



MATLAB & Its Practical Concepts

Outlines For Session-I

- What is Matlab?
- Uses of Matlab
- Environment
- Naming Rule
- Basic Operator
- Variables
- Datatypes
- Data Conversion
- Decision Statements
- Looping Structures
- Vectors, Matrix and Array
- Inbuilt constant
- Function Making

Session-I

What is Matlab?

- The name MATLAB stands for matrix laboratory.
- MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects, which together represent the state-of-the-art in software for matrix computation.
- MATLAB® is a programming platform designed specifically for engineers and scientists to analyze and design systems and products that transform our world. The heart of MATLAB is the MATLAB language, a matrix-based language allowing the most natural expression of computational mathematics.

Continue.

What Can I Do With MATLAB?

- ☐ Analyze data
- ☐ Develop algorithms
- ☐ Create models and applications
- Millions of engineers and scientists worldwide use MATLAB for a range of applications, in industry and academia, including deep learning and machine learning, signal processing and communications, image and video processing, control systems, test and measurement, computational finance, and computational biology.

Uses Of Matlab

- MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation.

Typical uses include:

- Math and computation
- Algorithm development
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including Graphical User Interface building

Environment

- MATLAB window components:

- File Path

- Add Path of Code Folder

Workspace

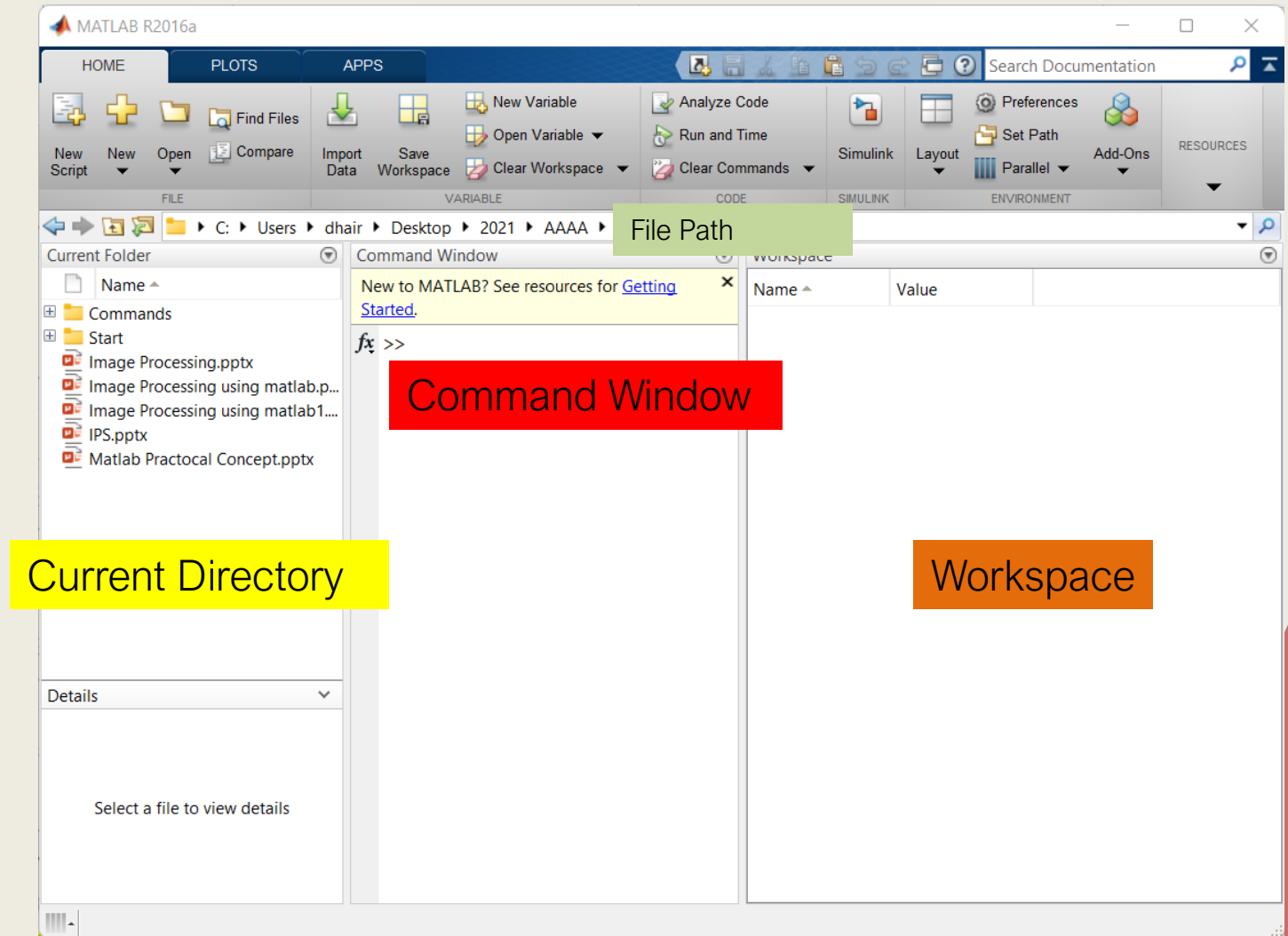
Displays all the defined variables

Command Window

To execute commands in the MATLAB environment

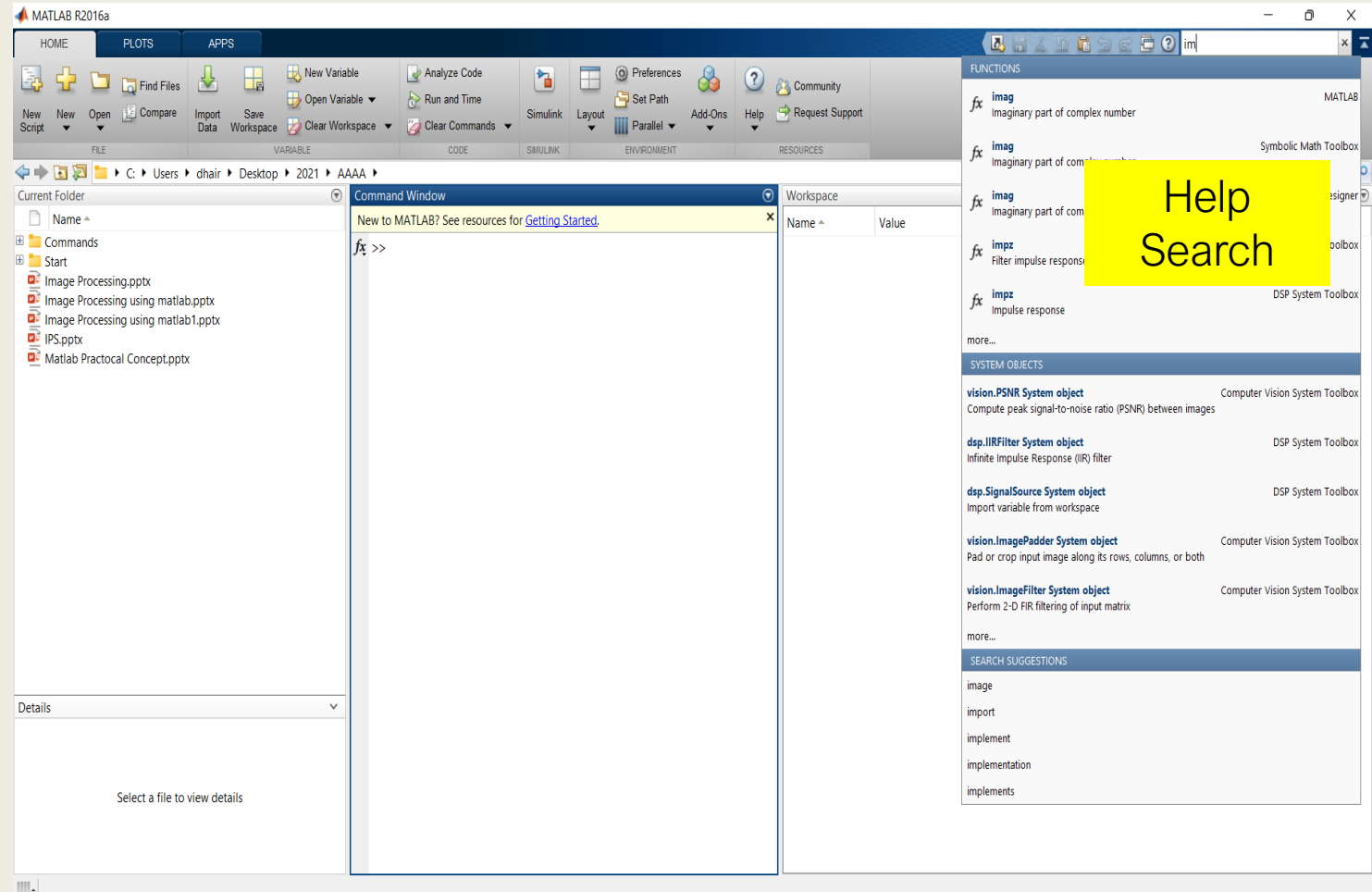
Current Directory

View Folder and m-files



Matlab Help

- MATLAB Help is an extremely powerful assistance to learning MATLAB
- Help not only contains the theoretical background, but also shows demos for implementation
- MATLAB Help can be opened by using the HELP pull-down menu



Naming Rule

Variable naming rules

- must be unique in the first 63 characters
- must begin with a letter
- may not contain blank spaces or other types of punctuation
- may contain any combination of letters, digits, and underscores
- are case-sensitive
- should not use Matlab keyword

Pre-defined variable names

- pi

Basic Operator

Operator	Description
+	Addition or Unary plus
-	Subtraction or unary minus
.*	Scalar Multiplication
./	Scalar Division
.\	Scalar Right Division
:	Colon Operator
.^	Power
.'	Transpose
'	Complex Conjugate Transpose

