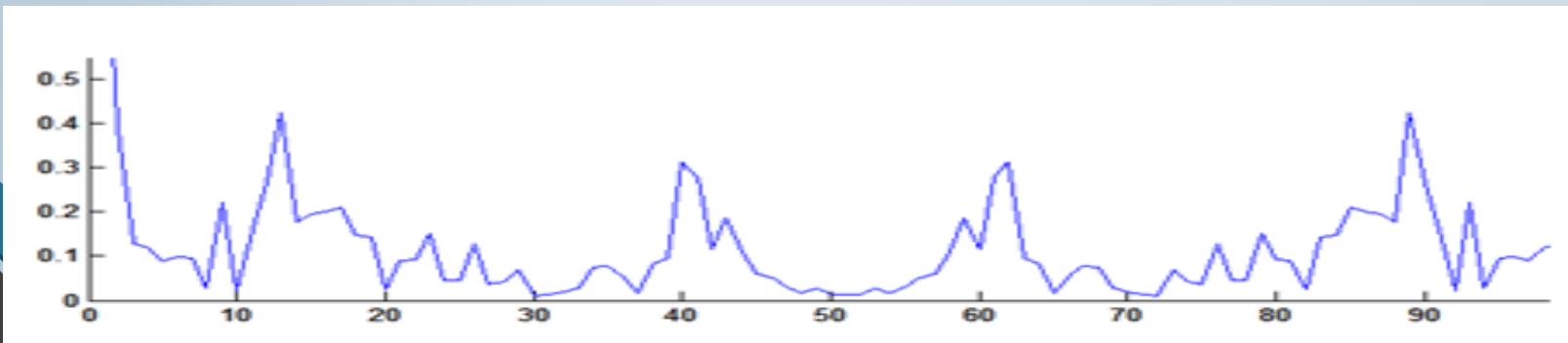
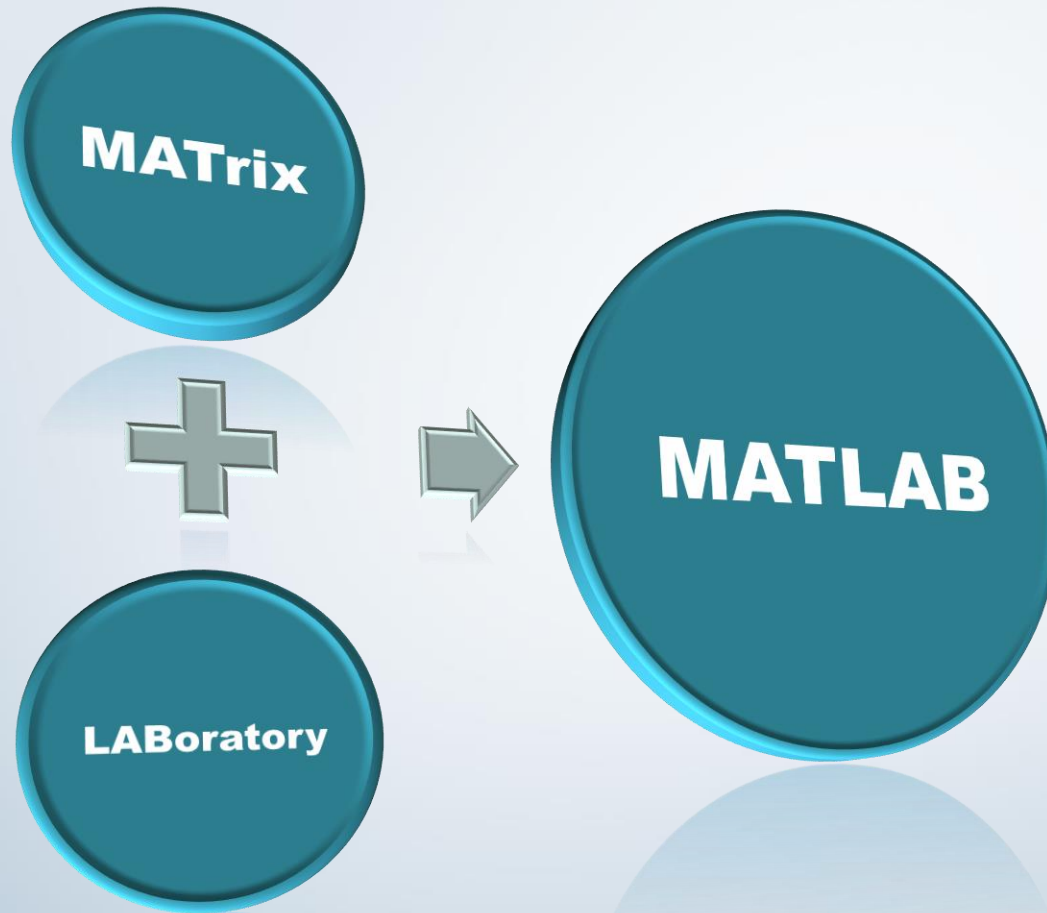


