
**AGH University of Science and Technology**

**Department of Telecommunication**

Zbiginew Hulicki

# Intruduction to image processing

## - *Using Matlab* -

# Basic Arithmetical and Logic Operations on Images

# Digital image processing

## Introduction:

## What is an image?

An image is a visual representation of an object, a person, or a scene produced by an optical device such as a lens, or a camera. This representation is two dimensional (2D).

**A digital image** is a representation of a two-dimensional image using a finite number of points, usually referred to as picture elements, pels, or **pixels**. Each pixel is represented by one or more numerical values:

- For monochrome (gray scale) images, a single value representing the intensity of the pixel (usually in a [0, 255] range) is enough.
- For color images, three values representing the amount of red (R), green (G), and blue (B).

## What is digital image processing?

Digital image processing is a method to perform some operations on image ,to enhance or to extract some useful information from it, and usually treating image a two dimensions signal while applying already set signal processing methods to it.

We can divide the purpose of digital image processing into five groups :

1. Visualization - Observe the objects that are not visible.
2. Image sharpening and restoration - To create a better image
3. Image retrieval - Seek for the image of interest.
4. Measurement of pattern – Measures various objects in an image.
5. Image Recognition – Distinguish the objects in an image

## Applications of digital image processing:

- Image sharpening and restoration
- Medical field
- Remote sensing
- Transmission and encoding
- Machine/Robot vision
- Color processing
- Pattern recognition
- Video processing
- Microscopic Imaging

Since we are dealing with digital image as a matrix of pixels, we can define different **types of digital images**:

1. **Binary**: Each pixel is just black or white. Since there are only two possible values for each pixel (0,1), we only need one bit per pixel.

2. **Gray scale**: Each pixel is a shade of gray, normally from 0 (black) to 255 (white). This range means that each pixel can be represented by eight bits, or exactly one byte. Other gray scale ranges are used, but generally they are a power of 2.
3. **True Color or RGB:** Each pixel has a particular color; that color is described by the amount of red, green and blue in it. If each of these components has a range 0–255, this gives a total of 256^3 different possible colors. Such an image is a "stack" of three matrices; representing the red, green and blue values for each pixel. This means that for every pixel there correspond 3 values.

## Operations on digital images:

In this tutorial we are aiming to describe Three aspects of operations on digital images, and the implementation of these operations using MATLAB environment as follow:

- Arithmetic Operations.
- Logic operations.
- Image histograms.

## 1- Arithmetic operations:

Arithmetic operation on images is the implementation of standard arithmetic operation such as: addition, subtraction, multiplication ,division ,which are performed pixel by pixel between two images or among many images ,or adding a value to each image pixel value which can be used to achieve some effects on image, such as contrast adjustment for example.

## 1.1-Image addition:

This is also called pixel addition. An addition operator adds two images in pixel by pixel fashion.

As we mentioned above, we can also add a value to an image pixel value.

## 1.1.1- two images addition:

**Figure 1.1.1**

## Matlab implementation example :

```
clc;

clear ;

im1 = imread('coins1.jpg'); % reading image
im2 = imread('coins2.jpg'); % reading image

%im1_gray = rgb2gray(im1); % converting RGB image to gray scale *
%im2_gray = rgb2gray(im2); % converting RGB image to gray scale

%im1_bw= im2bw(im1);       % converting RGB image to black and white **
%im2_bw = im2bw(im1);      % converting RGB image to black and white

addition_result = imadd(im1,im2); % performing the addition on the two images


figure('Name','Addition operation','NumberTitle','on')

 subplot(2,2,1),
 imshow(im1);            %Display the image
 title('image 1');

 subplot(2,2,2),
 imshow(im2);            %Display the image
 title('image 2');

 subplot(2,2,[3,4]),
 imshow(addition_result); %Display the result
 title('Operation : addition')
```

Comment:

1. Generally, images must be of the same dimension and of the same data type (e.g. 8 bit integer) for addition and subtraction to be possible between them.
2. (*,**) , Addition operation could be performed on gray scale images and black and white images as well, and to achieve that, RGB images should be converted to gray scale or black and white if they are not originally in gray scale, or B&W.