Practice

$$A=10$$

$$B=20$$

$$C=A+B$$

$$D=A-B$$

$$E=A*B$$

$$F=A/B$$

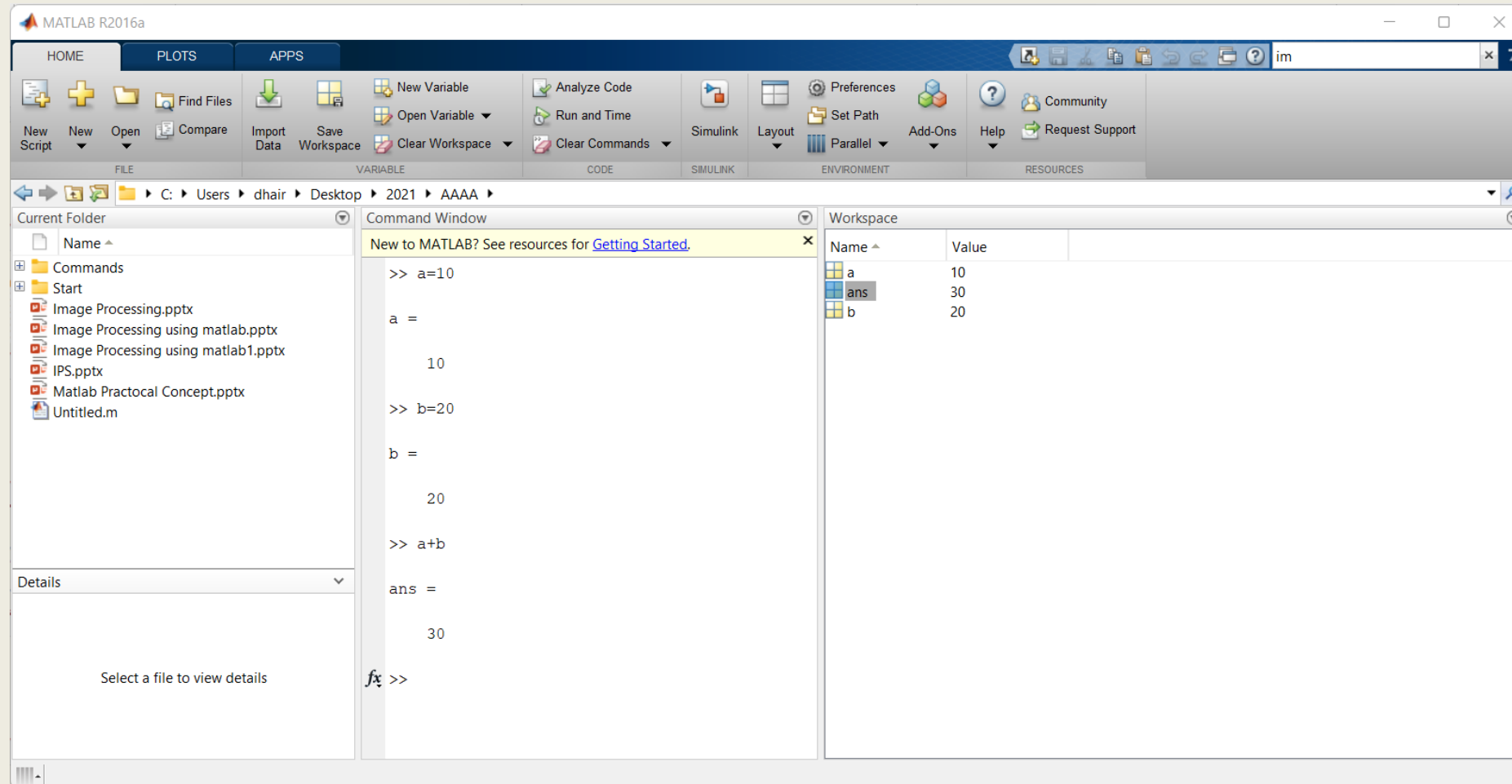
$$G=1:10$$

$$H=10^2$$

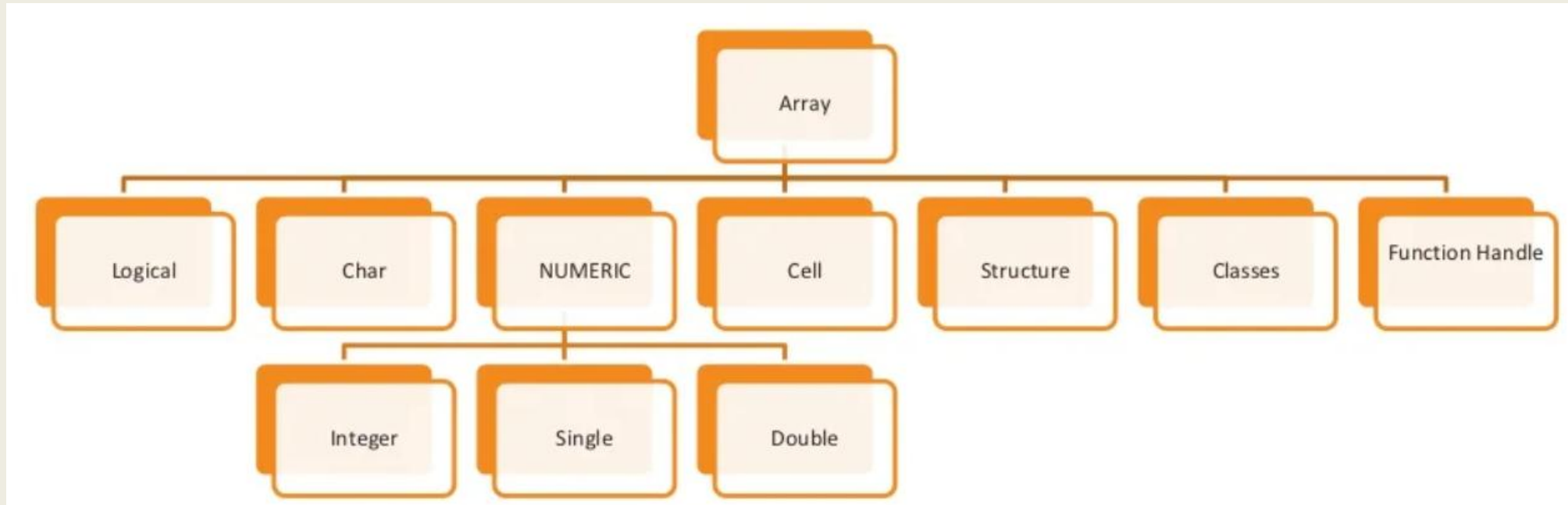
Variables

- Matlab Does not require any type declaration or dimension statement
- Matlab uses conventional decimal notation, scientific notation latter **e** to specify power of ten scale factor.
- Matlab store all numbers internally using the long format specific by the IEEE floating point standard.
- Once a variable is entered into the system, you can refer to it later.
- Variables must have values before they are used.
- When an expression returns a result that is not assigned to any variable, the system assigns it to a variable named **ans**, which can be used later.

Practice



Datatypes



Continue.

Data Type	Range of Values	Conversion Function
Signed 8-bit integer	-2^7 to 2^7-1	int8
Signed 16-bit integer	-2^{15} to $2^{15}-1$	int16
Signed 32-bit integer	-2^{31} to $2^{31}-1$	int32
Signed 64-bit integer	-2^{63} to $2^{63}-1$	int64
Unsigned 8-bit integer	0 to 2^8-1	uint8
Unsigned 16-bit integer	0 to $2^{16}-1$	uint16
Unsigned 32-bit integer	0 to $2^{32}-1$	uint32
Unsigned 64-bit integer	0 to $2^{64}-1$	uint64

Datatype Conversion

Function	Purpose
char	Convert to character array (string)
int2str	Convert integer data to string
mat2str	Convert matrix to string
num2str	Convert number to string
str2double	Convert string to double-precision value
str2num	Convert string to number
native2unicode	Convert numeric bytes to Unicode characters
unicode2native	Convert Unicode characters to numeric bytes
base2dec	Convert base N number string to decimal number
bin2dec	Convert binary number string to decimal number
dec2base	Convert decimal to base N number in string
dec2bin	Convert decimal to binary number in string
dec2hex	Convert decimal to hexadecimal number in string
hex2dec	Convert hexadecimal number string to decimal number
hex2num	Convert hexadecimal number string to double-precision number
num2hex	Convert singles and doubles to IEEE hexadecimal strings
cell2mat	Convert cell array to numeric array
cell2struct	Convert cell array to structure array
cellstr	Create cell array of strings from character array
mat2cell	Convert array to cell array with potentially different sized cells
num2cell	Convert array to cell array with consistently sized cells
struct2cell	Convert structure to cell array

Practice

```
%Convert the array to character
A = [77 65 84 76 65 66];
C = char(A)
%Convert the array to string
B = ['Past','Present','Future']
D = cellstr(B)
%Convert the value int to string
chr = int2str(256)
%Convert the value matrix to string
chr1 = mat2str([3.85 2.91; 7.74 8.99])
%Convert the value number to string
S = num2str(pi)
%Convert the value string to double
X = str2num('100')
%Convert the value string to double
Y = str2double('3.1416')
%Convert the value base to decimal
baseStr = '1B';
D1 = base2dec(baseStr,12)
%Convert the value to a decimal to base number.
D2 = 23;
baseStr = dec2base(D2,12)
%Convert a character vector that represents a hexadecimal value to a decimal number.
hexStr = '3FF';
D3 = hex2dec(hexStr)
```

Determine Data Type

Function	Purpose
is	Detect state
isa	Determine if input is object of specified class
iscell	Determine whether input is cell array
iscellstr	Determine whether input is cell array of strings
ischar	Determine whether item is character array
isfield	Determine whether input is structure array field
isfloat	Determine if input is floating-point array
ishghandle	True for Handle Graphics object handles
isinteger	Determine if input is integer array
isjava	Determine if input is Java object
islogical	Determine if input is logical array
isnumeric	Determine if input is numeric array
isobject	Determine if input is MATLAB object
isreal	Check if input is real array
isscalar	Determine whether input is scalar
isstr	Determine whether input is character array
isstruct	Determine whether input is structure array
isvector	Determine whether input is vector
class	Determine class of object
validateattributes	Check validity of array
whos	List variables in workspace, with sizes and types

Practice

```
x = 3
isinteger(x)
isfloat(x)
isvector(x)
isscalar(x)
isnumeric(x)
```

```
x = 23.54
isinteger(x)
isfloat(x)
isvector(x)
isscalar(x)
isnumeric(x)
```

```
x = [1 2 3]
isinteger(x)
isfloat(x)
isvector(x)
isscalar(x)
```

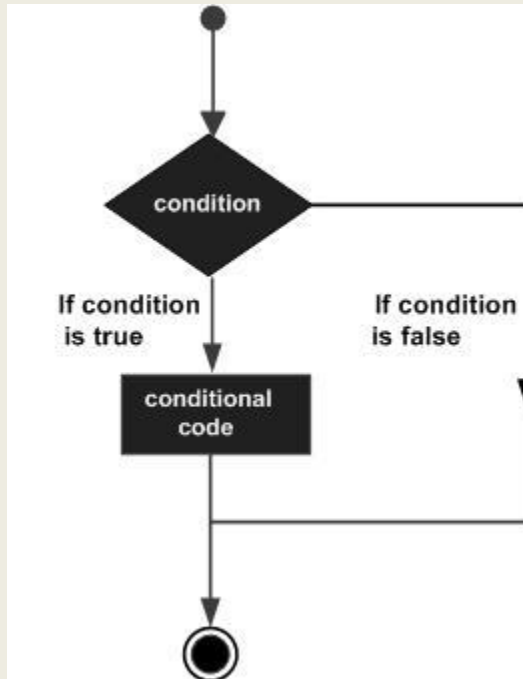
```
x = 'Hello'
isinteger(x)
isfloat(x)
isvector(x)
isscalar(x)
isnumeric(x)
```

Initial Command

- To clear Window – clc
- To closed all fig- close all
- To clear work space area- clear all

Decision Statements

- Decision making structures require that the programmer should specify one or more conditions to be evaluated or tested by the program, along with a statement or statements to be executed if the condition is determined to be true, and optionally, other statements to be executed if the condition is determined to be false.



