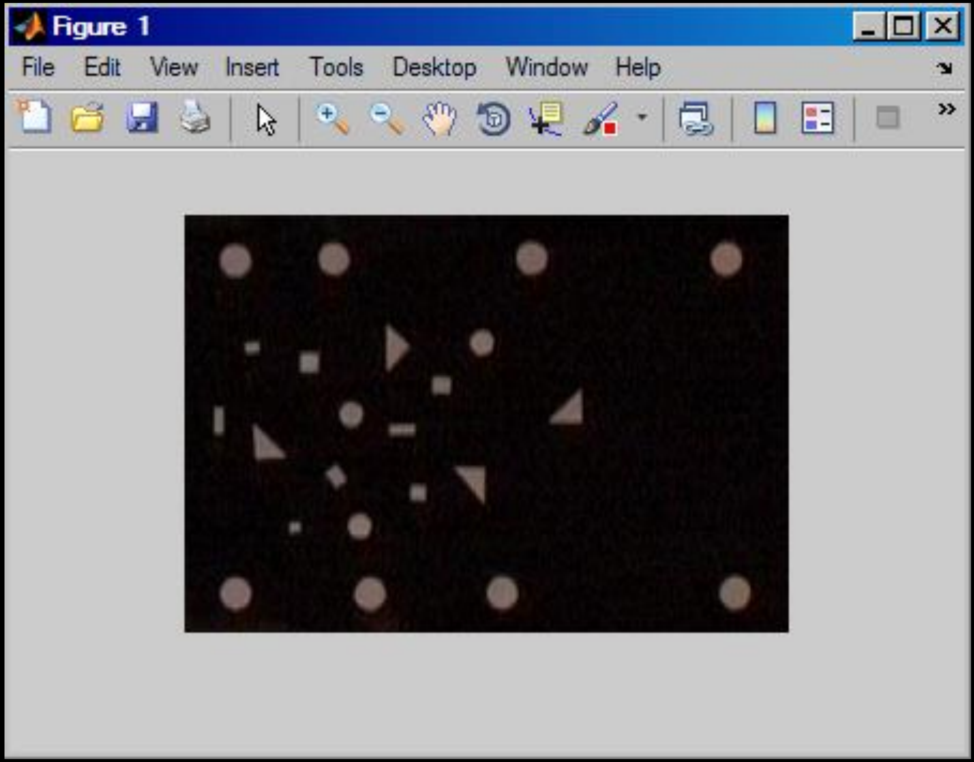
serdar.aritan@hacettepe.edu.tr

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

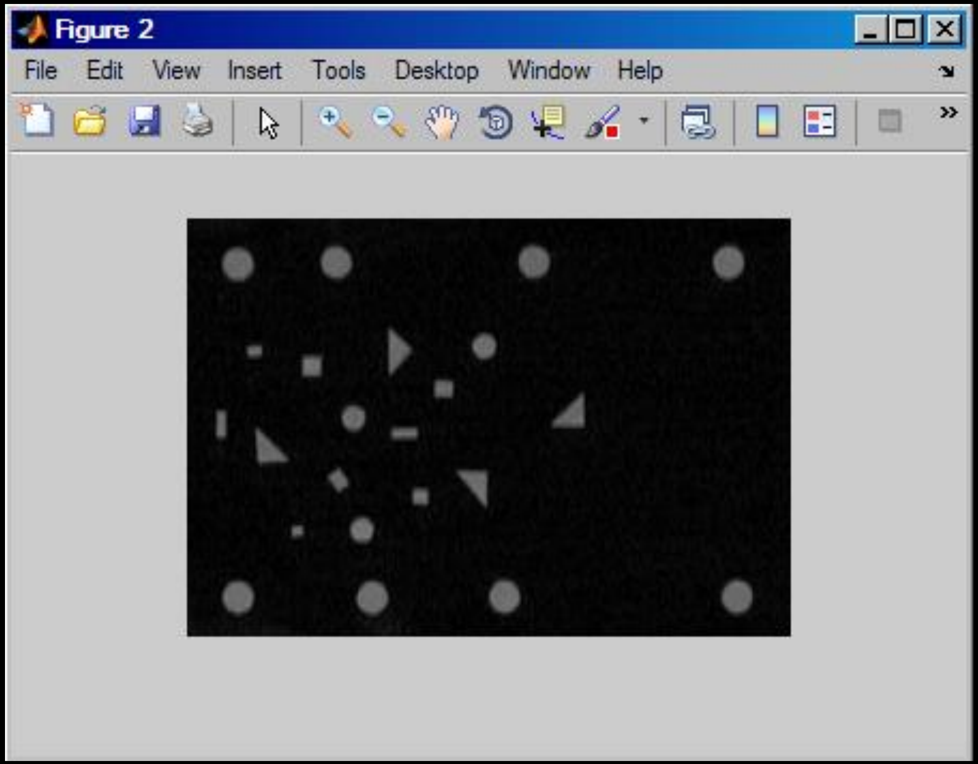
Yansıtıcı işaret yakalama

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

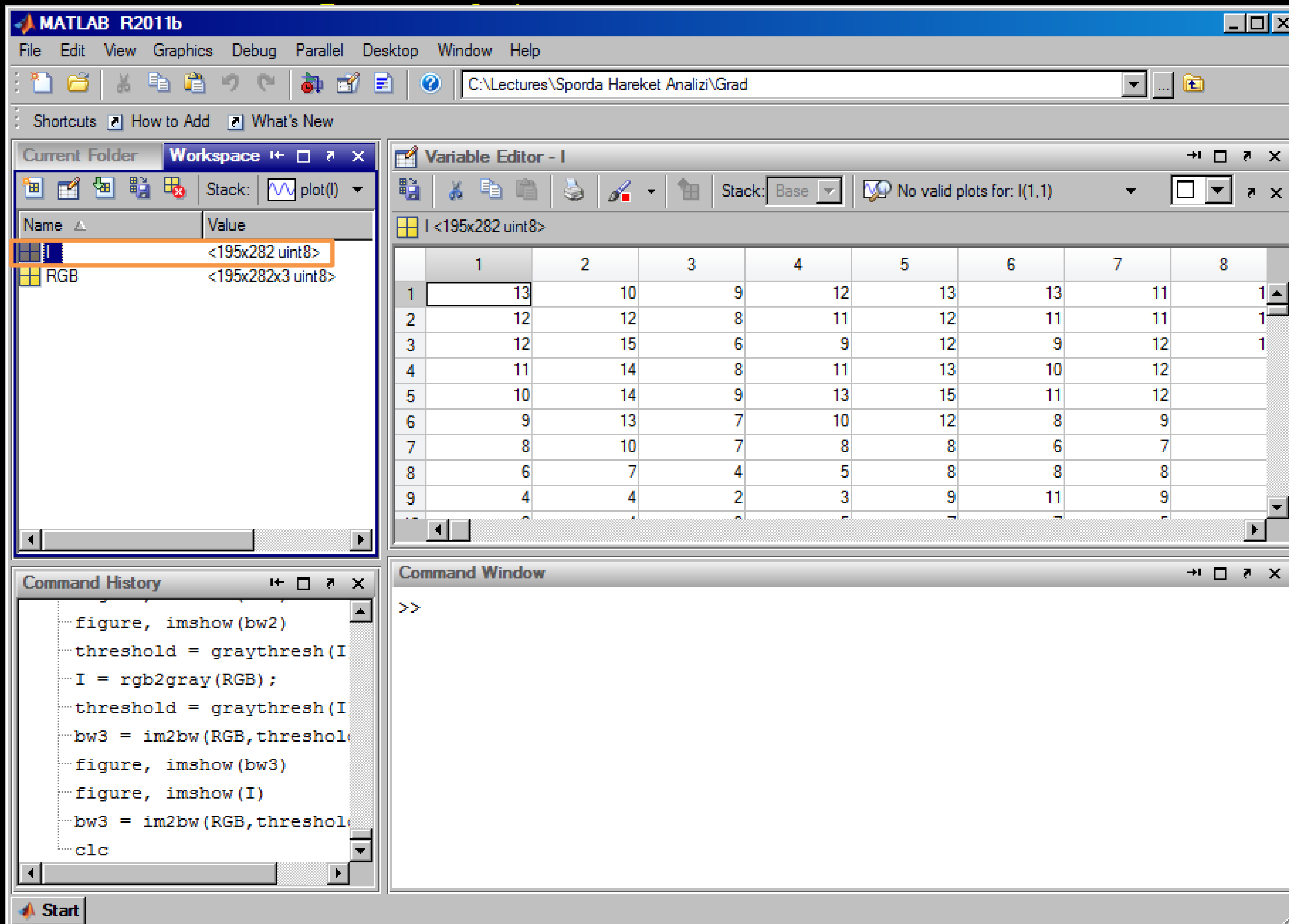


Yansıtıcı işaret yakalama

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

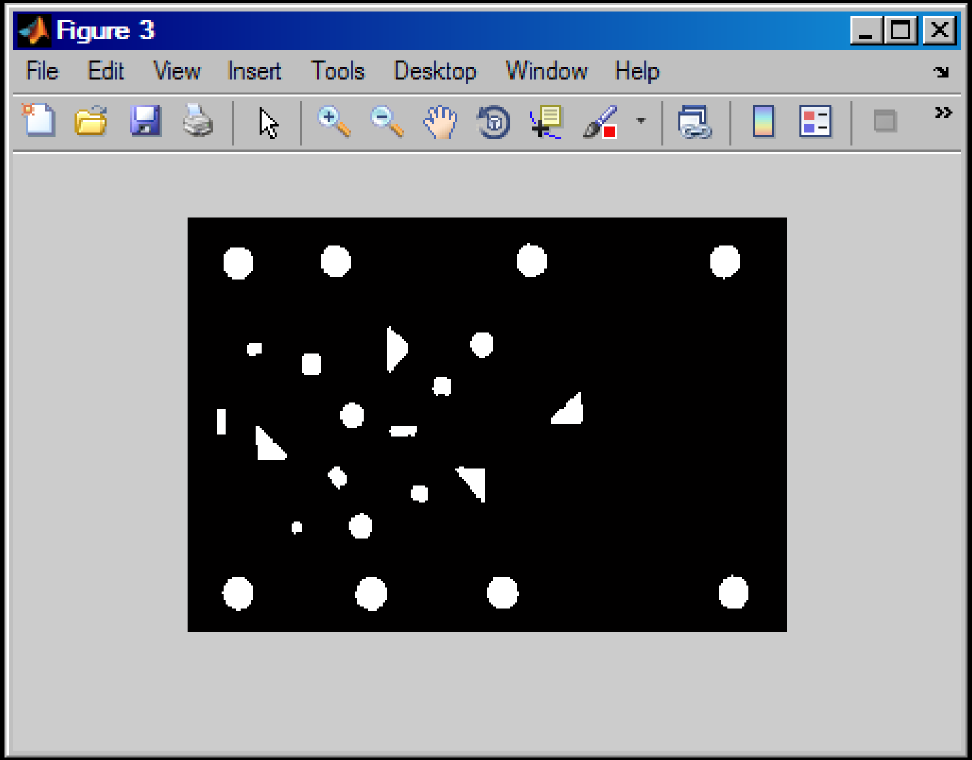