## 1.1.2- image to constant value addition:

Adding a positive constant value to each pixel location increases its value, and the result called “Contrast adjustment”

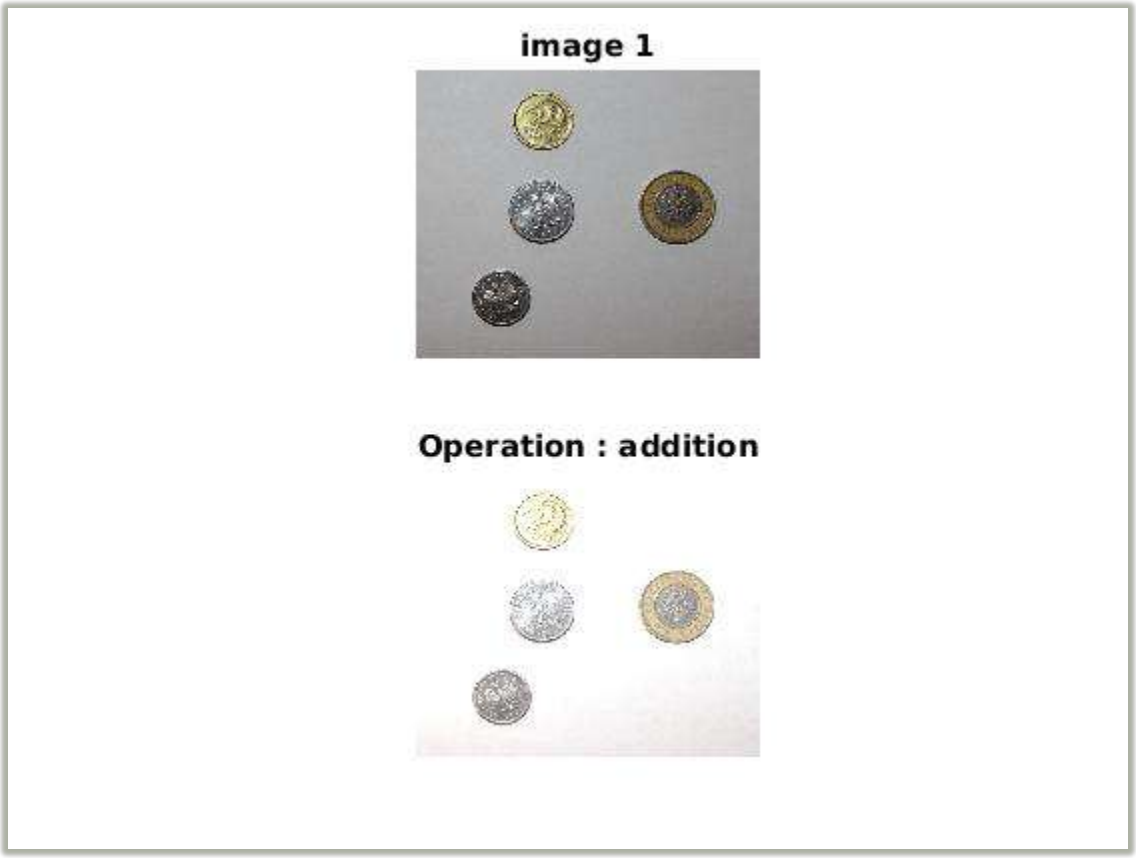


**Figure 1.1.2**

### Matlab implementation example :

```
clc;

clear ;

im1 = imread('coins1.jpg'); % reading image

addition_result = imadd(im1,100); % Add 100 to each pixel value in image1

figure('Name','Addition operation','NumberTitle','on')

 subplot(2,2,[1,2]),
 imshow(im1);            %Display the image
 title('image 1');
```

```
subplot(2,2,[3,4]),
imshow(addition_result); %Display the result
title('Operation : addition')
```

## 1.2 -Image subtraction:

Subtracting one image from another shows us the difference between images, and Subtracting a constant value from each pixel (like addition) can also be used as a basic form of contrast adjustment.