If-end

Create a script file and type the following code –

```
a = 10;
% check the condition using if statement
    if a < 20
        % if condition is true then print the
following
        fprintf('a is less than 20\n' );
    end
fprintf('value of a is : %d\n', a);
```

When you run the file, it displays the following result –

```
a is less than 20
value of a is : 10
```

If-else-end

- Create a script file and type the following code –

```
a = 100;  
% check the boolean condition  
if a < 20  
    % if condition is true then print the following  
    fprintf('a is less than 20\n' );  
else  
    % if condition is false then print the following  
    fprintf('a is not less than 20\n' );  
end  
fprintf('value of a is : %d\n', a);
```

- When the above code is compiled and executed, it produces the following result –

```
a is not less than 20  
value of a is : 100
```

If-elseif-elseif-end

- Create a script file and type the following code in it –

```
a = 100;
%check the boolean condition
if a == 10
    % if condition is true then print the following
    fprintf('Value of a is 10\n' );
elseif( a == 20 )
    % if else if condition is true
    fprintf('Value of a is 20\n' );
elseif a == 30
    % if else if condition is true
    fprintf('Value of a is 30\n' );
else
    % if none of the conditions is true '
    fprintf('None of the values are matching\n');
    fprintf('Exact value of a is: %d\n', a );
end
```

- When the above code is compiled and executed, it produces the following result–

```
None of the values are matching
Exact value of a is: 100
```


Nested if

- Create a script file and type the following code in it –

```
a = 100;
b = 200;
% check the boolean condition
if( a == 100 )

    % if condition is true then check the following
    if( b == 200 )

        % if condition is true then print the following
        fprintf('Value of a is 100 and b is 200\n' );
    end
end

end
fprintf('Exact value of a is : %d\n', a );
fprintf('Exact value of b is : %d\n', b );
```

- When you run the file, it displays –

```
Value of a is 100 and b is 200
Exact value of a is : 100
Exact value of b is : 200
```

Switch

```
grade = 'B';
switch(grade)
case 'A'
    fprintf('Excellent!\n' );
case 'B'
    fprintf('Well done\n' );
case 'C'
    fprintf('Well done\n' );
case 'D'
    fprintf('You passed\n' );
case 'F'
    fprintf('Better try again\n' );
otherwise
    fprintf('Invalid grade\n' );
end
```

- When you run the file, it displays –

Well done

Nested Switch

```
a = 100;
b = 200;
switch(a)
    case 100
        fprintf('This is part of outer switch %d\n', a );
        switch(b)
            case 200
                fprintf('This is part of inner switch %d\n', a );
            end
        end
end

fprintf('Exact value of a is : %d\n', a );
fprintf('Exact value of b is : %d\n', b );
```

- When you run the file, it displays –
- This is part of outer switch 100
- This is part of inner switch 100
- Exact value of a is : 100
- Exact value of b is : 200
-

Looping Structures

```
for a = 10:20
    fprintf('value of a: %d\n', a);
end
```

- When you run the file, it displays the following result –

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
value of a: 20
```

Interval loop

```
for a = 1.0: -0.1: 0.0  
    disp(a)  
end
```

- When you run the file, it displays the following result –

```
1  
0.90000  
0.80000  
0.70000  
0.60000  
0.50000  
0.40000  
0.30000  
0.20000  
0.10000  
0
```

Custom Loop

```
for a = [24,18,17,23,28]  
    disp(a)  
end
```

- When you run the file, it displays the following result –

24

18

17

23

28

Array

- All variables of all data types in MATLAB are multidimensional arrays. A vector is a one-dimensional array and a matrix is a two-dimensional array.
- We have already discussed vectors and matrices. In this chapter, we will discuss multidimensional arrays. However, before that, let us discuss some special types of arrays.

Special Arrays in MATLAB

```
zeros(5)
```

- MATLAB will execute the above statement and return the following result –

```
ans =
```

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Continue.

```
ones (4, 3)
```

- MATLAB will execute the above statement and return the following result –

```
ans =
```

1	1	1
1	1	1
1	1	1
1	1	1

Continue.

```
eye (4)
```

- MATLAB will execute the above statement and return the following result –

```
ans =
```

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

Continue

```
rand(3, 5)
```

- MATLAB will execute the above statement and return the following result –

- `ans =`

0.8147	0.9134	0.2785	0.9649	0.9572
0.9058	0.6324	0.5469	0.1576	0.4854
0.1270	0.0975	0.9575	0.9706	0.8003

A Magic Square

- A magic square is a square that produces the same sum, when its elements are added row-wise, column-wise or diagonally.

`magic(4)`

- MATLAB will execute the above statement and return the following result –

- `ans =`

16	2	3	13
5	11	10	8
9	7	6	12
4	14	15	1

Multidimensional Arrays

- An array having more than two dimensions is called a multidimensional array in MATLAB. Multidimensional arrays in MATLAB are an extension of the normal two-dimensional matrix.

```
a = [7 9 5; 6 1 9; 4 3 2]
```

- MATLAB will execute the above statement and return the following result –

- a =

7	9	5
6	1	9
4	3	2

Inbuilt Command

Function	Purpose
length	Length of vector or largest array dimension
ndims	Number of array dimensions
size	Array dimensions
diag	Diagonal matrices and diagonals of matrix
flipdim	Flips array along specified dimension
fliplr	Flips matrix from left to right
flipud	Flips matrix up to down
repmat	Replicates and tile array
reshape	Reshapes array
rot90	Rotates matrix 90 degrees
sort	Sorts array elements in ascending or descending order
sortrows	Sorts rows in ascending order
transpose	Transpose

Function Making

- Both scripts and functions allow you to reuse sequences of commands by storing them in program files. Scripts are the simplest type of program, since they store commands exactly as you would type them at the command line. Functions provide more flexibility, primarily because you can pass input values and return output values. For example, this function named `fact` computes the factorial of a number (`n`) and returns the result (`f`).

```
function f = fact(n)
```

```
    f = prod(1:n);
```

```
end
```

Primary and Sub-Functions

```
function [x1,x2] = quadratic(a,b,c)

%this function returns the roots of
% a quadratic equation.
% It takes 3 input arguments
% which are the co-efficients of x2, x and the
%constant term
% It returns the roots

d = disc(a,b,c);

x1 = (-b + d) / (2*a);

x2 = (-b - d) / (2*a);

end    % end of quadratic
```

```
function dis = disc(a,b,c)

%function calculates the discriminant

dis = sqrt(b^2 - 4*a*c);