Yansıtıcı işaret yakalama



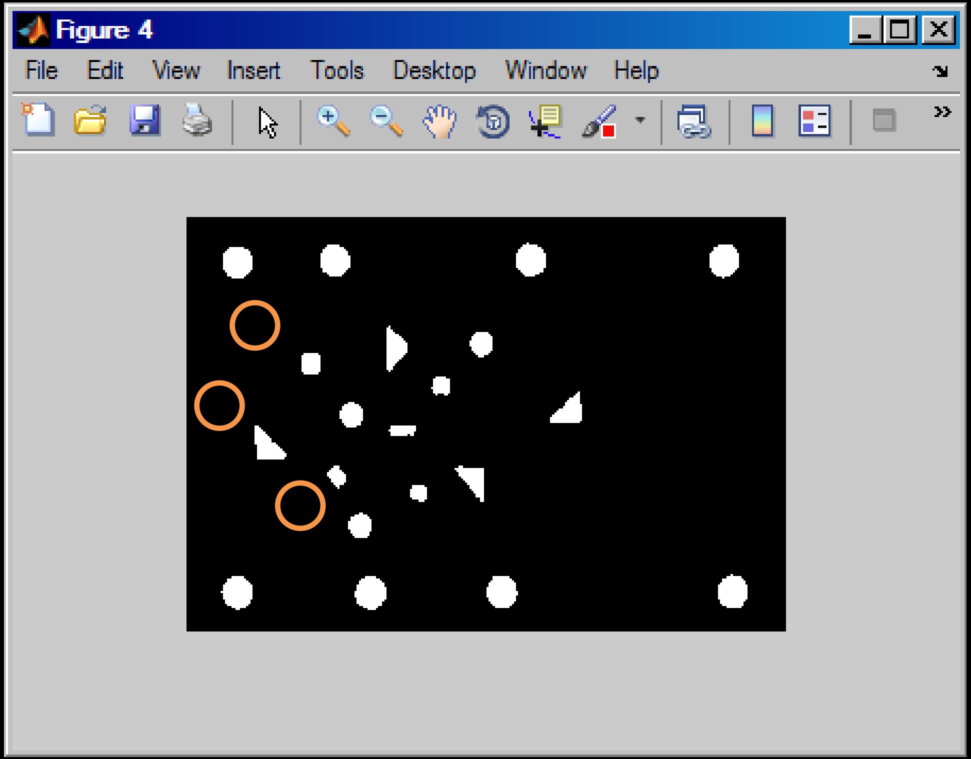
Yansıtıcı işaret yakalama

```
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

bwboundaries

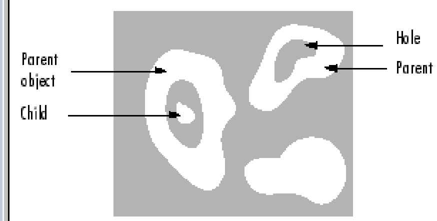
Trace region boundaries in binary image

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.

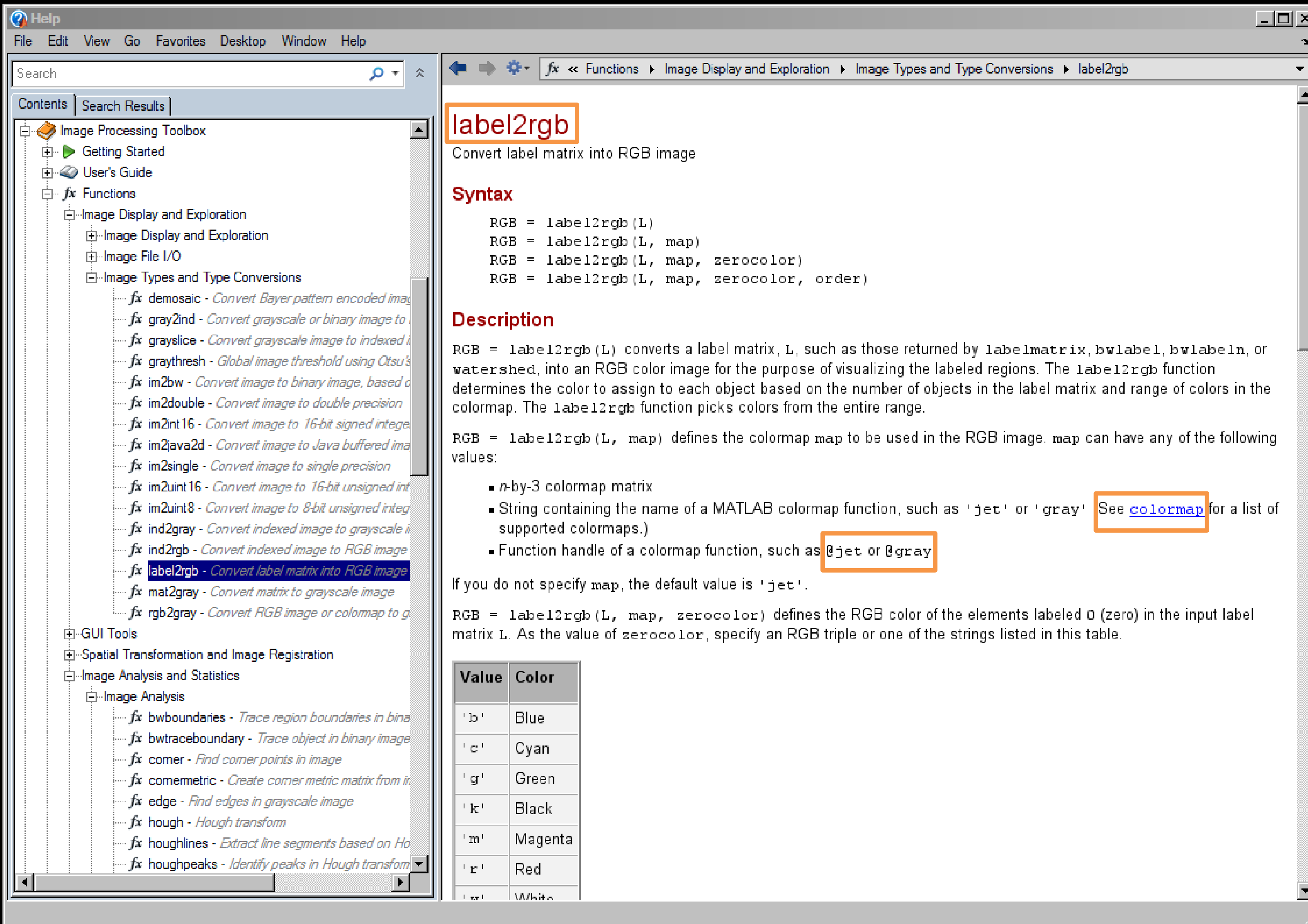


`bwboundaries` returns `B`, a P-by-1 cell array, where P is the number of