## 1.2.1 – Two images subtraction:


**Figure 1.2.1**

### Matlab implementation example:

```
clc;
```

```
clear ;

im1 = imread('coins1.jpg'); % reading image
im2 = imread('coins2.jpg'); % reading image

%im1_gray = rgb2gray(im1); % converting RGB image to gray scale *
%im2_gray = rgb2gray(im2); % converting RGB image to gray scale

%im1_bw= im2bw(im1);       % converting RGB image to black and white **
%im2_bw = im2bw(im1);      % converting RGB image to black and white

subtract_result = imsubtract(im1,im2); % performing the subtraction on the two
images


figure('Name','Subtraction operation','NumberTitle','on')

 subplot(2,2,1),
 imshow(im1);            %Display the image
 title('image 1');

 subplot(2,2,2),
 imshow(im2);            %Display the image
 title('image 2');

 subplot(2,2,[3,4]),
 imshow(subtract_result); %Display the result
 title('Operation : Subtraction ')
```

## 1.2.2 – constant value from image subtraction:

**Figure 1.2.2**

### Matlab implementation example :

```
clc;

clear ;

im1 = imread('coins1.jpg'); % reading image

subtraction_result = imsubtract(im1,100); % subtract 100 from each pixel value in
image1

figure('Name','subtraction operation','NumberTitle','on')

 subplot(2,2,[1,2]),
 imshow(im1);            %Display the image
 title('image 1');

 subplot(2,2,[3,4]),
 imshow(subtraction_result); %Display the result
 title('Operation : subtraction')
```

## 1.2.3- the absolute difference:

A useful variation on subtraction is the absolute difference between images ,This avoids the potential problem of integer overflow when the difference becomes negative.


**Figure 1.2.3**

**Matlab implementation example:**

```
clc;

clear ;

im1 = imread('coins1.jpg'); % reading image
im2 = imread('coins2.jpg'); % reading image


abs_diff_result = imabsdiff(im1,im2); % performing the absolute difference between
the two images


figure('Name','Absolute difference operation','NumberTitle','on')

 subplot(2,2,1),
 imshow(im1);            %Display the image
 title('image 1');

 subplot(2,2,2),
 imshow(im2);            %Display the image
 title('image 2');

 subplot(2,2,[3,4]),
 imshow(abs_diff_result); %Display the result
```

```
title('Operation : Absolute difference')
```

## 1.3- image multiplication and division:

Multiplication and division can be used as a simple means of contrast adjustment and extension to addition/subtraction, division can be used for image differencing, as dividing an image by another gives a result of 1.0 where the image pixel values are identical and a value not equal to 1.0 where differences occur. However, image differencing using subtraction computationally more efficient.