end    % end of sub-function
```


Nested Function

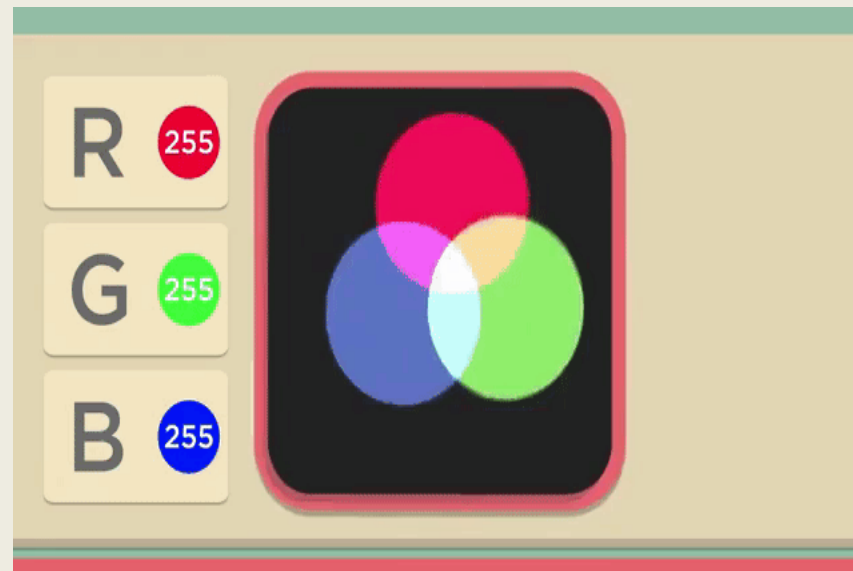
```
function [x1,x2] = quadratic2(a,b,c)
function disc % nested function
d = sqrt(b^2 - 4*a*c);
end % end of function disc
disc;
x1 = (-b + d) / (2*a);
x2 = (-b - d) / (2*a);
end % end of function quadratic2
```

- You can call the above function from command prompt as ?
- `quadratic2(2,4,-4)`

Outlines For Session-II

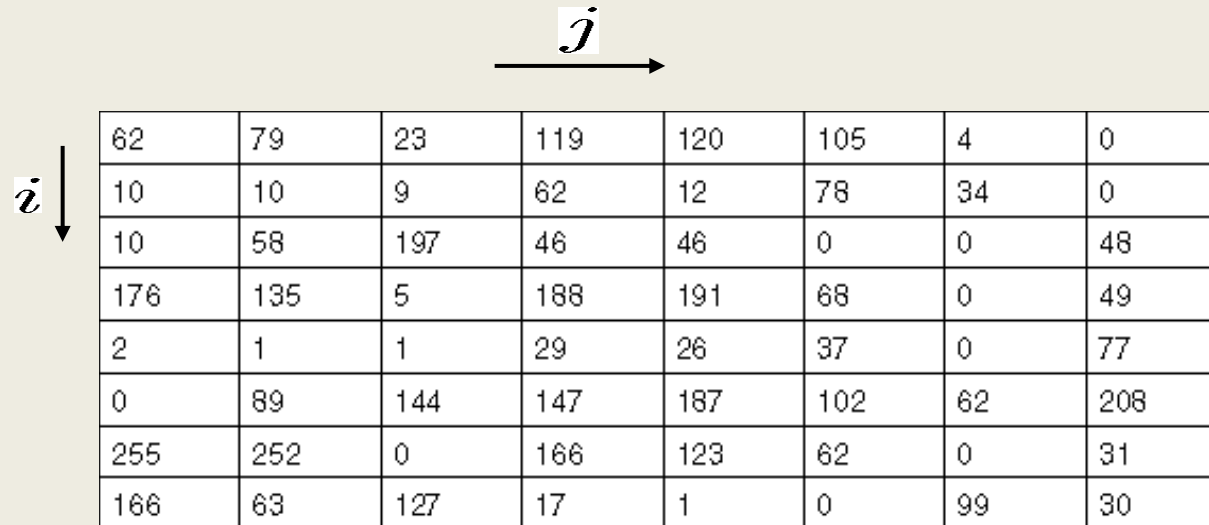
- What is Image?
- Types of Image
- Basic Function
- Arithmetical Operation
- Color Conversion Model
- Histogram Equalization
- Thresholding Function
- Filtering Function
- Morphological Function
- Edge Detection
- Future Research Direction

Session-II



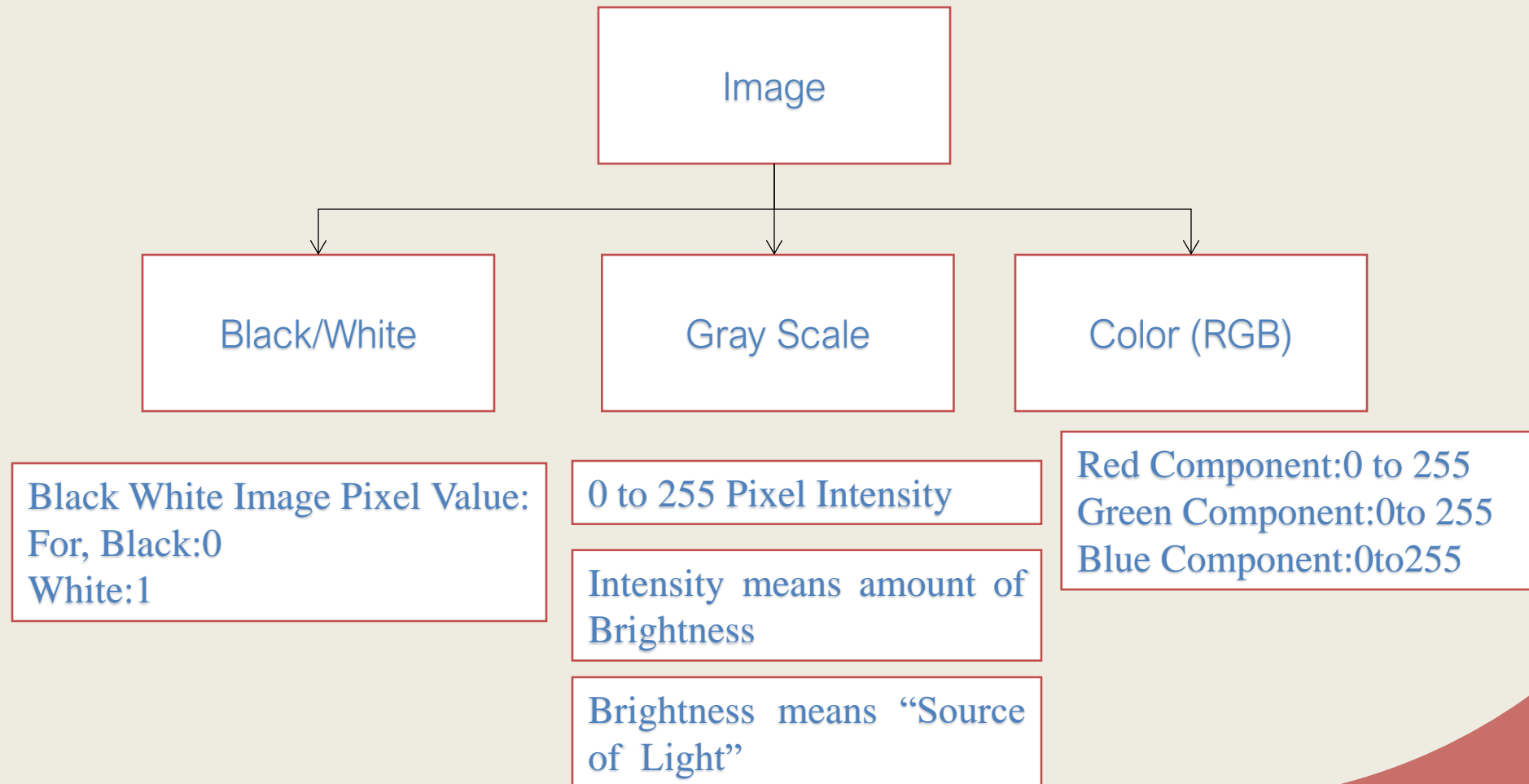
What is Image?

- A Image is a Set of Pixel(Picture Elements) .
- We can think of an **image** as a function, f , from \mathbb{R}^2 to \mathbb{R} :
 - $f(x, y)$ gives the **intensity** at position (x, y)
- The image can now be represented as a matrix of integer values

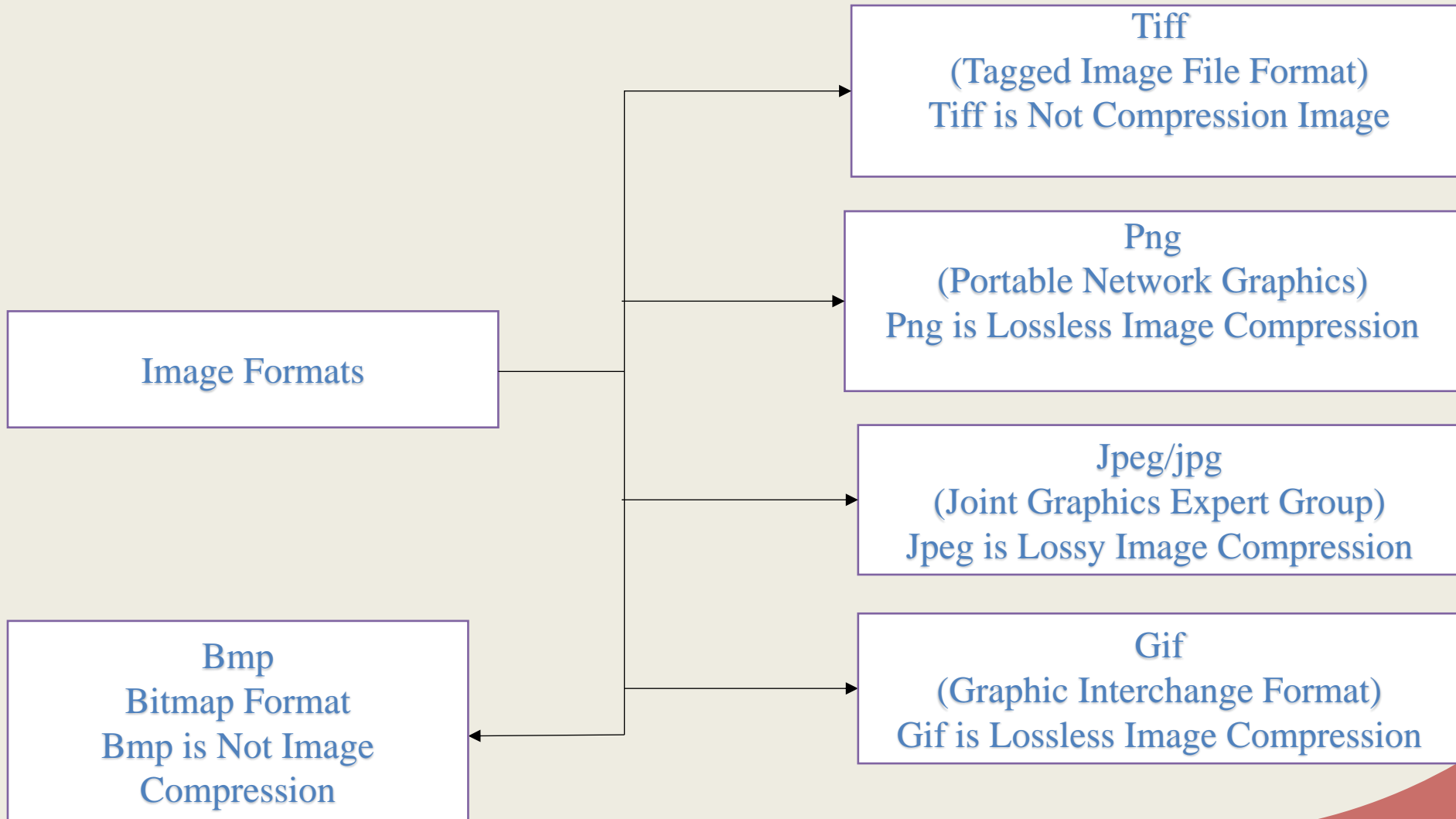


62	79	23	119	120	105	4	0
10	10	9	62	12	78	34	0
10	58	197	46	46	0	0	48
176	135	5	188	191	68	0	49
2	1	1	29	26	37	0	77
0	89	144	147	187	102	62	208
255	252	0	166	123	62	0	31
166	63	127	17	1	0	99	30

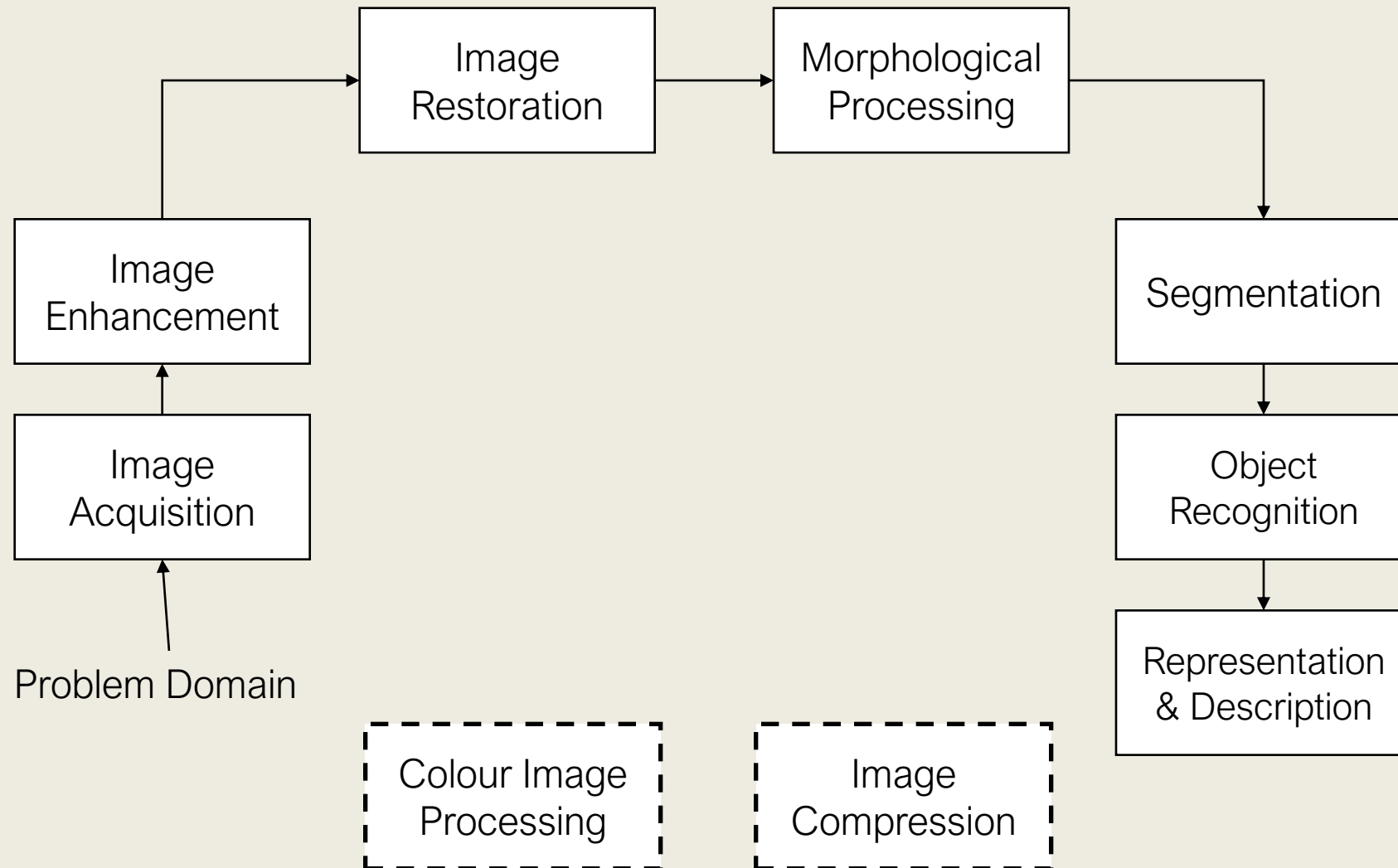