objects and holes. Each cell in the cell array contains a Q-by-2 matrix. Each row in the matrix contains the row and column coordinates of a boundary pixel. Q is the number of boundary pixels for the corresponding region.

`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.

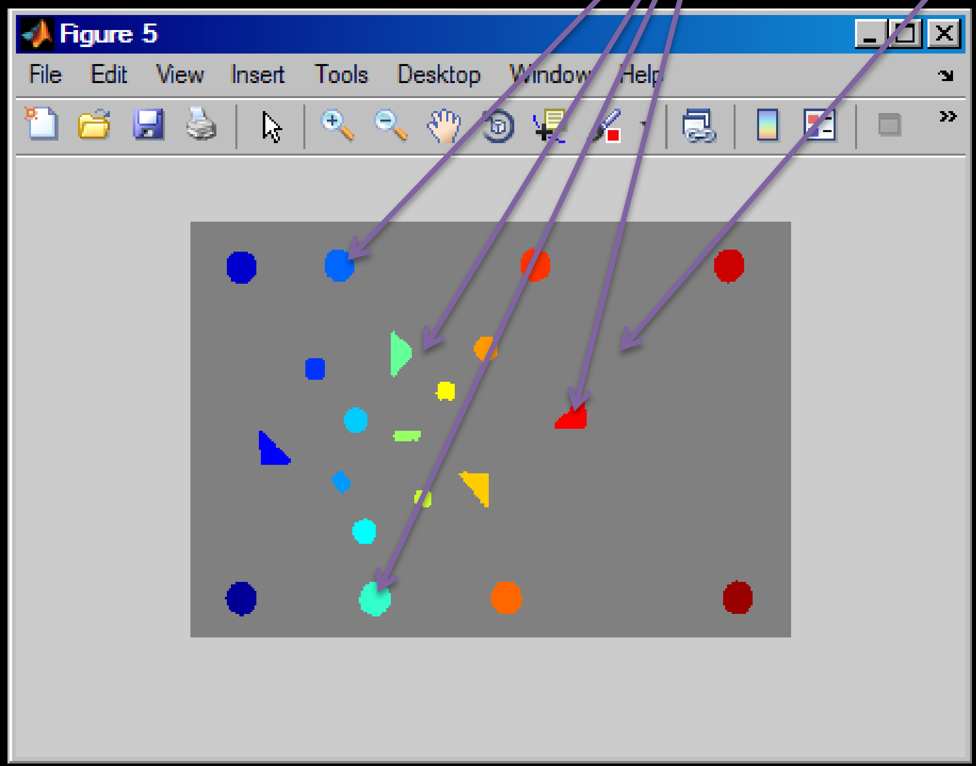
Yansıtıcı işaret yakalama



Yansıtıcı işaret yakalama

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