Workshop on Image Processing using **MATLAB**

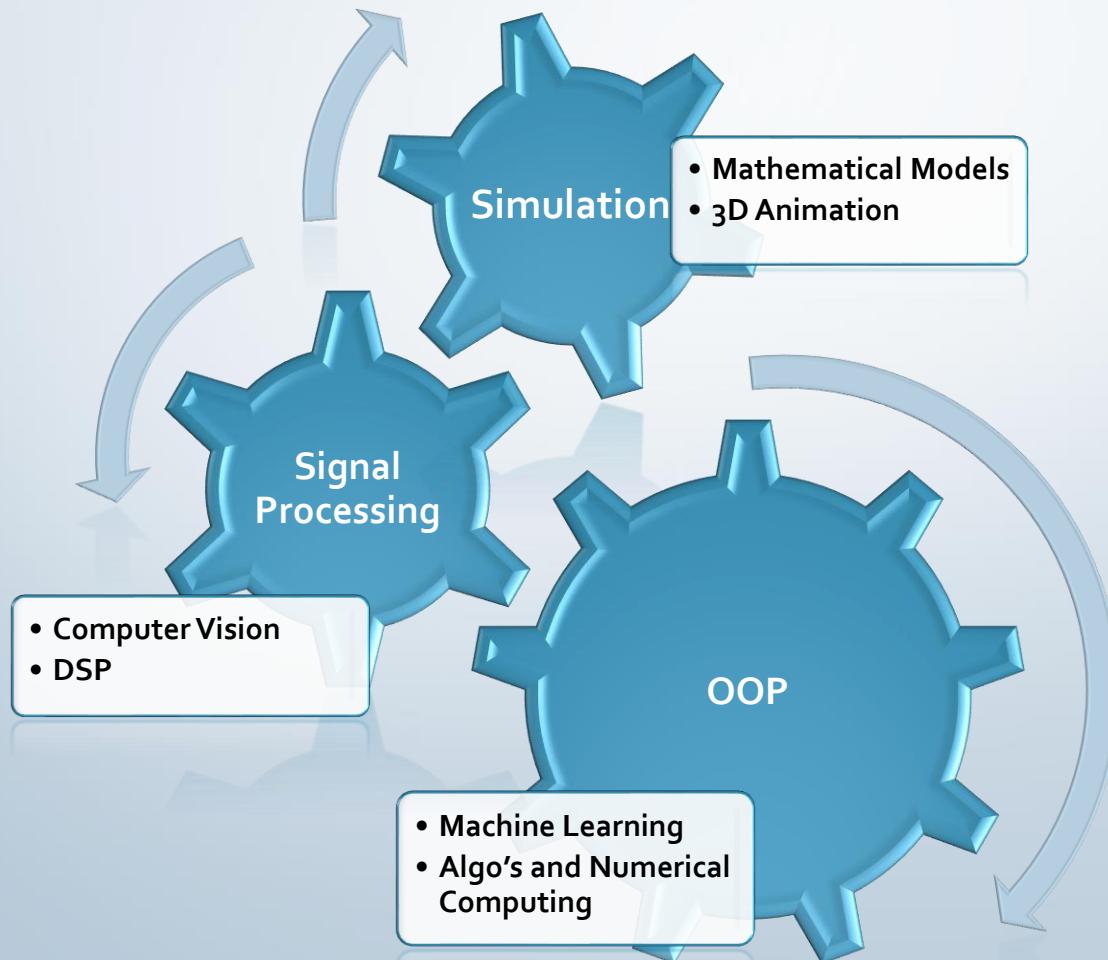




So, Let's have some FUN with Matrices!!

MATLAB

A Fourth Generation Programming Language



Know your MATLAB window

The screenshot displays the MATLAB R2014a environment. The top ribbon includes tabs for HOME, PLOTS, APPS, EDITOR, PUBLISH, and VIEW. The EDITOR tab is active, showing a script file named 'Script.m' with the following code:

```
1 close all;
2 T=4;
3 gap=10;
4 nbits=5;
5 alpha=0.01;
6 t=[-5:1/T:5];
7 filter=src(T,alpha);
8 bits_or=randint(nbits,1);
9 bits=2*bits_or-1;
10 nupscale=1+(nbits-1)*gap*T;
11 upscale=zeros(nupscale,1);
12 upscale(1:gap*T:nupscale)=bits;
13 mod=conv(upscale,filter);
14 stem(upscale);
15 figure;
16 stem(filter);
```

On the left, the 'Current Folder' pane shows the file structure, with a label 'Working Directory' pointing to it. On the right, the 'Workspace' pane lists variables and their values:

Name	Value	Size
alpha	0.0100	1x1 double
ans	[1,3,4,2]	1x4 double
bits	[1;-1;-1;-1;-1]	5x1 logical
bits_or	[1;1;0;0;1]	5x1 logical
check	5x1 logical	5x1 logical
demod	[1;1;0;0;1]	5x1 logical
demod_noise	241x1 double	241x1 double
filter	1x41 double	1x41 double
gap	10	1x1 double
i	5	1x1 double
init	240	1x1 double
mod	201x1 double	201x1 double
nbits	5	1x1 double
noise_sig	201x1 double	201x1 double
nupscale	161	1x1 double
src	2	1x1 double

At the bottom, the 'Command Window' shows the following commands and output:

```
>> plot(bits);
>> check = bits > 0;
>> magic(2)

ans =

     1     3
     4     2
```

A label 'Command History' points to the Command Window. The Windows taskbar at the bottom shows the system clock as 03:46 PM on 28-09-2016.