### 1.3.1 - Image multiplication:

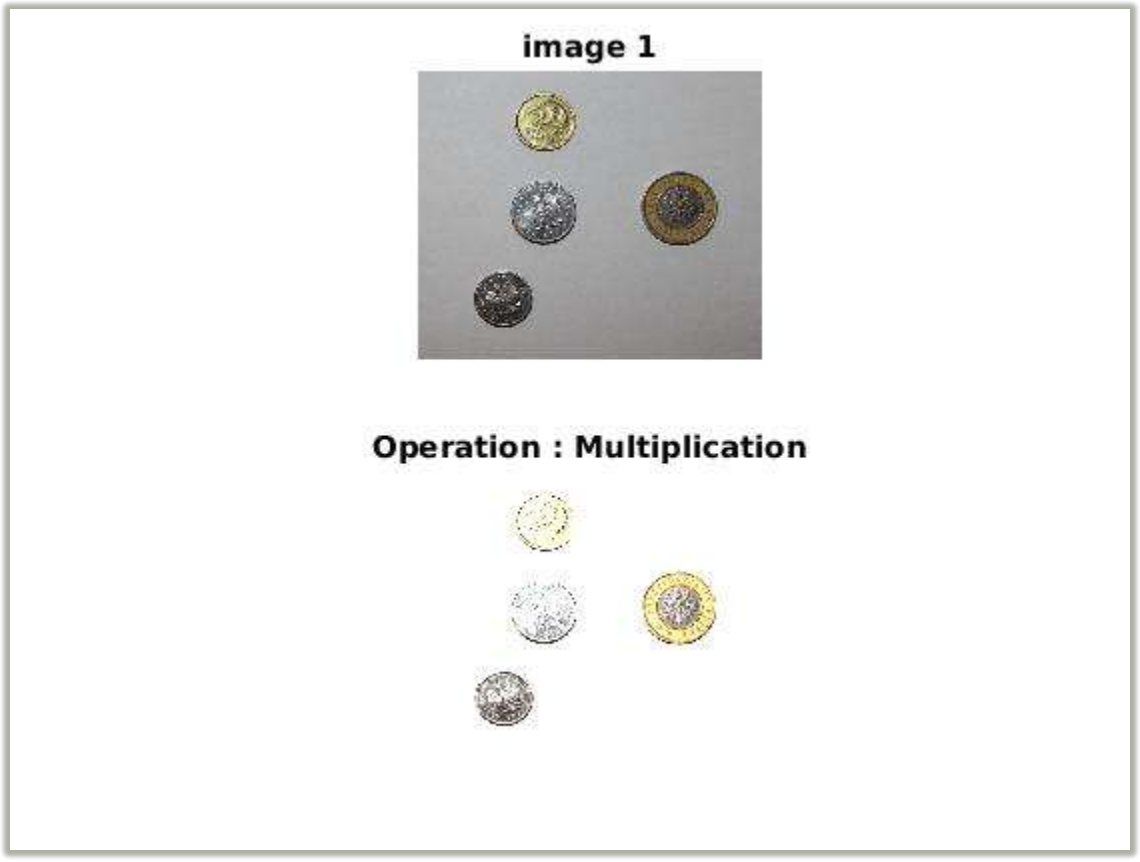

**Figure1.3.1**

### Matlab implementation example:

```
clc;

clear ;

im1 = imread('coins1.jpg'); % reading image

multiplication_result = immultiply(im1,2.5); %multiply image by 2.5

figure('Name','subtraction operation','NumberTitle','on')

 subplot(2,2,[1,2]),
 imshow(im1);            %Display the image
 title('image 1');
 subplot(2,2,[3,4]),
 imshow(multiplication_result); %Display the result
 title('Operation : Multiplication')
```