Types of Image (2D)



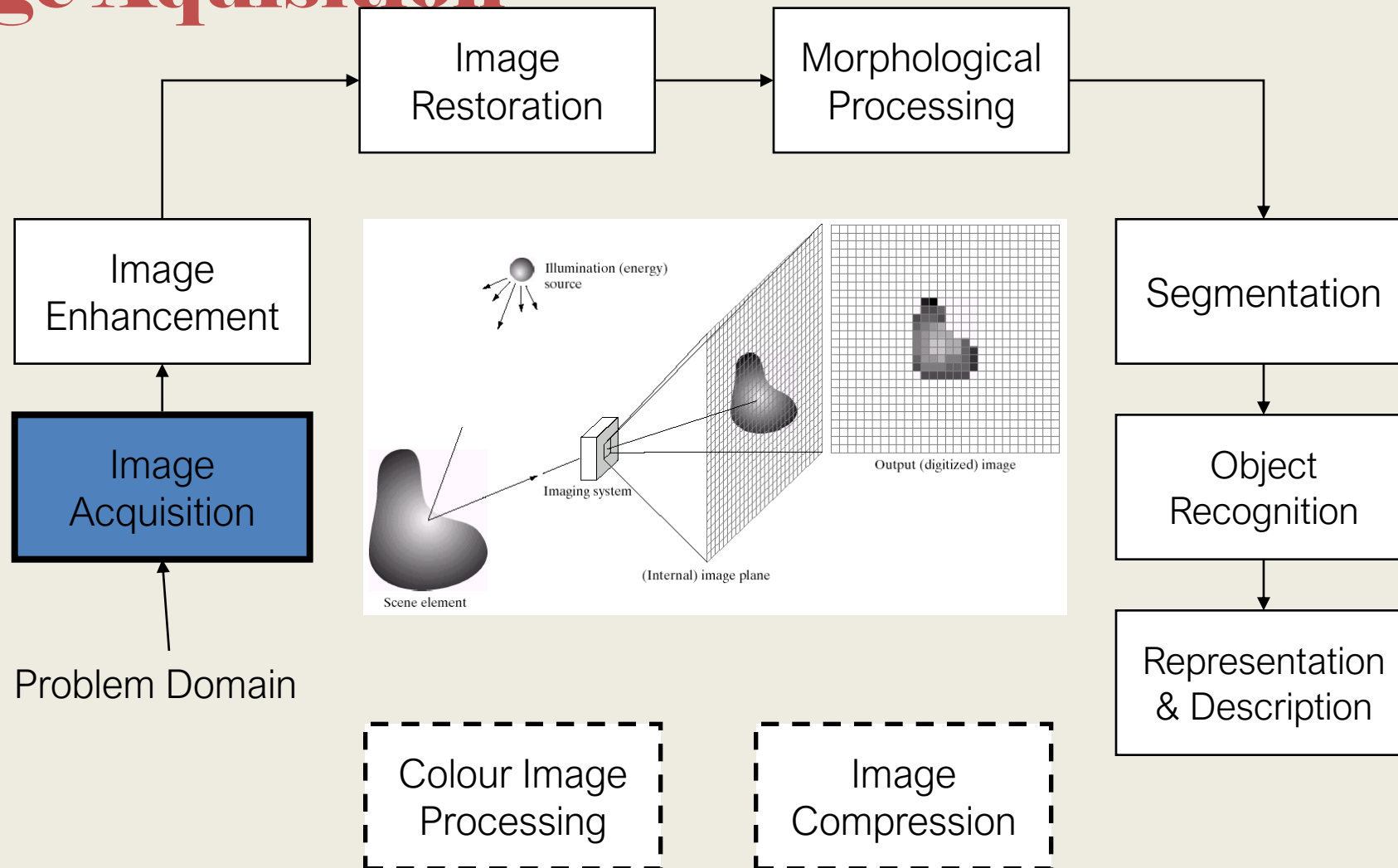
Formats of Image



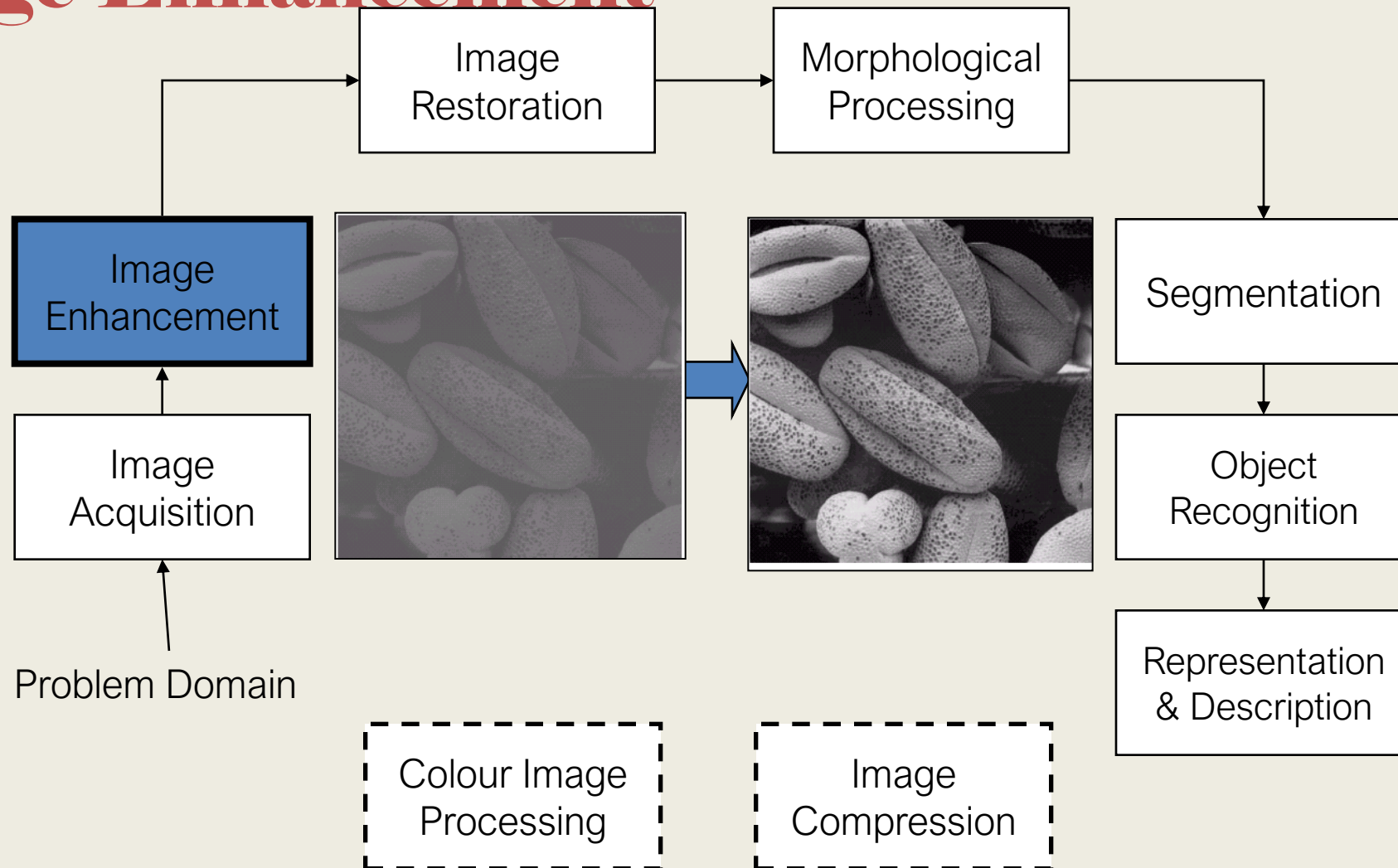
Key Stages in Digital Image Processing



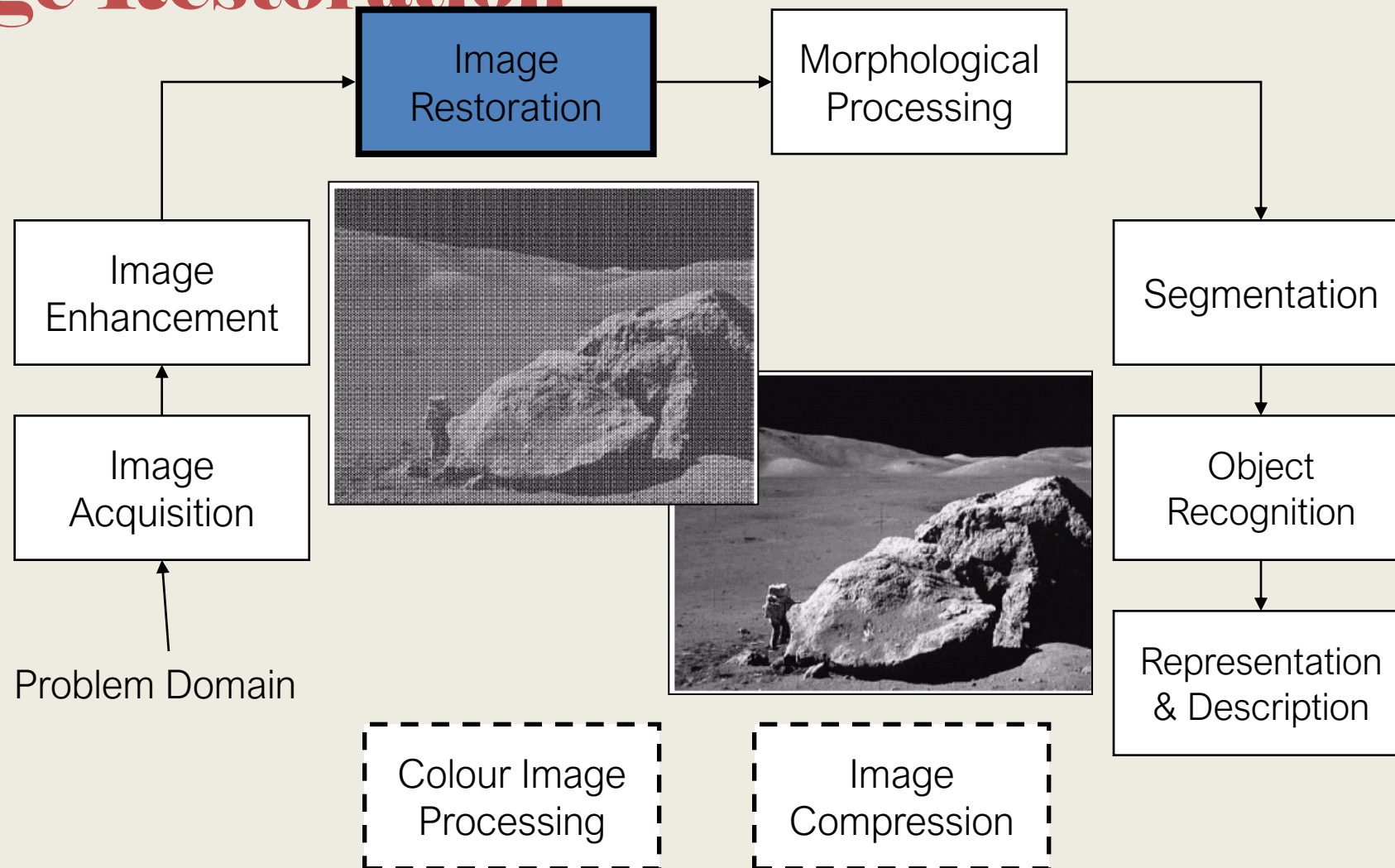
Key Stages in Digital Image Processing: Image Aquisition



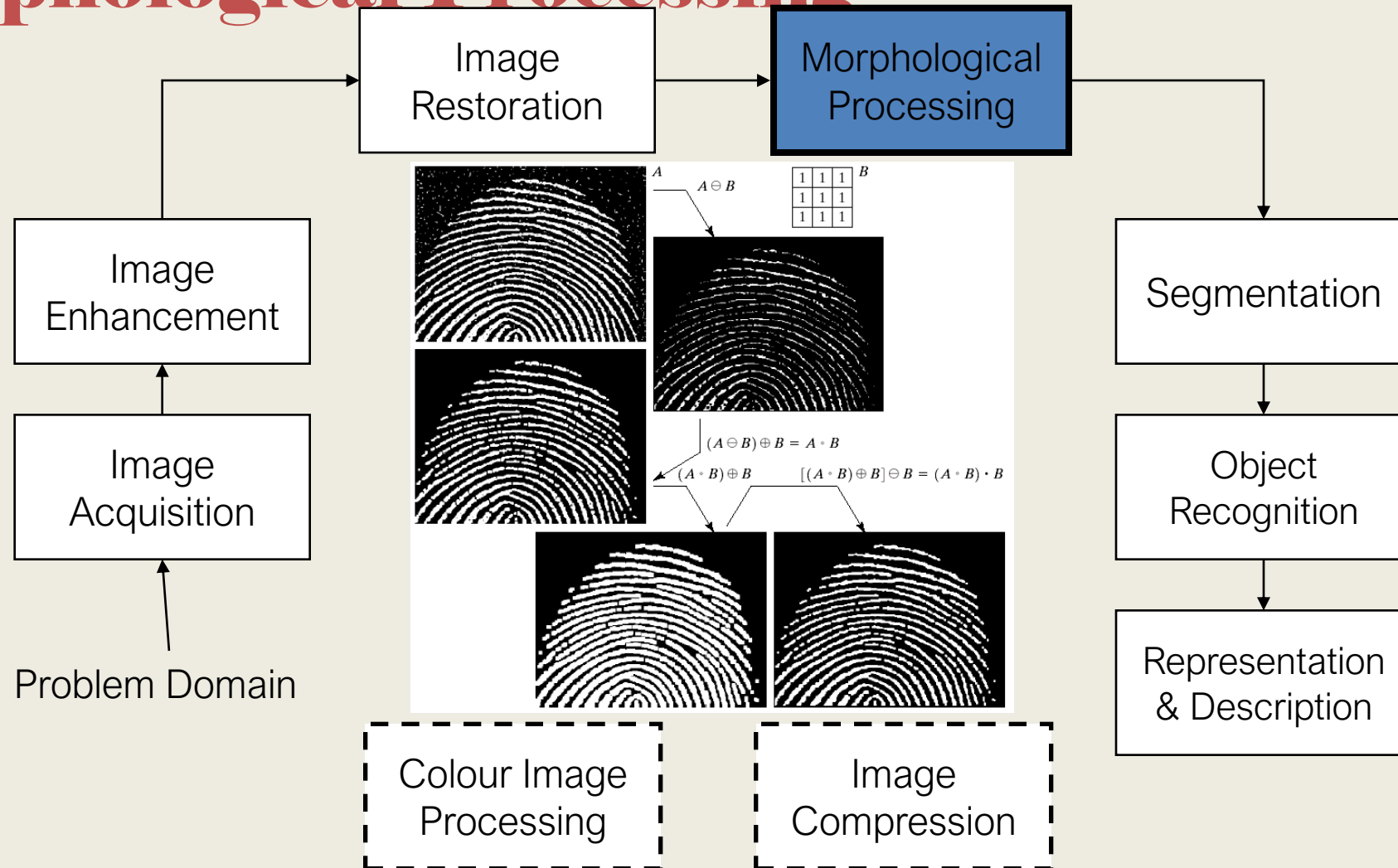
Key Stages in Digital Image Processing: Image Enhancement



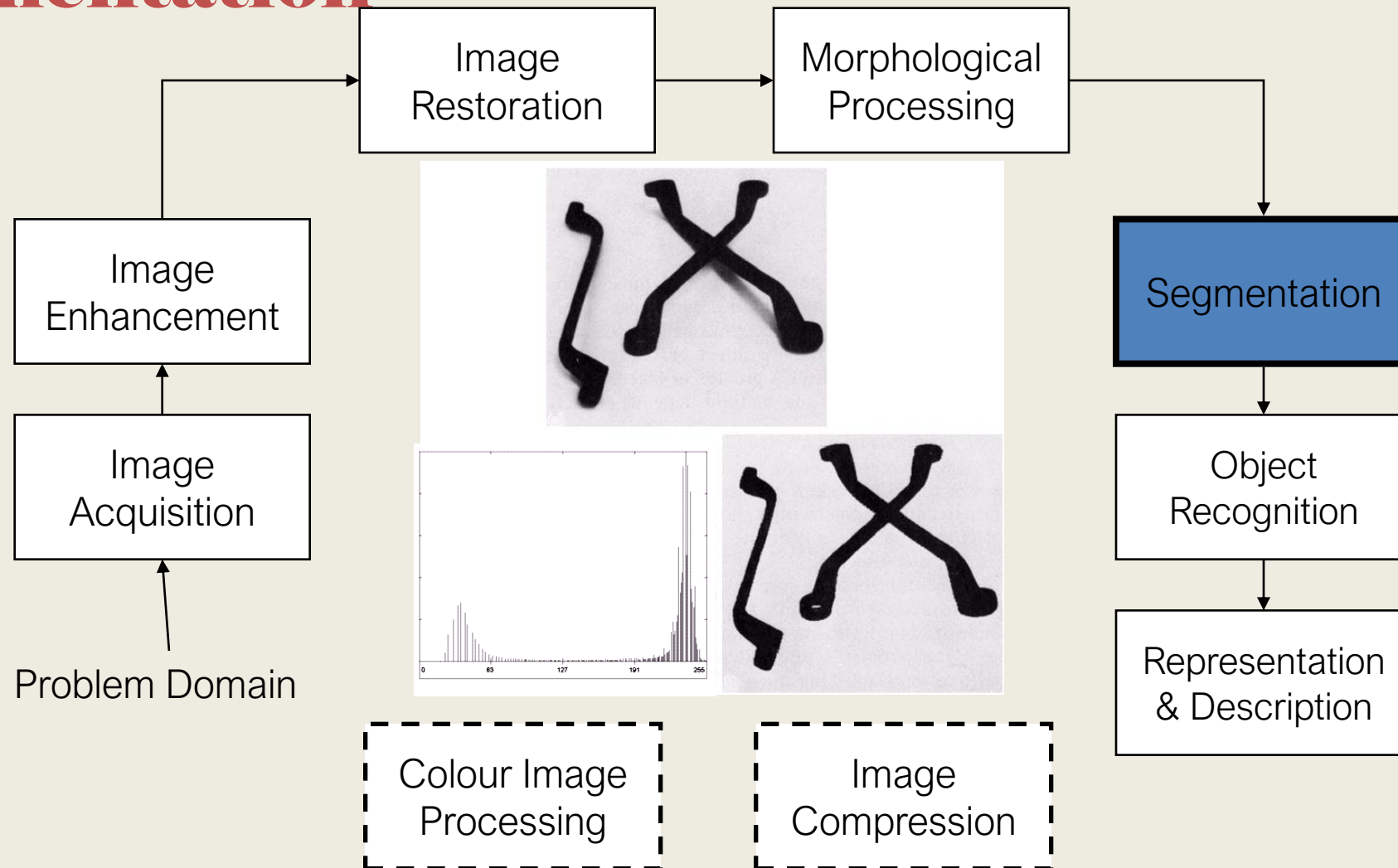
Key Stages in Digital Image Processing: Image Restoration



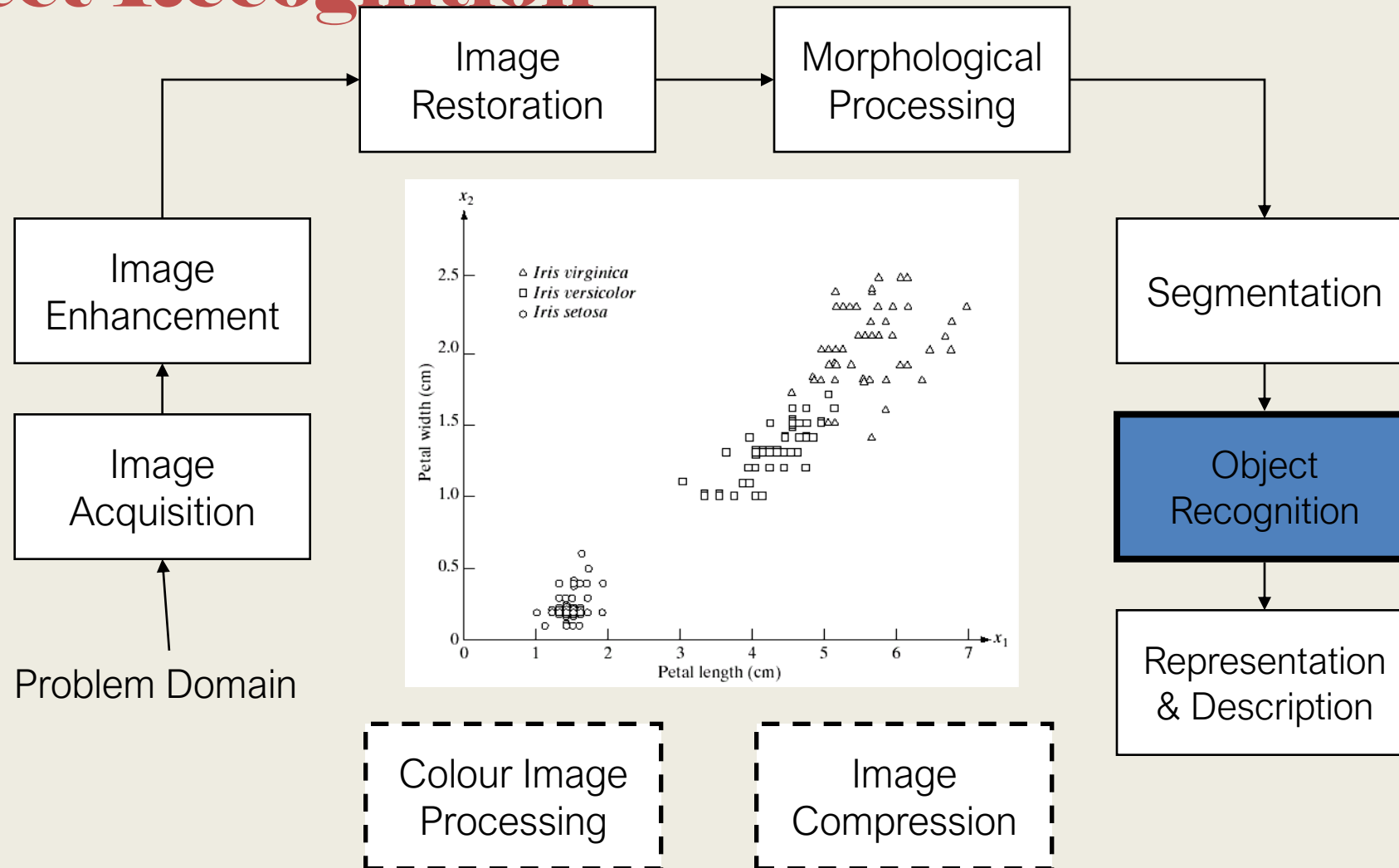
Key Stages in Digital Image Processing: Morphological Processing



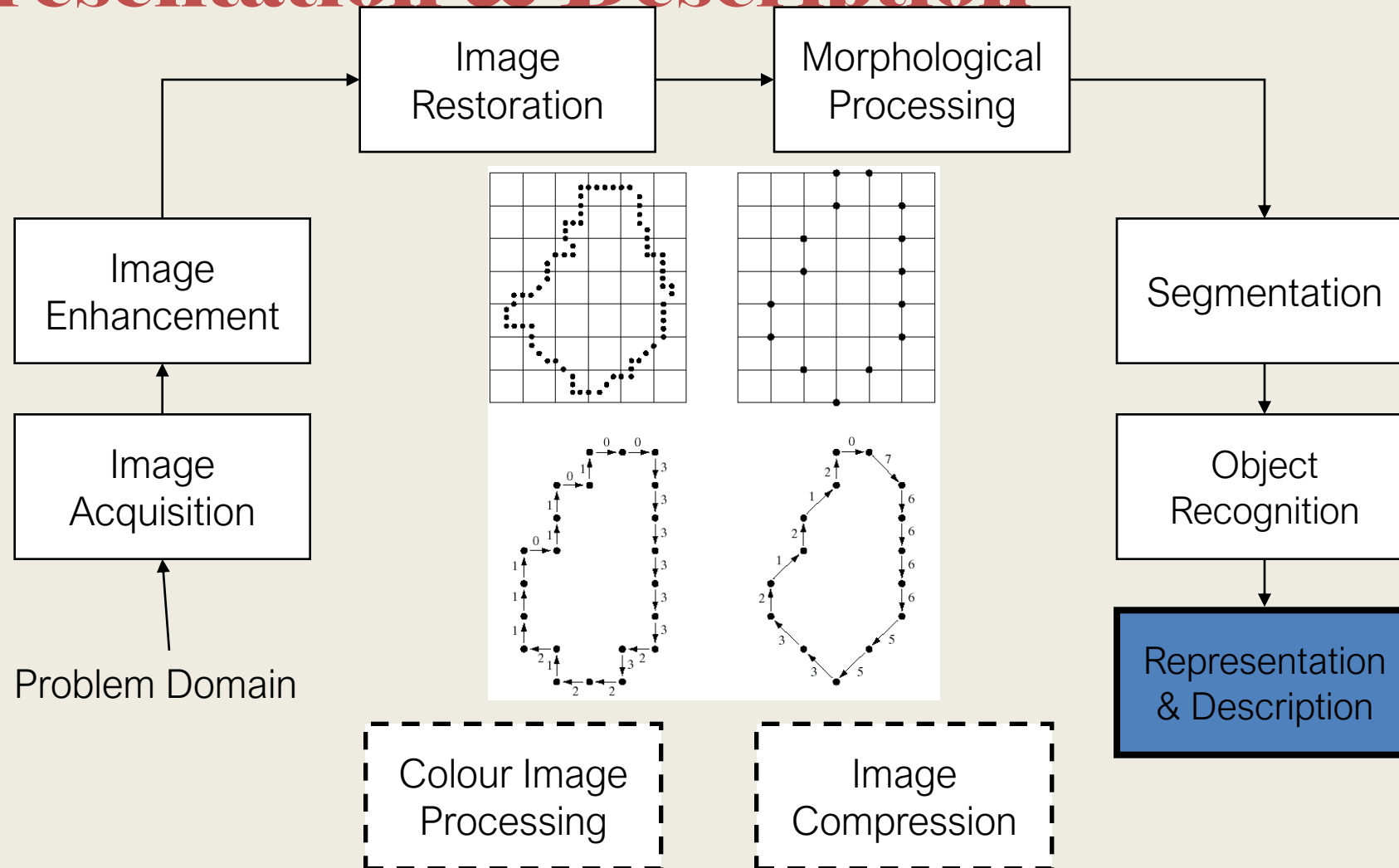
Key Stages in Digital Image Processing: Segmentation



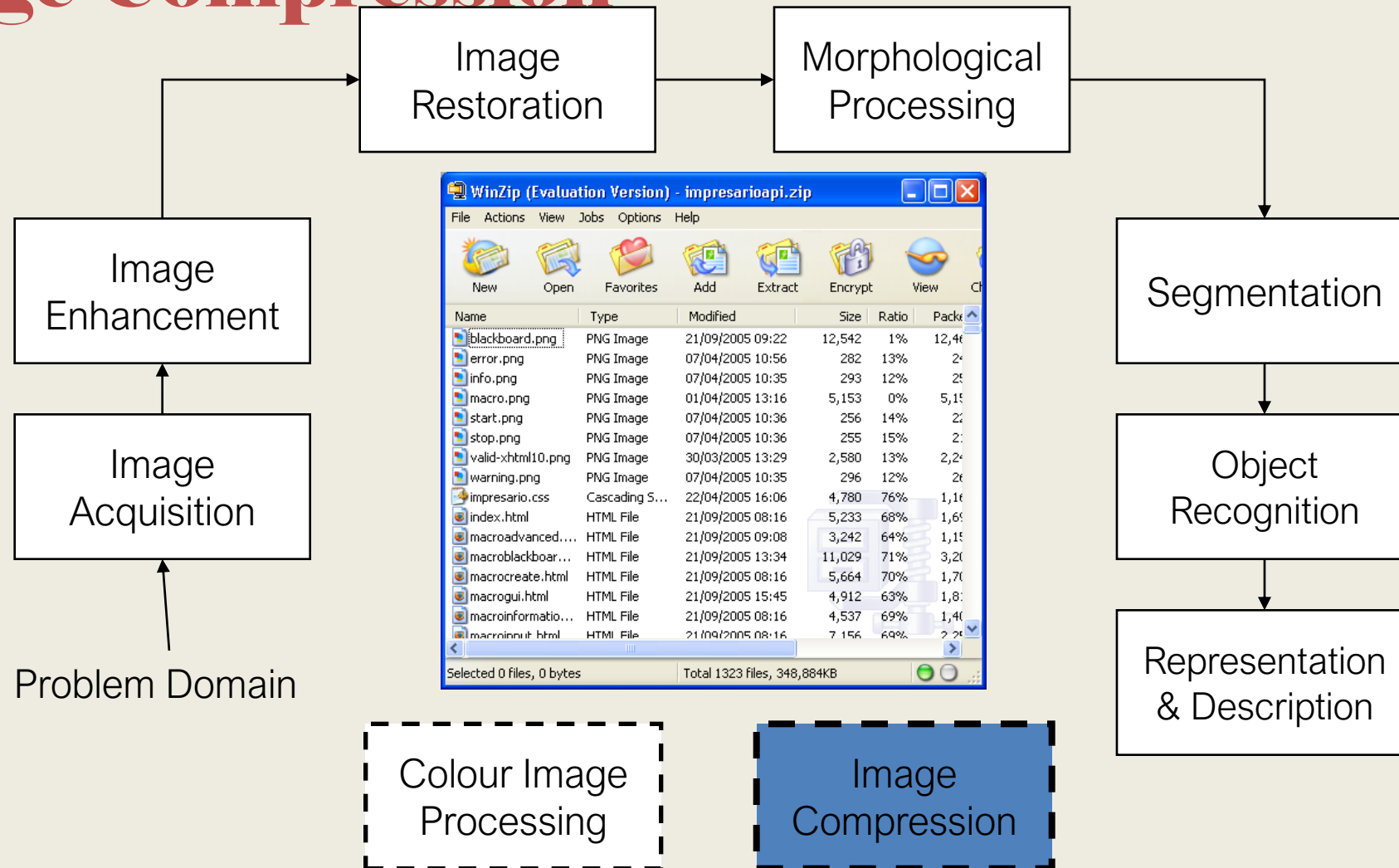
Key Stages in Digital Image Processing: Object Recognition



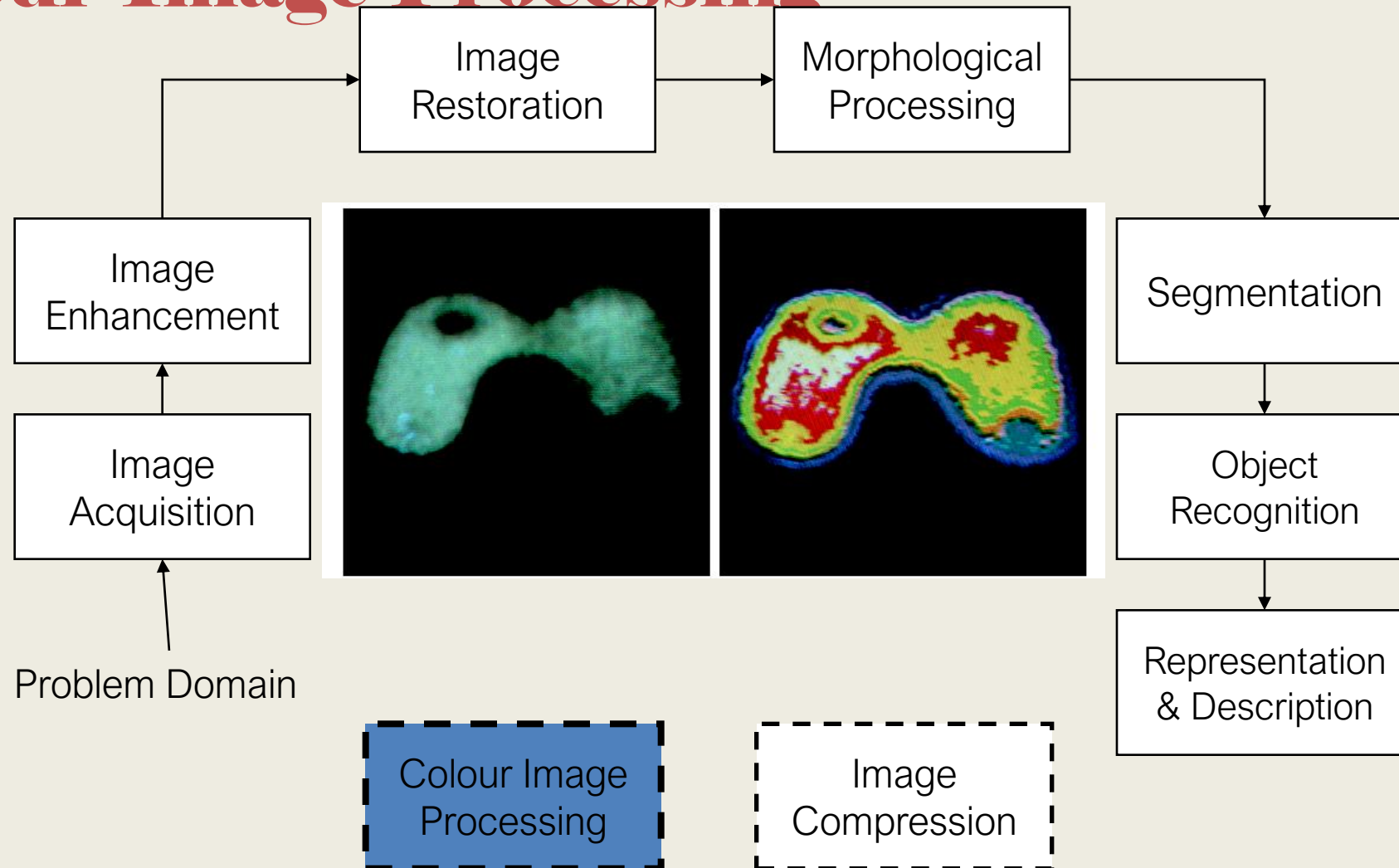
Key Stages in Digital Image Processing: Representation & Description



Key Stages in Digital Image Processing: Image Compression



Key Stages in Digital Image Processing: Colour Image Processing



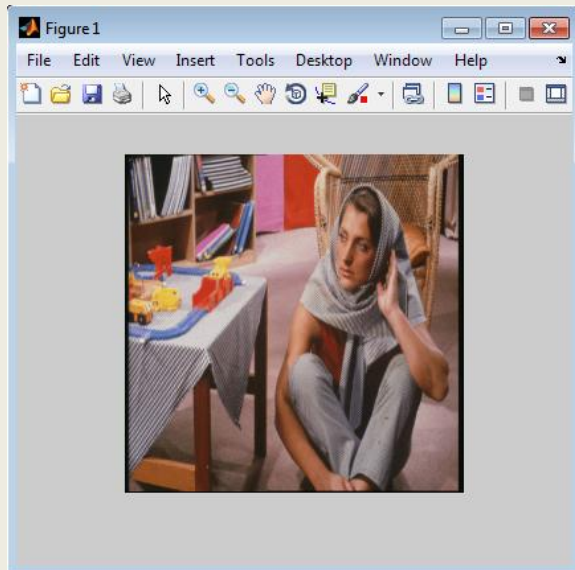
Basic Function

- `Imread()`
- `Imshow()`
- `Resize()`
- `Imwrire()`
- `Rgb2gray()`
- `Figure()`
- `Subplot()`
- `Title()`

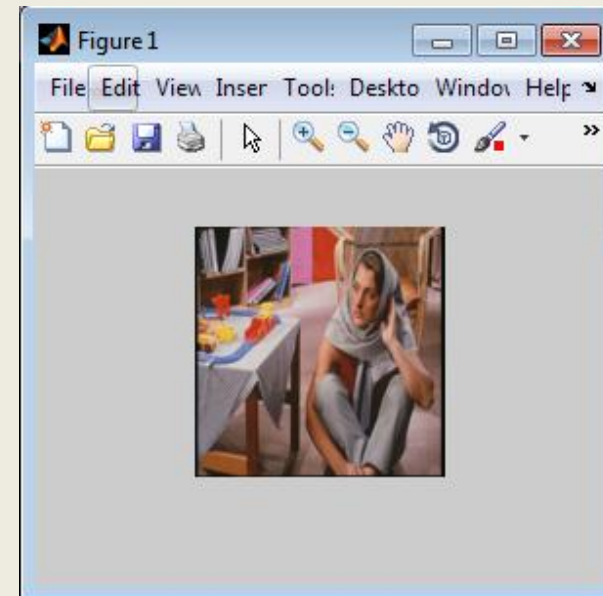
Basic Function

- Change the Size of Image
- `I=imresize(img,[row col])`