Basics of MATLAB

Variables

Creating
Arrays

Indexing
Of
Arrays

Operations
On
Arrays

Functions

Control
Flow



Variables

Creating
Arrays

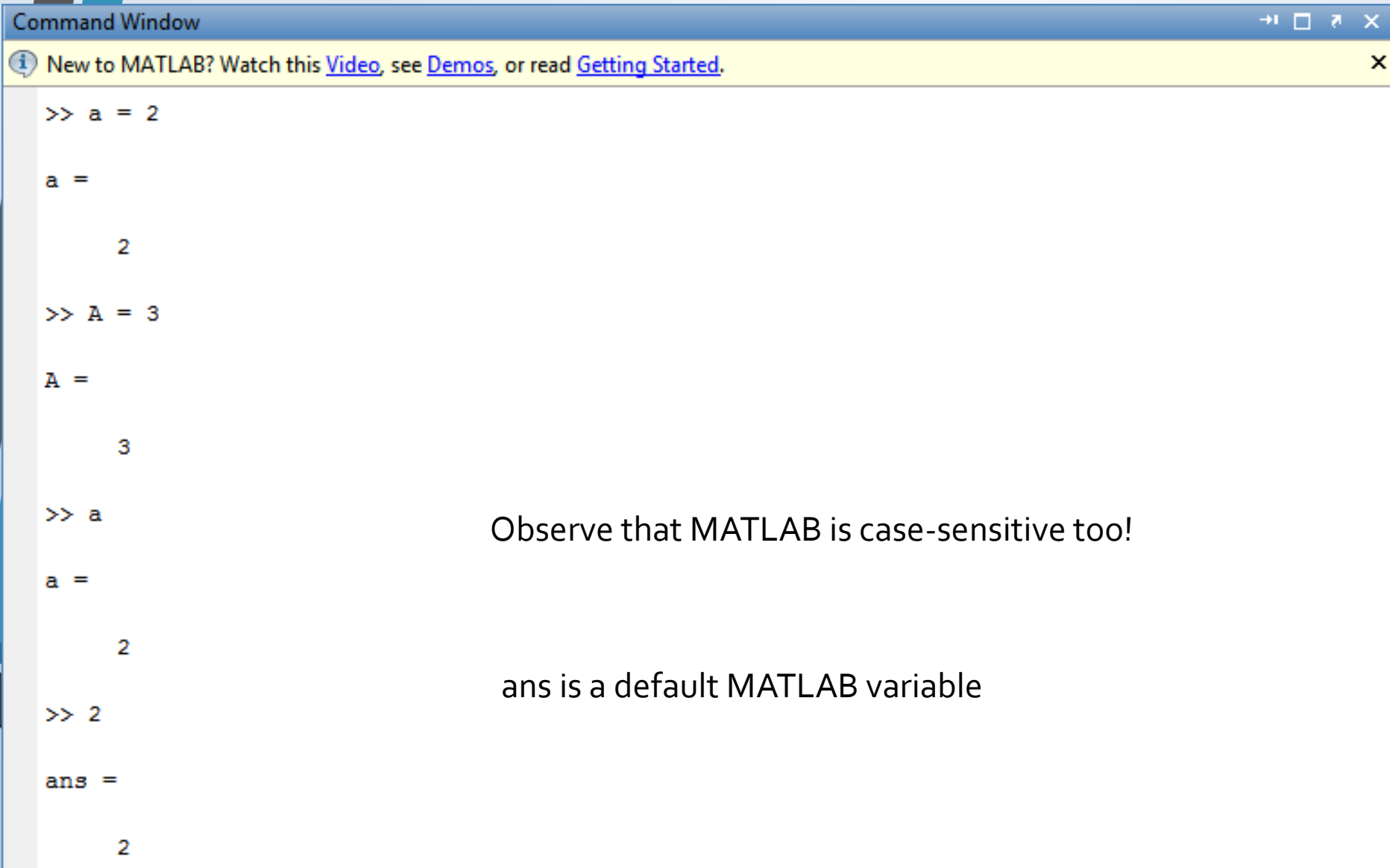
Indexing
Of
Arrays

Operations
On
Arrays

Functions

Control
Flow

Variables

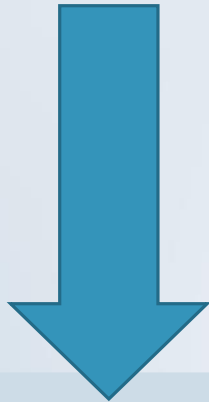


The screenshot shows the MATLAB Command Window interface. At the top, there is a title bar 'Command Window' with standard window controls. Below it is a yellow information banner that reads: 'New to MATLAB? Watch this [Video](#), see [Demos](#), or read [Getting Started](#).' The main area of the window contains the following MATLAB commands and their outputs:

```
>> a = 2  
  
a =  
  
    2  
  
>> A = 3  
  
A =  
  
    3  
  
>> a  
  
a =  
  
    2  
  
>> 2  
  
ans =  
  
    2
```

Observe that MATLAB is case-sensitive too!

ans is a default MATLAB variable



Variables

Creating
Arrays

Indexing
Of
Arrays

Operations
On
Arrays

Functions

Control
Flow

Creating Arrays

Command Window

New to MATLAB? Watch this [Video](#), see [Demos](#), or read [Getting Started](#).

```
>> A = [1,2,3]
```

```
A =
```

```
    1    2    3
```

Commas or spaces! Both will do!

```
>> B = [1 2 3]
```

```
B =
```

```
    1    2    3
```

```
>> C = [1 2 3; 4 5 6]
```

```
C =
```

```
    1    2    3  
    4    5    6
```

Semi colon is used to suppress output

```
>> D = [1 2 3];
```

```
>> D
```

```
D =
```

```
    1    2    3
```



Variables



Creating
Arrays



Indexing
Of
Arrays

Operations
On
Arrays

Functions

Control
Flow

Indexing of Arrays

Command Window

 New to MATLAB? Watch this [Video](#), see [Demos](#), or read [Getting Started](#). 

```
>> D = [1 2 3; 4 5 6; 7 8 9]
```

```
D =
```

1	2	3
4	5	6
7	8	9

Observe that the first element is (1,1) and not (0,0)

```
>> D(2,3)
```

```
ans =
```

```
6
```

```
>> D(1:2, 2)
```

```
ans =
```

```
2  
5
```

The colon operator used for traversal

```
>> D(1:2, 2:3)
```

```
ans =
```

2	3
5	6

Indexing of Arrays

Command Window

New to MATLAB? Watch this [Video](#), see [Demos](#), or read [Getting Started](#).

```
>> D
```

```
D =
```

1	2	3
4	5	6
7	8	9

```
>> D(:, 2)
```

```
ans =
```

2
5
8

```
>> D(2, :)
```

```
ans =
```

4	5	6
---	---	---

```
>> D(:, :)
```

```
ans =
```

1	2	3
4	5	6
7	8	9

How to check size of an array??

What if we use different brackets for indexing ????



Variables



Creating
Arrays



Indexing
Of
Arrays

Operations
On
Arrays

Functions

Control
Flow

Operations on Arrays

Command Window

New to MATLAB? Watch this [Video](#), see [Demos](#), or read [Getting Started](#).

```
>> D + 2
```

```
ans =
```

3	4	5
6	7	8
9	10	11

Matrix + Scalar

```
>> D - 2
```

```
ans =
```

-1	0	1
2	3	4
5	6	7

Matrix - Scalar

```
>> D * 2
```

```
ans =
```

2	4	6
8	10	12
14	16	18

Matrix * Scalar

```
>> D / 2
```

```
ans =
```

0.5000	1.0000	1.5000
2.0000	2.5000	3.0000
3.5000	4.0000	4.5000

Matrix / Scalar

Operations on Arrays

Command Window

```
>> D = [ 5, 10; 15, 20]
```

D =

```
     5     10
    15     20
```

```
>> E = [10,15;30,45]
```

E =

```
    10    15
    30    45
```

```
>> D + E
```

ans =

```
    15    25
    45    65
```

Matrix + Matrix

```
>> D - E
```

ans =

```
    -5    -5
   -15   -25
```

Matrix - Matrix

Command Window

```
>> D - E
```

ans =

```
    -5    -5
   -15   -25
```

```
>> D * E
```

ans =

```
    350    525
    750   1125
```

Product of a Matrix

```
>> D .* E
```

ans =

```
    50    150
   450    900
```

Dot Product of a Matrix

```
>> D ./ E
```

ans =

```
    0.5000    0.6667
    0.5000    0.4444
```

Dot Division of a Matrix



Variables



Creating
Arrays



Indexing
Of
Arrays



Operations
On
Arrays



Functions

Control
Flow

Functions

☐ Why functions?