## 1.3.2 -Image Division:

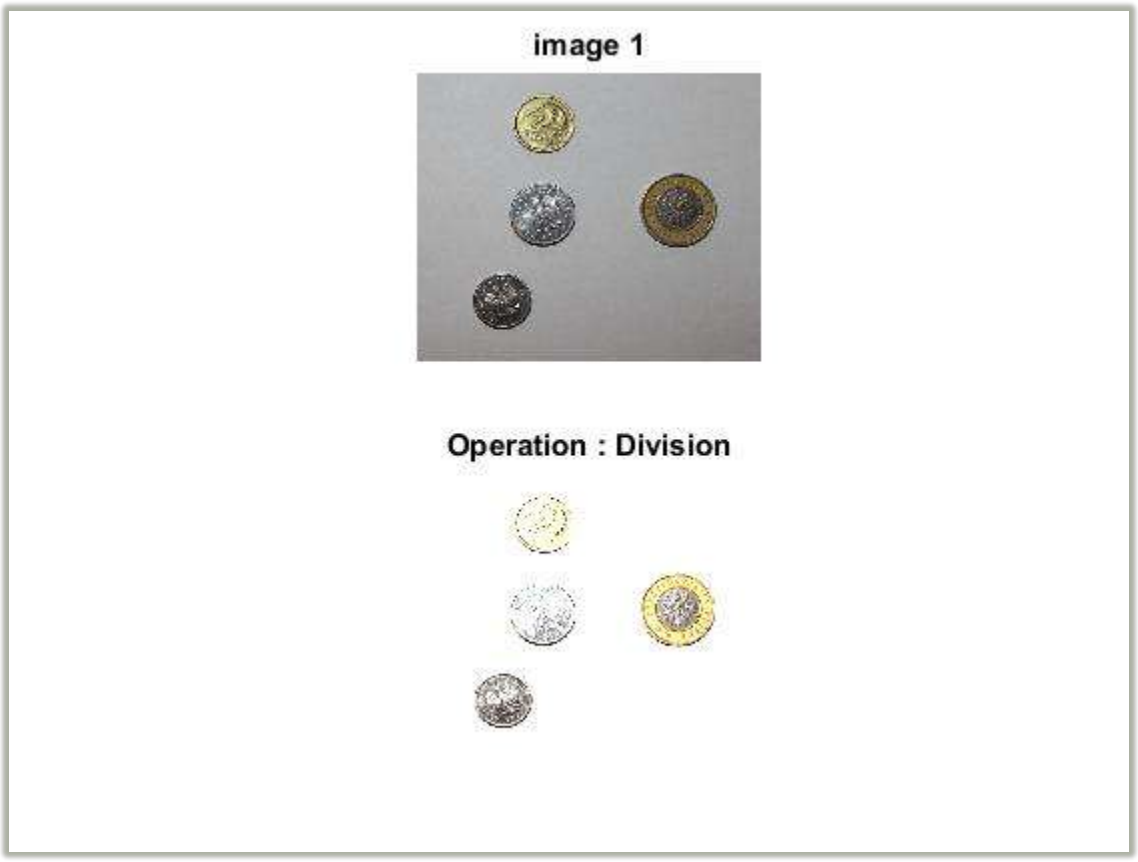


**Figure 1.3.2**

## Matlab implementation example:

```
clc;
clear ;

im1 = imread('coins1.jpg'); % reading image


multiplication_result = immultiply(im1,2.5); %multiply image by 2.5

figure('Name','Division operation','NumberTitle','on')

 subplot(2,2,[1,2]),
 imshow(im1);            %Display the image
 title('image 1');

 subplot(2,2,[3,4]),
 imshow(multiplication_result); %Display the result
 title('Operation : Division')
```

**Comment:**

Multiplying different images together or dividing them by one another is not a common operation in image processing.

---

## 2- Logic operations:

We can perform standard logical operations between images such as NOT, OR, XOR and AND. In general, logical operation is performed between each corresponding bit of the image pixel representation

## 2.1- NOT (inversion):

Not operation inverts the image representation, in the simplest case of a binary image, the (black) background pixels become (white) foreground and vice versa. For greyscale and color images, the procedure is to replace each pixel value : I input (i,j) as follow :

$$I_{output}(i,j) = MAX - I_{output(i,j)}$$

Where MAX is the maximum possible value in the given image representation. Thus, for an 8-bit grey-scale image (or for 8-bit channels within a color image), MAXĽ255.