`i=imread('barbara.png'); imshow(i);`



`j=imresize(i,0.5); imshow(j);`



Basic Function

- Conversion of Image
 - RGB to Gray scale Conversion
 - `d=rgb2gray(a);`
 - `figure; imshow(d);`
 - Separate Three Component from RGB Image
 - `r=c(:,:,1);`
 - `g=c(:,:,2);`
 - `b=c(:,:,3);`
- `>>| % ctrl+c to halt output!`

RGB to GrayScale Without Command

- `clc;`
- `clear all;`
- `close all;`
- `c=imread ('onion.png');`
- `r=c(:,:,1);`
- `g=c(:,:,2);`
- `b=c(:,:,3);`
- $e = .299*r + .587*g + .114*b;$
- `d1=rgb2gray(c);`
- `figure('Name','Original');`
- `imshow(c);`

- ❧ `figure('Name','Red');`
- ❧ `imshow(r);`
- ❧ `figure('Name','Green');`
- ❧ `imshow(g);`
- ❧ `figure('Name','Blue');`
- ❧ `imshow(b);`
- ❧ `figure('Name','gray manual');`
- ❧ `imshow(e);`
- ❧ `figure('Name','Gray by Inbuild');`
- ❧ `imshow(d1);`

Subplot Command

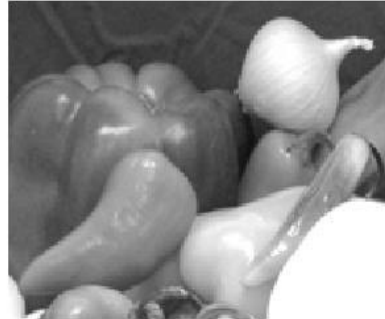
- Subplot: To arrange multiple images into single figure.
 - Subplot(row,column,position);
- Title: To give Title to particular image.
 - Title('Title of Image');

Subplot

original



Red



Green



Green



Grayscale image



Black white image

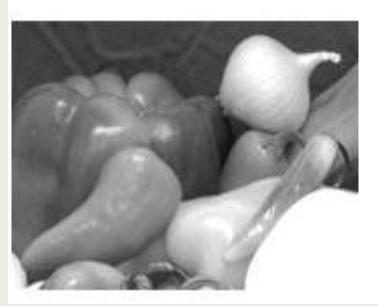


Example of Subplot

- `clc;`
- `clear all;`
- `close all;`
- `c=imread ('onion.png');`
- `r=c(:,:,1);`
- `g=c(:,:,2);`
- `b=c(:,:,3);`
- `d1=rgb2gray(c);`