- ☐ Repeatability

- ☐ Ease of use

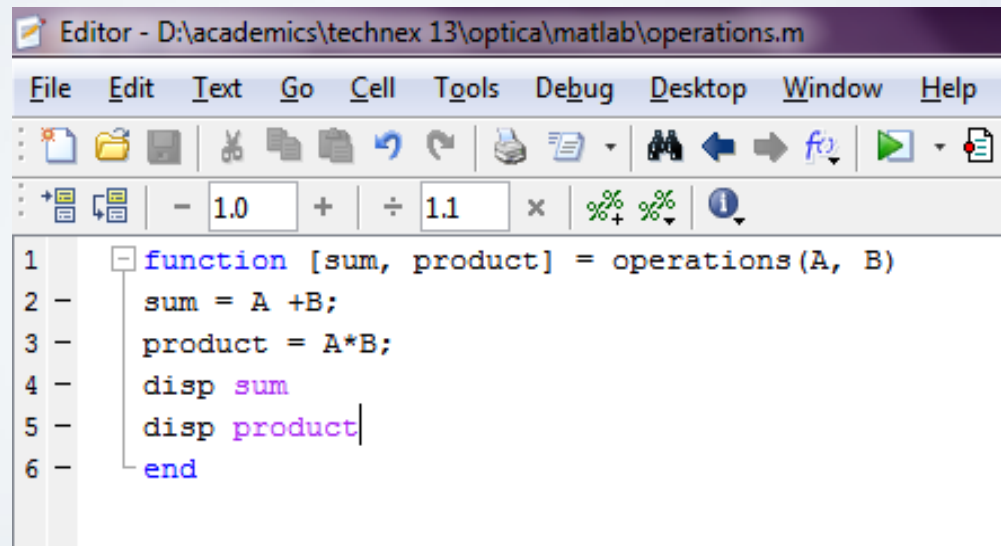
☐ Function header syntax:

function [output variables] = function_name (input variables)

e.g.

function[sum, product] = operations(A, B)

☐ Remember that the function name and the '.m file' name **MUST** be the same.



The screenshot shows a MATLAB editor window titled 'Editor - D:\academics\technex 13\optica\matlab\operations.m'. The window contains a function definition for 'operations'. The code is as follows:

```
1 function [sum, product] = operations(A, B)
2     sum = A + B;
3     product = A*B;
4     disp sum
5     disp product
6 end
```

Functions

Command Window

→ □ ↗ ✕

 New to MATLAB? Watch this [Video](#), see [Demos](#), or read [Getting Started](#). 

```
>> C = D + 2
```

```
C =
```

3	4	5
6	7	8
9	10	11

```
>> [s, p] = operations(C, D)
```

```
s =
```

4	6	8
10	12	14
16	18	20

```
p =
```

54	66	78
90	111	132
126	156	186

```
>> [s, p] = operations(C, D);
```

```
>>
```



Variables



Creating
Arrays



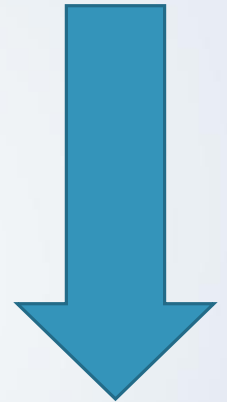
Indexing
Of
Arrays



Operations
On
Arrays



Functions



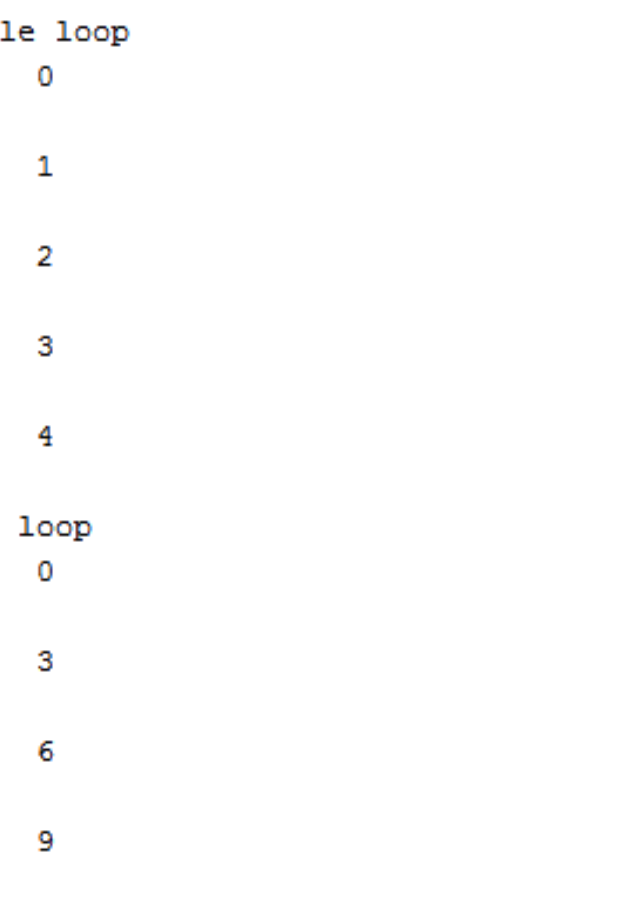
Control
Flow

Control Flow Statements

Editor - D:\academics\technex 13\optica\matlab\control_flow.m

File Edit Text Go Cell Tools Debug Desktop Window Help

1 function [] = control_flow()
 2 i = 0; j = 0;
 3
 4 disp('while loop');
 5 while i < 5
 6 disp(i)
 7 i = i+1;
 8 end
 9
 10 disp('for loop');
 11 for j = 0:3:9
 12 disp(j)
 13 end
 14
 15 disp('if else if statement');
 16 if j == 9
 17 disp('Completely Closed Interval')
 18 elseif j == 8
 19 disp('Semi Closed Interval')
 20 else
 21 disp('j is neither 8 nor 9')
 22 end
 23
 24 end



Command Window

New to MATLAB? Watch this [Video](#), see [Demos](#), or read [Getting Started](#)

```

while loop
    0

    1

    2

    3

    4

for loop
    0

    3

    6

    9

if else if statement
Completely Closed Interval
fx >> |

```



Variables



Creating
Arrays



Indexing
Of
Arrays



Operations
On
Arrays



Functions



Control
Flow

A Few Commands

```
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.

>> who

Your variables are:

A      B      C      D      a      ans     b      c      im      im1     p      s

>> whos

Name      Size      Bytes  Class  Attributes

A          1x3         24  double
B          1x3         24  double
C          3x3        72  double
D          3x3        72  double
a          1x1          8  double
ans        3x3        72  double
b        13x13     1352  double
c          1x5         40  double
im       287x250x3   215250  uint8
im1       75x100x3   22500  uint8
p          3x3         72  double
s          3x3         72  double

>> clear
>> who
>> whos
fx >> |
```

The command `'clc'` clears the screen of the command window
The command `'help'` helps. A LOT.