Taban için renk bilgisi : [gri]

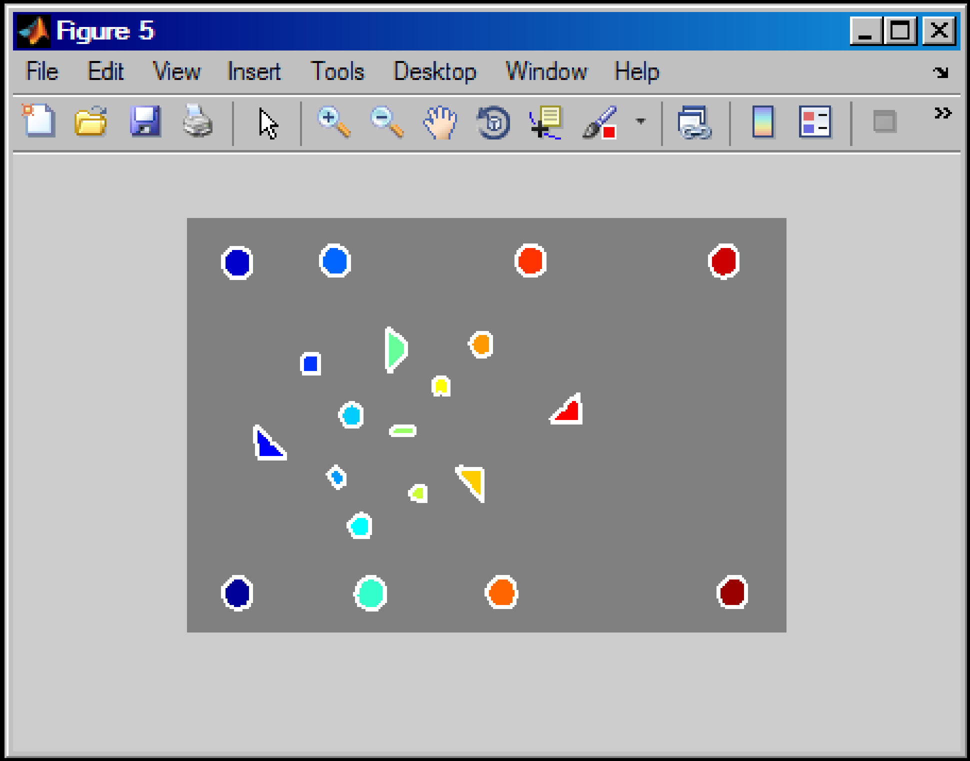


Yansıtıcı işaret yakalama

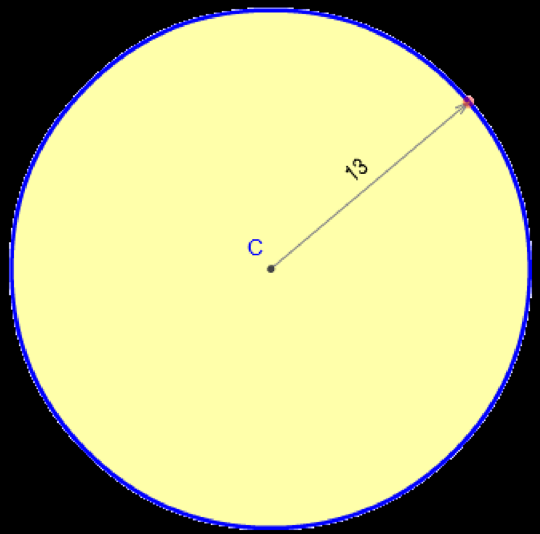
= 1 : 20

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 = $\pi.r^2$ = 3.1415 x $(13)^2$ = 530.9

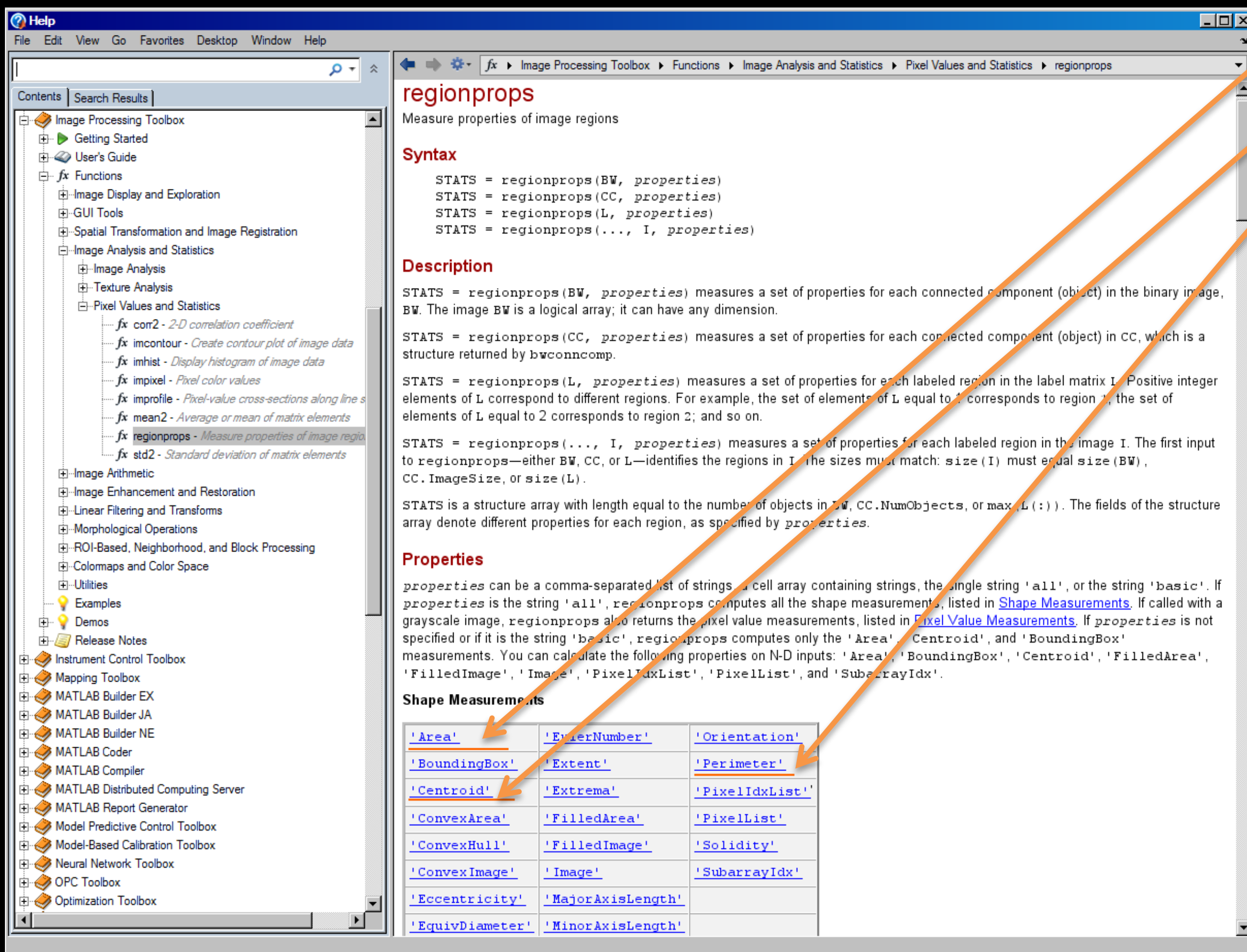
Alan = $(\text{Çevre})^2$ / 4.π = $(81.7)^2$ / 4 x 3.1415 = 530.9

Yuvarlaklık Katsayısı = 4.π.Alan / $(\text{Çevre})^2$ = 1

Yuvarlaklık Katsayısı ≈ 1 → Daha Yuvarlak

Yansıtıcı işaret yakalama

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



Help

File Edit View Go Favorites Desktop Window Help

Image Processing Toolbox ▸ Functions ▸ Image Analysis and Statistics ▸ Pixel Values and Statistics ▸ regionprops

Contents | Search Results

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

0.85 <= Yuvarlaklık Katsayısı >= 1.05