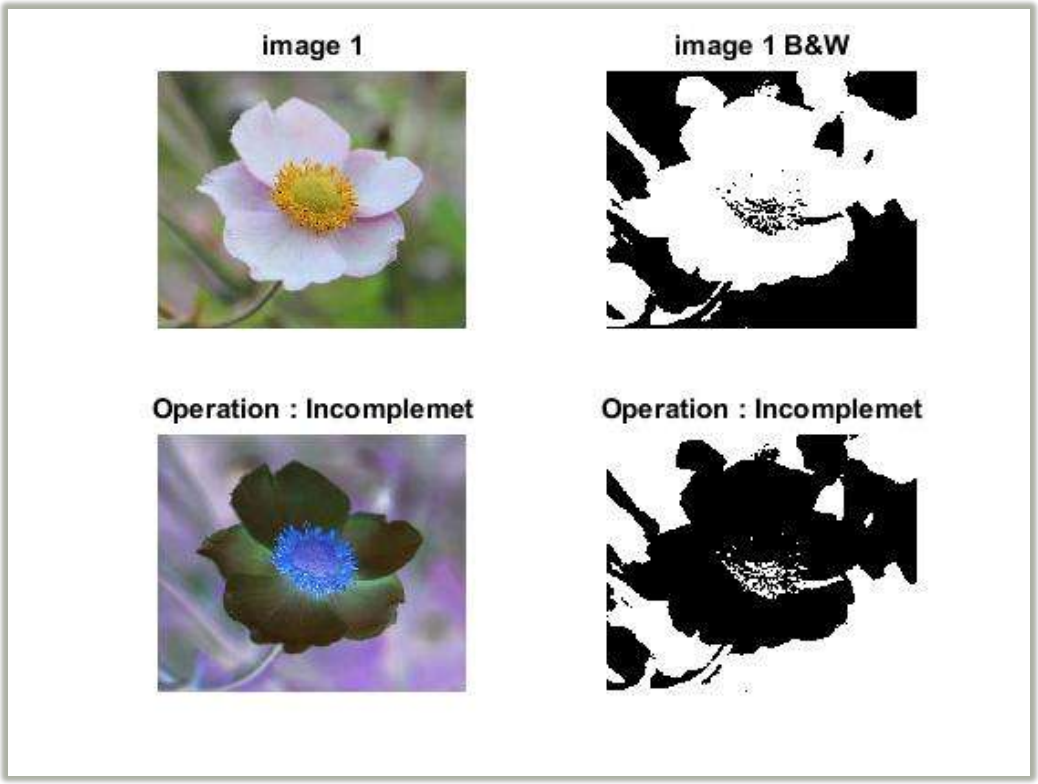


**Figure2.1**

## Matlab implementation example:

```
clc;
clear ;

im1 = imread('rose1.jpg'); % reading image

im1_bw = im2bw(im1);

incomplement_result_1 = imcomplement(im1); %Invert the RGB image
incomplement_result_2 = imcomplement(im1_bw);  %Invert the B&W image

figure('Name','Logic operation "Incomplemet"','NumberTitle','on')
 subplot(2,2,1),
 imshow(im1);            %Display the image
 title('image 1');
 subplot(2,2,3),
 imshow(incomplement_result_1); %Display the result
 title('Operation : Incomplemet')
  subplot(2,2,2),
 imshow(im1_bw);             %Display the image
 title('image 1 B&W');
 subplot(2,2,4),
 imshow(incomplement_result_2); %Display the result
 title('Operation : Incomplemet')
```

## 2.2- OR / XOR:

Logical OR(and XOR) is useful for processing binary-valued images (0 or 1) to detect objects which have moved between frames.