A Few Functions

- Trigonometric: `sin()`, `sind()`, `cos()`, `cosd()`, etc
- Inverse trigonometric: `asin()`, `asind()`, `acos()`, `acosd()`, etc
- `min()`, `max()`
- `size()`
- `sort()`
- `zeros()`
- `ones()`
- `eye()`

**Finally, Enough of
Basics!**

**Let's have some
Images!! :D**

Images



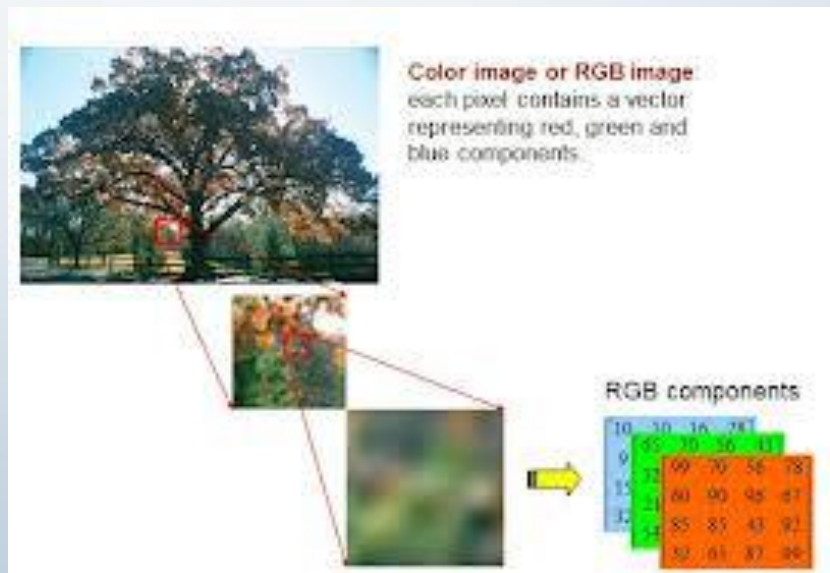
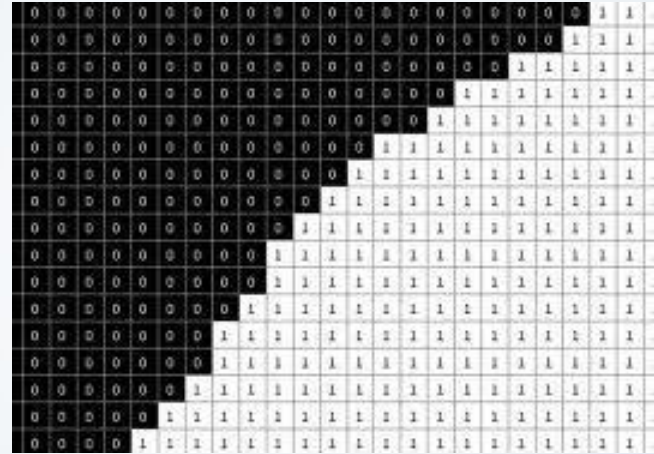
Black
and
White
(Binary)

Gray
scale

Colour

Images in MATLAB

- Each image is seen as a matrix of size equal to the *number of pixel rows x number of pixel columns*
- Each pixel has a value of intensity
- Each element of the matrix contains the value of this intensity at the corresponding to the pixel it represents



Images in MATLAB (Contd.)

Binary

- All the elements of the matrix are either zero or one
- Zero represents black and 1 represents white

Gray scale

- All the elements of the matrix lie between 0 and 255
- Zero represents Black, 255 represents White and the intermediate values represent shades of Gray.

RGB

- Each color has a specific RGB value!
- RGB Images are seen as 3D matrices with the 1st plane corresponding to R, 2nd to G and 3rd to B

Working with Images

Basics

- `imread()`
- `imshow()`
- `imtool()`

Pre-Processing

- `imcrop()`
- `imresize()`

Analysis

- Basic Analysis using Data Cursor
- Conversion to Binary i.e. `im2bw()`
- L-Matrix i.e. `bwlabel()`
- B-Matrix i.e. `bwboundaries()`
- `regionprops()`

Processing

- Morphological Operations
- Contrast Enhancement
- Histogram Equalization
- Image Smoothing
- Blurring

Working with Images

Basics

- `imread()`
- `imshow()`
- `imtool()`



Pre-Processing

- `imcrop()`
- `imresize()`

Analysis

- Basic Analysis using Data Cursor
- Conversion to Binary i.e. `im2bw()`
- L-Matrix i.e. `bwlabel()`
- B-Matrix i.e. `bwboundaries()`
- `regionprops()`

Processing

- Noise Removal / Noise Reduction
- Contrast Enhancement
- Histogram Equalization
- Image Smoothing
- Blurring