```
ratioLow = 0.85;
ratioUp  = 1.05;
```

Yansıtıcı işaret yakalama

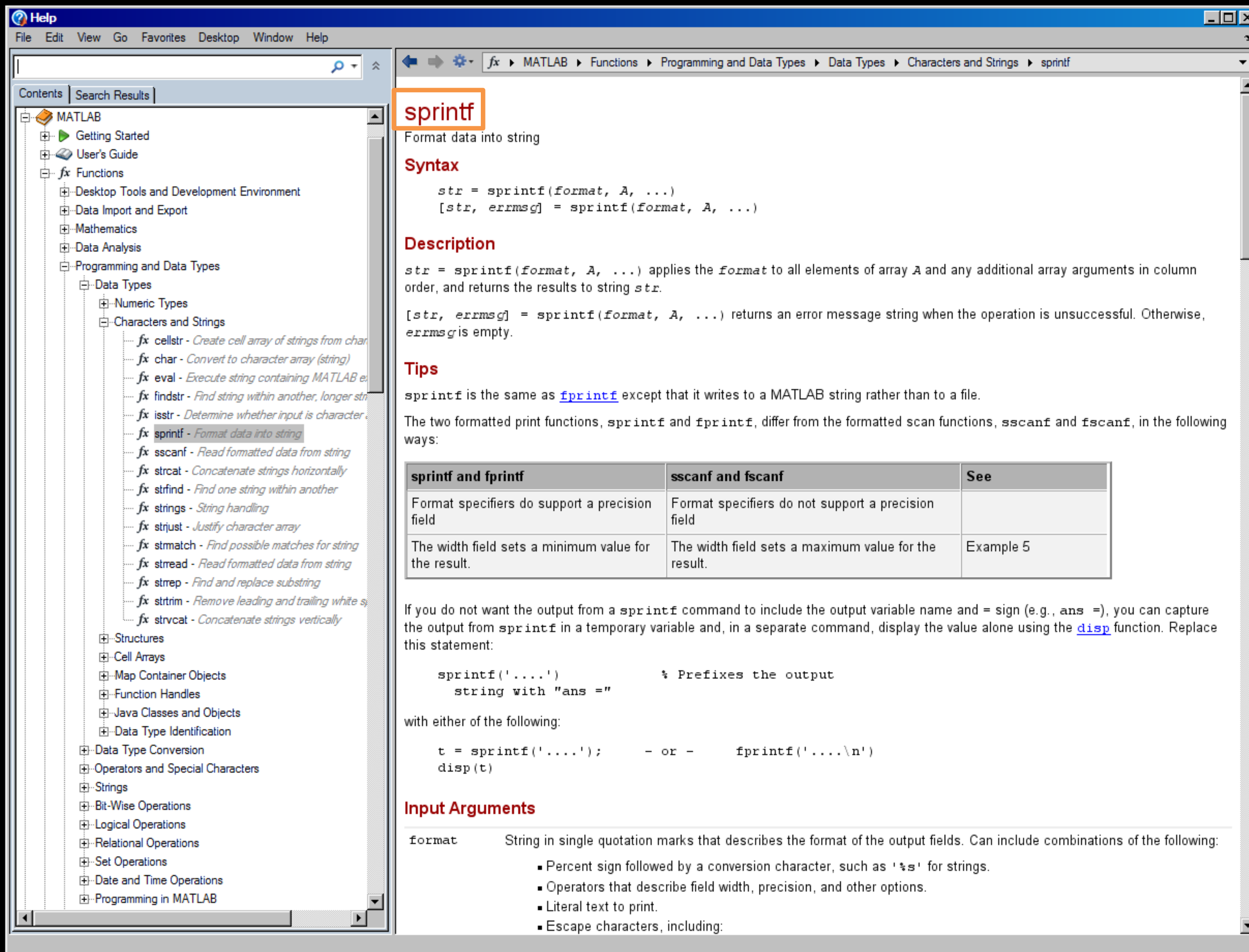
```
for k = 1:length(B)
      % 'k' etiketindeki nesnenin cevresi
      perimeter = stats(k).Perimeter;
      % 'k' etiketindeki nesnenin alanı
      area = stats(k).Area;
      % yuvarlaklık oranını hesapla
      ratio = 4*pi*area / perimeter^2;
      .
      .
      .