### 2.2.1- Logic OR operation:

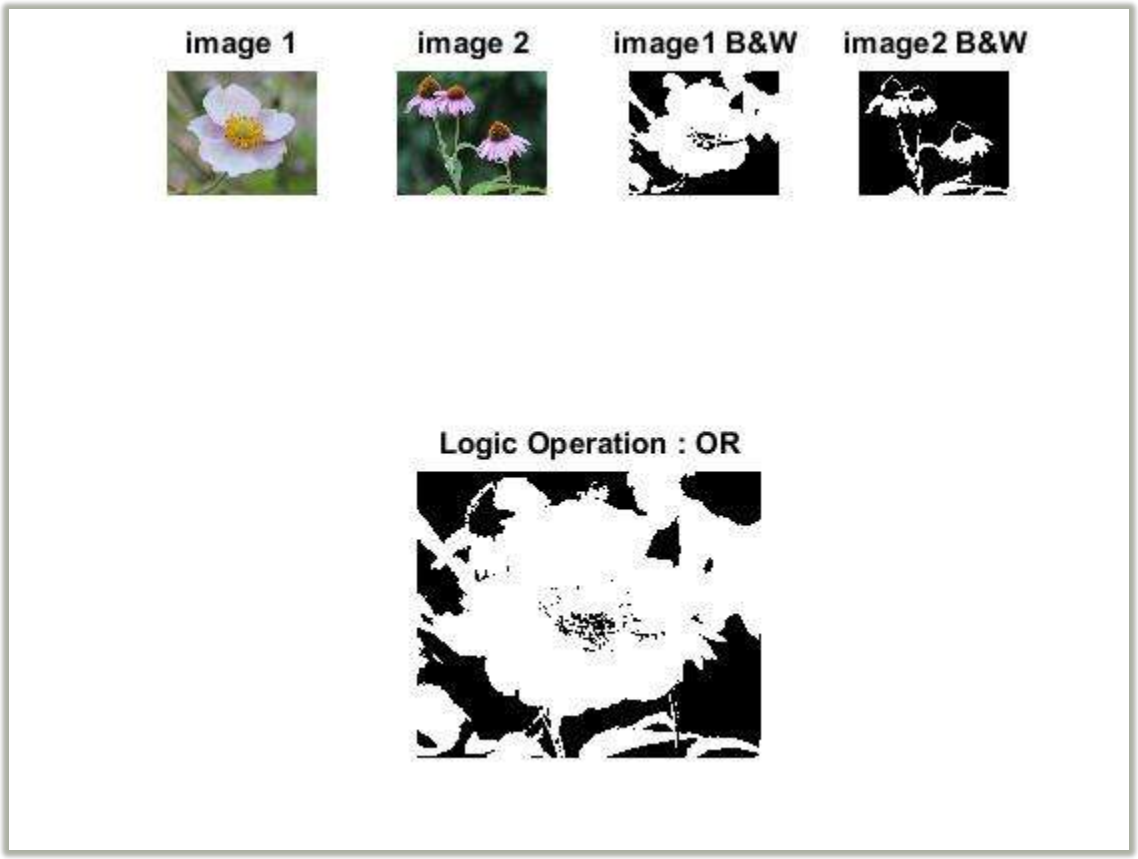


**Figure 2.2.1**

### Matlab implementation example:

```
clc;
clear ;

im1 = imread('rose1.jpg'); % reading image
im2 = imread('rose2.jpg'); % reading image

im1_bw = im2bw(im1);        % converting RGB image to black and white
im2_bw = im2bw(im2);       % converting RGB image to black and white

Operation_or_result = or(im1_bw,im2_bw); % performing OR logic on the images
```

```
figure('Name','Logic operation "OR"','NumberTitle','on')

 subplot(4,4,1),
 imshow(im1);            %Display the image
 title('image 1');

 subplot(4,4,2),
 imshow(im2);            %Display the image
 title('image 2');

 subplot(4,4,3),
 imshow(im1_bw);            %Display the image
 title('image1 B&W');

 subplot(4,4,4),
 imshow(im2_bw);            %Display the image
 title('image2 B&W');

 subplot(2,2,[3,4]),
 imshow(Operation_or_result); %Display the result
 title('Logic Operation : OR')
```