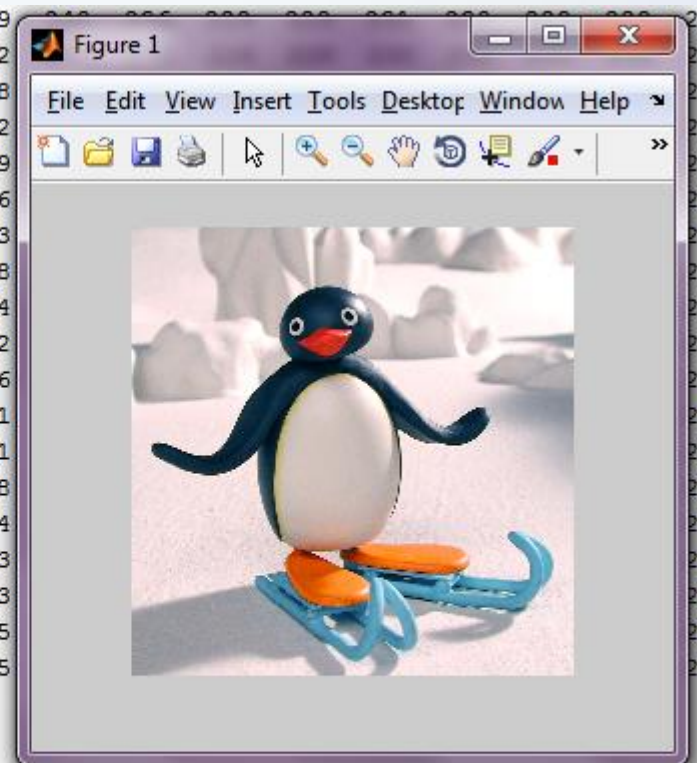
Basics

The input argument: A Matrix
The input argument: File name along with extension

- `imread()`: Reads the image as a matrix
- `imshow()`: Shows the image as an 'image'
- `imtool()`: Same as `imshow` but with different tools

181	184	194	211	229	239
208	201	199	211	231	242
221	218	216	221	230	238
229	229	229	228	229	232
224	226	227	227	226	229
219	219	220	222	224	226
223	220	219	220	223	223
226	223	220	218	218	218
224	223	220	216	213	214
230	230	223	216	217	222
226	226	222	217	220	226
220	220	219	218	223	231
217	217	217	219	224	231
218	218	218	220	224	228
223	221	221	223	223	224
228	224	224	226	224	223
231	227	226	228	226	223
227	227	230	232	229	225
227	227	230	232	229	225

```
>> im = imread('pingu.jpg');  
>> imshow(im)  
Warning: Image is too big to fit on screen; displaying at 33%  
> In imuitools\private\initSize at 73  
   In imshow at 262  
>>
```



Working with Images

Basics

- `imread()`
- `imshow()`
- `imtool()`



Pre-Processing

- `imcrop()`
- `imresize()`



Analysis

- Basic Analysis using Data Cursor
- Conversion to Binary i.e. `im2bw()`
- L-Matrix i.e. `bwlabel()`
- B-Matrix i.e. `bwboundaries()`
- `regionprops()`

Processing

- Morphological Operations
- Contrast Enhancement
- Histogram Equalization
- Image Smoothing
- Blurring

Pre-Processing

- `imcrop()`:

- `imcrop(im)` : Interactive crop tool
- `imcrop(im, [x1,y1,x2,y2])`

- `imresize()`:

- `imresize(im, 0.5)`
- `imresize(im, [200, 200])`
- `imresize(im, [200, NaN])` or `imresize(im, [NaN, 200])`
- `imresize(im, 0.5, 'nearest')` or `imresize(im, 0.5, 'bilinear')` or `imresize(im, 0.5, 'bicubic')`

Working with Images

Basics

- `imread()`
- `imshow()`
- `imtool()`



Pre-Processing

- `imcrop()`
- `imresize()`



Analysis

- Basic Analysis using Data Cursor
- Conversion to Binary i.e. `im2bw()`
- L-Matrix i.e. `bwlabel()`
- B-Matrix i.e. `bwboundaries()`
- `regionprops()`



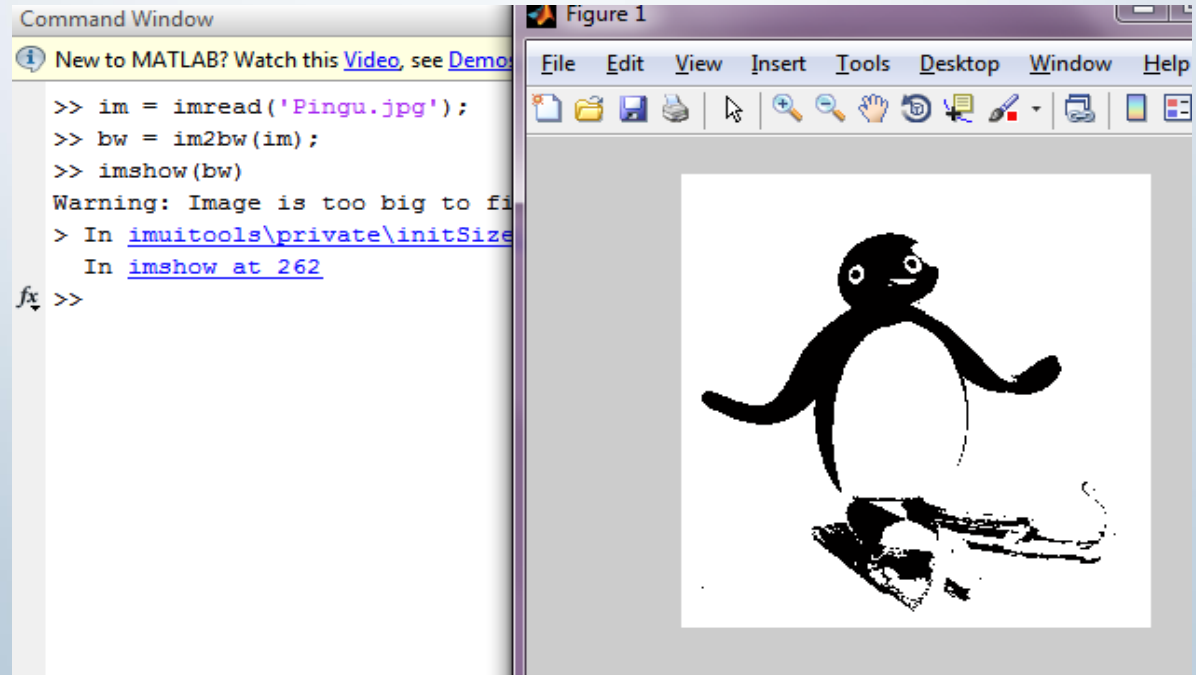
Processing

- Noise Removal/ Noise Reduction
- Contrast Enhancement
- Histogram Equalization
- Image Smoothing
- Blurring