end
```

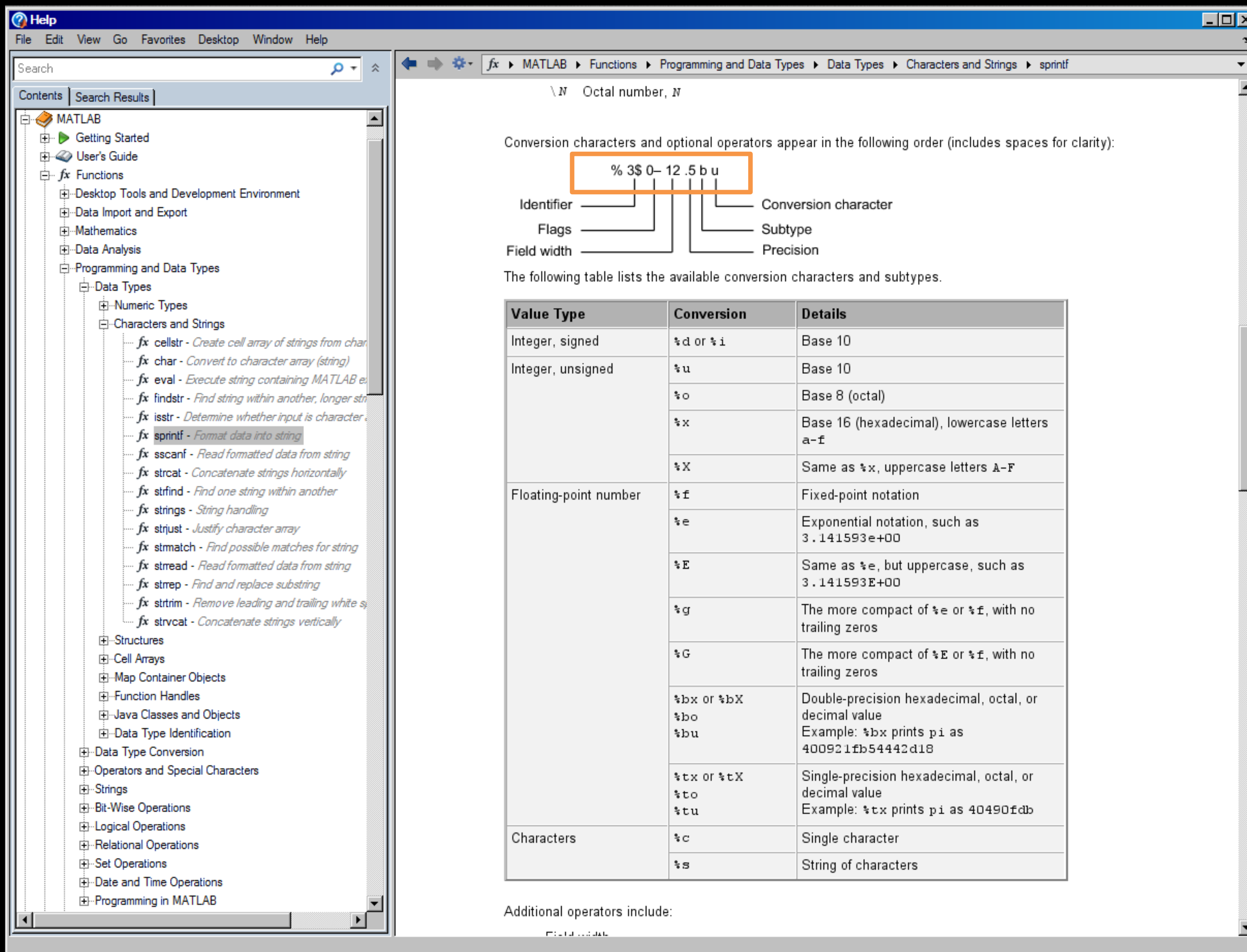
Yansıtıcı işaret yakalama

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

Yansıtıcı işaret yakalama



Yansıtıcı işaret yakalama



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.

Help

File Edit View Go Favorites Desktop Window Help

MATLAB ▸ User's Guide ▸ Programming Fundamentals ▸ Program Components ▸ Operators ▸

Search

Contents | Search Results