- `d2=im2bw(c);`
- `subplot(2,3,1); imshow(c); title('original');`
- `subplot(2,3,2); imshow(r);title('Red');`
- `subplot(2,3,3); imshow(g);title('Green');`
- `subplot(2,3,4); imshow(b);title('Blue');`
- `subplot(2,3,5); imshow(d1); title('Grayscale image');`
- `subplot(2,3,6); imshow(d2);title('Black white image');`

Convert R-G-B Component to RGB

- `clc;`
- `clear all;`
- `close all;`
- `c=imread ('onion.png');`
- `r=c(:,:,1);`
- `g=c(:,:,2);`
- `b=c(:,:,3);`
- `fin(:,:,1)=r;`
- `fin(:,:,2)=g;`
- `fin(:,:,3)=b;`
- `figure; imshow(fin);`



R Component



G Component



B Component



Basic Function

- Size: To Find size of image.
 - `[q w e]=size(c);`
 - `q=row`
 - `W=col`
 - `e= dimension`
- `imresize`: To Change size of image.
 - `d1=imresize(c,[64 64]);`
 - `imshow(d1);`
 - `d2=imresize(c,1/4);`
 - `imshow(d2);`

Basic Function

- **imcrop:** To crop the image.
 - `d3=imcrop(c);`
 - `imshow(d3);`
 - `as=imcrop(c,[25 35 50 65]);`
 - `imshow(as);`
- **Imwrite:** To store the image.
 - `imwrite(as,'1.jpg');`
- **imrotate:** To rotate the image.
 - `d4=imrotate(c,270);`
 - `imshow(d4);`

Arithmetic Operation on Image

- **imsubtract:**
- To subtract one image from second image.
 - **Application: Background Subtraction(Removal)**

Original



Background



Object



To add two image(**imadd**)