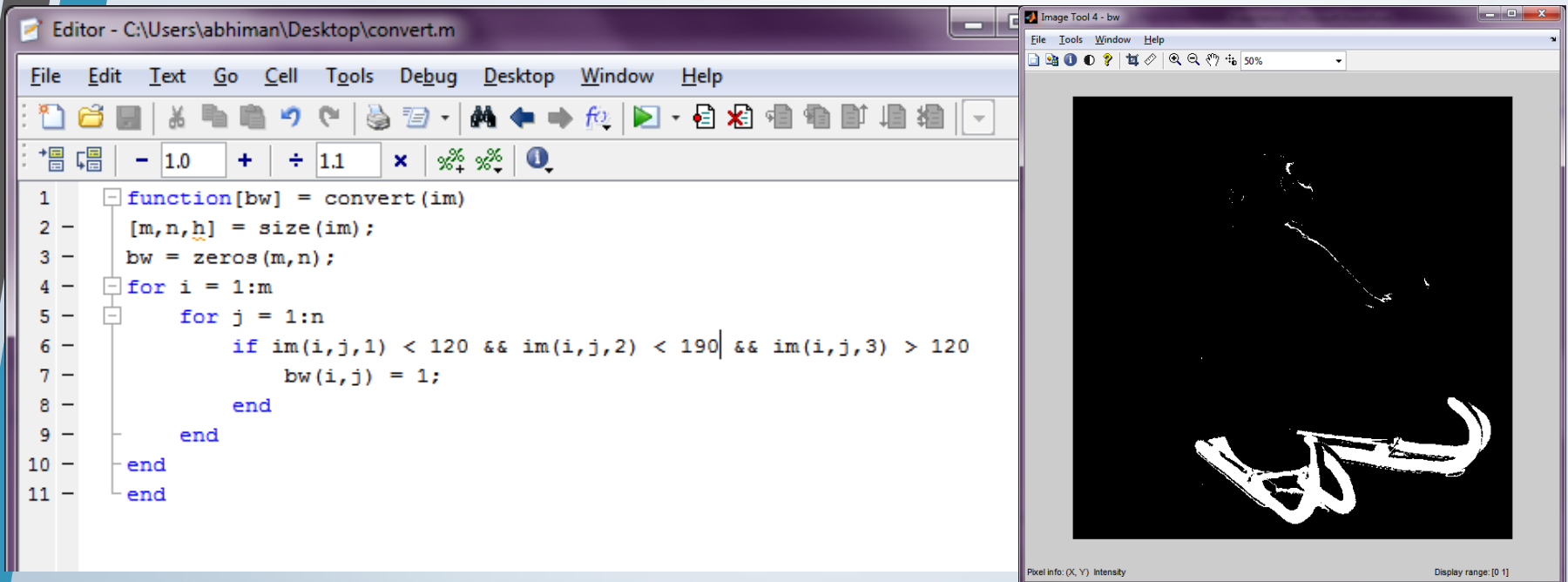
Conversion to Binary (stay tuned for more)

- Why converting?
 - Less information in binary => Computational Ease
 - Can focus on only the area of interest
 - Most of the analysis can be done on Binary only

- `im2bw()`: This function converts a RGB image to a binary image



How to set the threshold ourselves??



Command Window

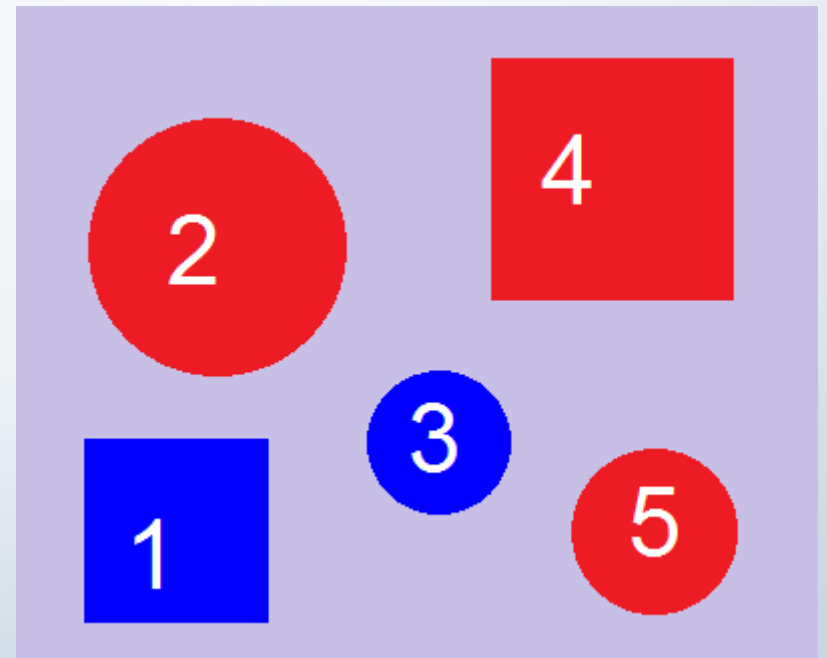
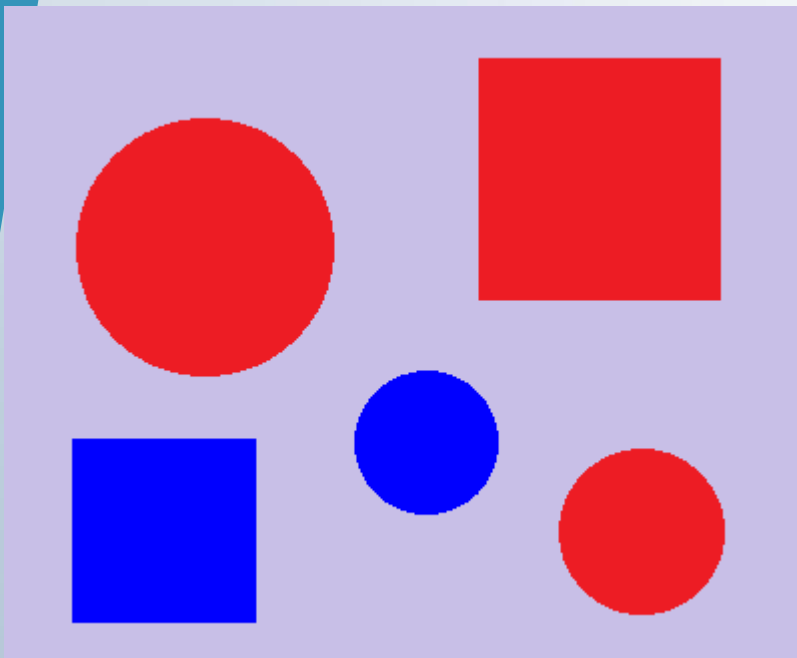
 New to MATLAB? Watch this [Video](#), see [Demos](#), or read [Getting Started](#).

```
>> pingu = imread('Pingu.jpg');
>> skates = convert(pingu);
>> imtool(skates)
fx >> |
```