```
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:

Help

MATLAB ▸ User's Guide ▸ Programming Fundamentals ▸ Program Components ▸ Operators ▸

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

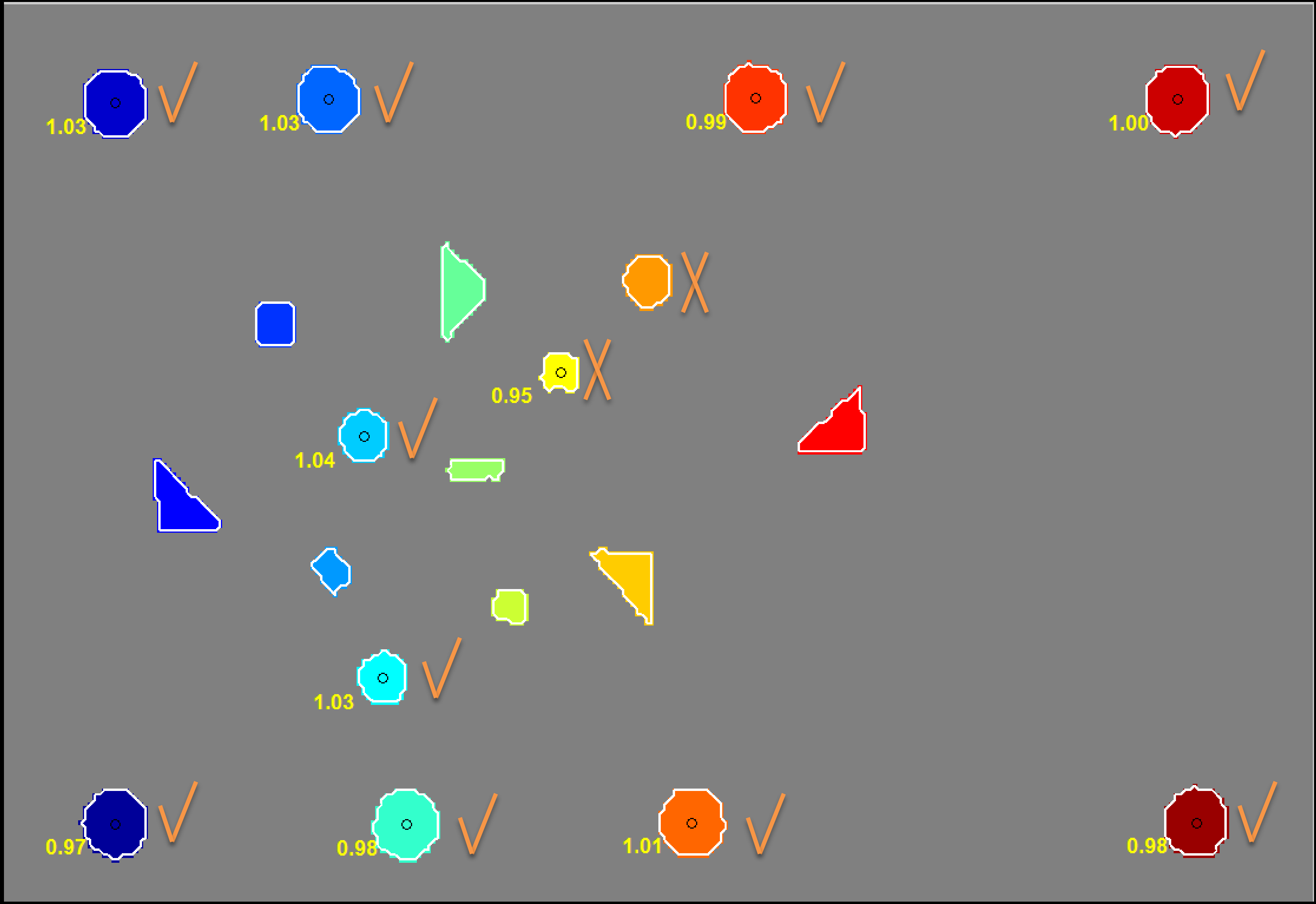
1

7

10

```
for k = 1:length(B)
      .
      .
      .
  % 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
```

!


1.03
1.03
0.99
1.00
0.95
1.04
1.03
0.97
0.98
1.01
0.98

Ev Ödevi

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 of travel composed of small bubbles approximately 0.01

2

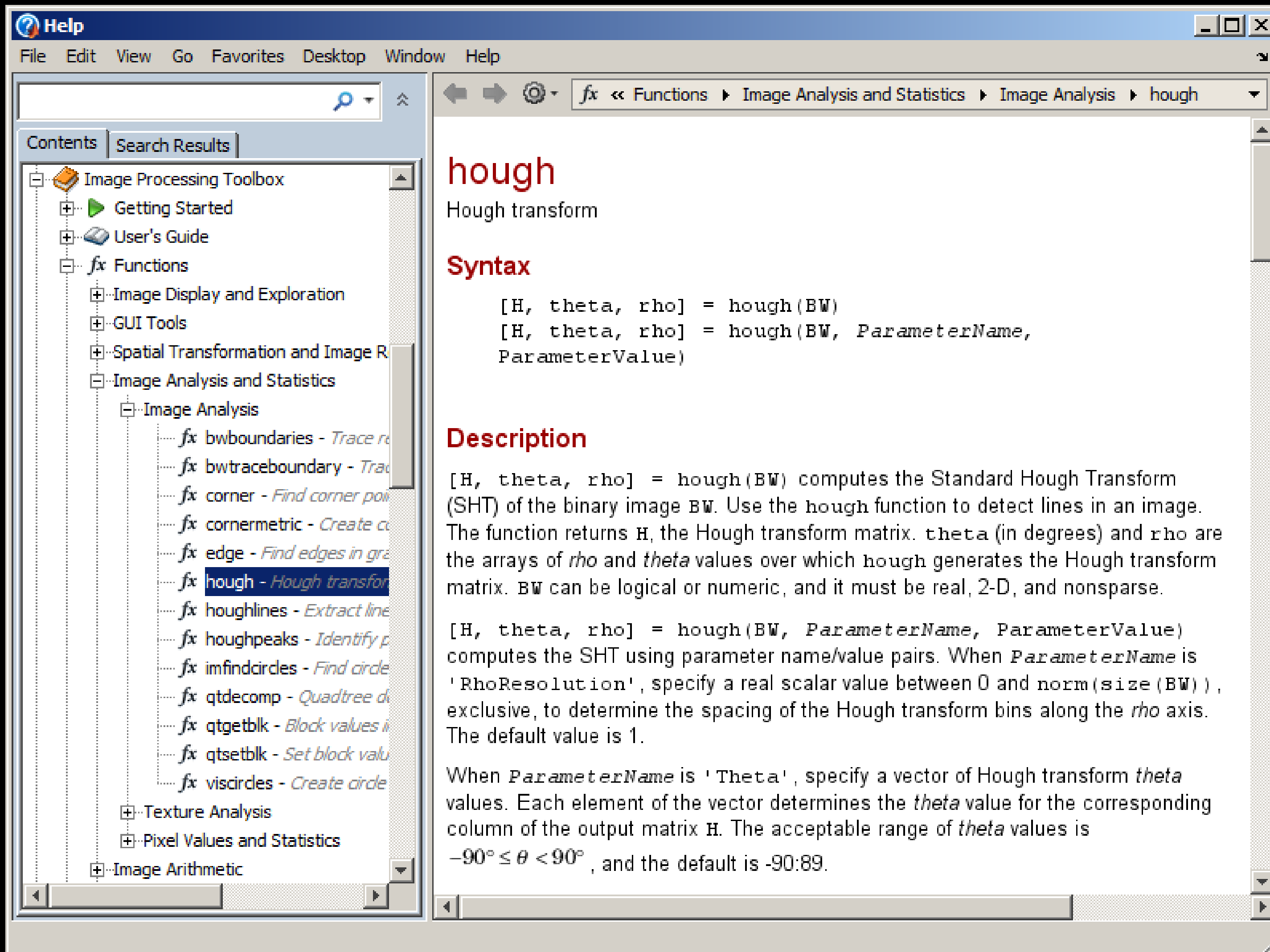
of the point on the line segment from the horizontal midline 109 of the framelet 108.

(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 that the rectangular coordinates of the knots 112 in picture

MATLAB da Standart Hough Transformation



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.

Ev Ödevi #2

1. Geliştirdiğiniz program
2. Ayrıca Hugh dönüşümünü kullanarak yansıtıcı işaretlerin yerini bulmaya çalışınız.

Ödev Teslim Tarihi: 30 Ekim 2019 Çarşamba 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
```