```
❧ clc
❧ close all
❧ clear all
❧ I = imread('rice.png');
❧ J = imread('cameraman.tif');
❧ K = imadd(J,I,'uint16');
❧ subplot(131);
❧ imshow(I);
❧ subplot(132);
❧ imshow(J);
❧ subplot(133);
❧ imshow(K,[]);
```



To Multiply two image(**immultiply**)

```
❧ clc
❧ close all
❧ clear all
❧ I = imread('moon.tif');
❧ I16 = uint16(I);
❧ J = immultiply(I16,I16);
❧ subplot(131);
❧ imshow(I);
❧ subplot(132);
❧ imshow(I16,[]);
❧ subplot(133);
❧ imshow(J);
```



Rotation

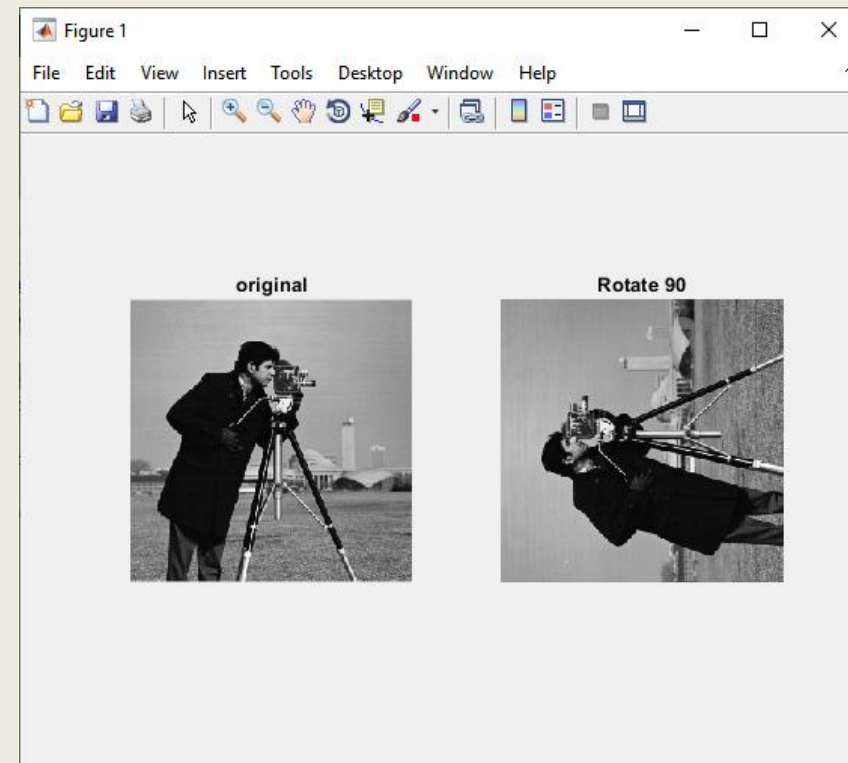
- `B = imrotate(A,angle)` rotates image `A` by `angle` degrees in a counterclockwise direction around its center point. To rotate the image clockwise, specify a negative value for `angle`.

```
img=imread('cameraman.tif');
```

```
rt=imrotate(img,90);
```

```
subplot(1,2,1);imshow(img);title('original');
```

```
subplot(1,2,2);imshow(rt);title('Rotate 90');
```



Application of Arithmetic Operation

- Addition
 - *Averaging images for noise removal*
- Subtraction
 - *Removal of background from images*
 - *Image enhancement*
 - *Image matching*
 - *Moving/displaced object tracking*
- Multiplication
 - *Superimposing of texture on an image*
 - *Convolution and correlation of images*
- And and or operations
 - *To remove the unnecessary area of an image through mask operations*

Thank You