bwlabel() and L-matrix

- `bwlabel(binary_image)` will return a L-matrix which basically the Label matrix
- L-Matrix is a 2 D matrix of the same size as that of the image.
- The various objects present in an image are given filled with the pixels of the same value, the object number.
- Object number is decided by traversing from left to right and then top to bottom as tie-breaker.
- Each object in the binary image is numbered 1,2,3,... and all the pixels of L corresponding to the objects in binary image have value respectively 1, 2, 3,....
- The background pixels are 0 by default.
- `L = bwlabel(bw);`
`imtool(bw);`

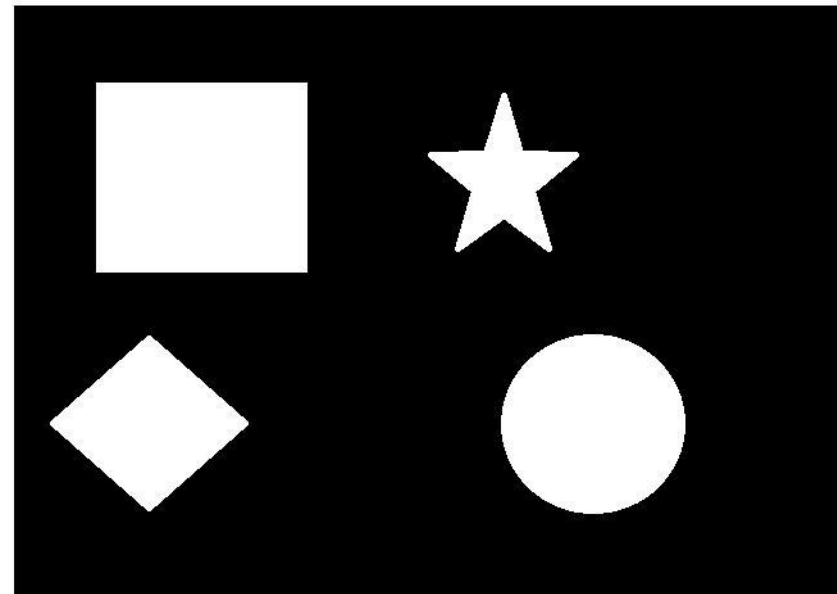
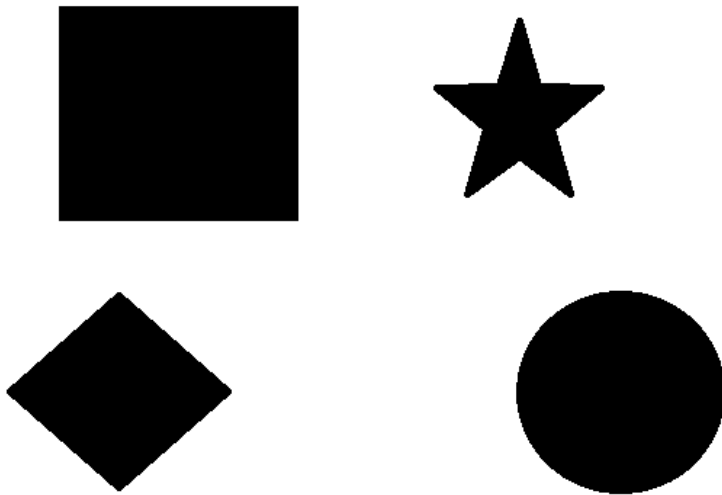
L-matrix labels



Algorithm :BwLabel

- Bwlabel works on the concept of connected regions.
- It starts from the left top most pixel and travels column wise(left to right and top to bottom).
- It keeps a record of all the pixels it has visited and never visits the ones which it has already visited.
- Black(pixel value 0) is considered background and white (pixel value 1) is considered as an object.
- If at all it finds a white pixel,it continues it search in the eight directions from the pixel marking every pixel visited as read.
- Bwlabel(BW,4) searches only in four directions.

Difference and no of objects, area



Do They have the same no of objects?
Background is always black (pixel value
`max(L(:));`
`Sum(sum(L==1));`

regionprops()

- It is used to measure properties of image regions like centroid, perimeter, area, etc.
- Syntax:
`STATS=regionprops(L, properties);`
- `STATS` is a structure array of length equal to the number of labelled objects in `L`.
- `properties` are a comma-separated list of various properties to be measured.

syntax

- `Stats=regionprops(L, 'area', 'perimeter');`
- 1st object properties
- `Stats(1)`
- `Area`
- `Stats(1).Area`

Working with Images

Basics

- `imread()`
- `imshow()`
- `imtool()`



Pre-Processing

- `imcrop()`
- `imresize()`



Analysis

- Basic Analysis using Data Cursor
- Conversion to Binary i.e. `im2bw()`
- L-Matrix i.e. `bwlabel()`
- B-Matrix i.e. `bwboundaries()`
- `regionprops()`



Processing

- Morphological Operations
- Contrast Enhancement
- Image Smoothing
- Image Blurring

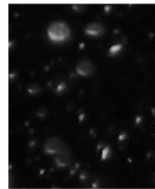


Morphological Operations

- `imerode()`
- `imdilate()`
- `imopen()`
- `imclose()`
- `imfill()`
- `bwmorph()`
- `bwperim()`

Perform a morphological close operation on the image.

Remove the smaller objects
1. Read the image
Ei
I = imread('snowflakes.png');
imshow(I, []);



2. Create a disk-shaped structuring element

se = strel('disk', 5);

3. Remove snowflakes smaller than the structuring element

I_opened = imopen(I, se);
figure, imshow(I_opened, []);



```
closeBW = imclose(originalBW, se);  
figure, imshow(closeBW)
```



Image created in step 2.

Contacts

- Om Sahoo
 - 7607986662
- Mayank Garg
 - 9355514525
- Ankit Mishra
 - 7068563377