### 2.2.2- Logic XOR operation:

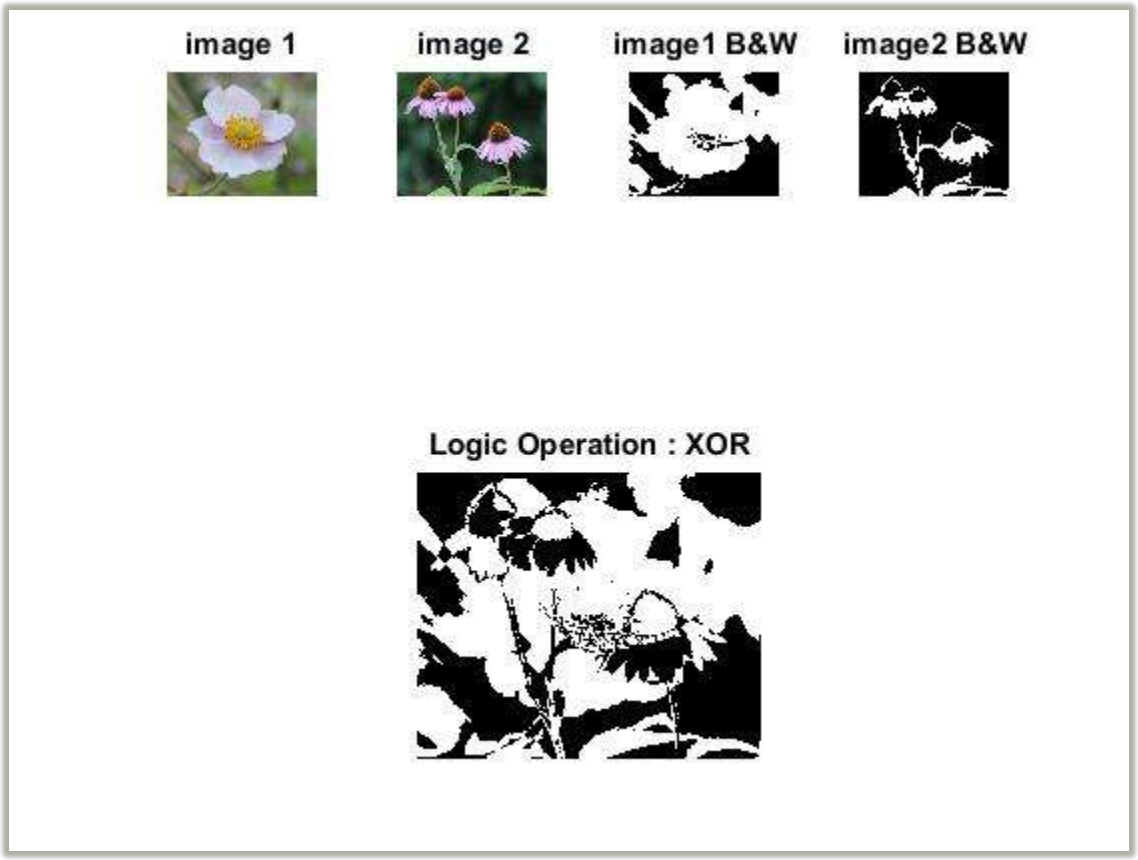


**Figure 2.2.2**

## Matlab implementation example:

```
clc;
clear ;

im1 = imread('rose1.jpg'); % reading image
im2 = imread('rose2.jpg'); % reading image


im1_bw = im2bw(im1);       % converting RGB image to black and white *
im2_bw = im2bw(im2);      % converting RGB image to black and white

Operation_xor_result = xor(im1_bw,im2_bw); % performing OR logic on the images


figure('Name','Logic operation "XOR"','NumberTitle','on')

 subplot(4,4,1),
 imshow(im1);             %Display the image
 title('image 1');

 subplot(4,4,2),
 imshow(im2);             %Display the image
 title('image 2');

 subplot(4,4,3),
 imshow(im1_bw);            %Display the image
 title('image1 B&W');

 subplot(4,4,4),
 imshow(im2_bw);            %Display the image
 title('image2 B&W');


 subplot(2,2,[3,4]),
 imshow(Operation_xor_result); %Display the result
 title('Logic Operation : XOR')
```

**Comments:**

- Note that the images are first converted to binary using the Matlab im2bw function (with an automatic threshold (section2.4).
- note that the resulting images from the im2bw function and the xor logical operation is of Matlab type 'logical'.

## 2.3- AND:

Logical AND is commonly used for detecting differences in images, highlighting target regions with a binary mask or producing bit-planes through an image.

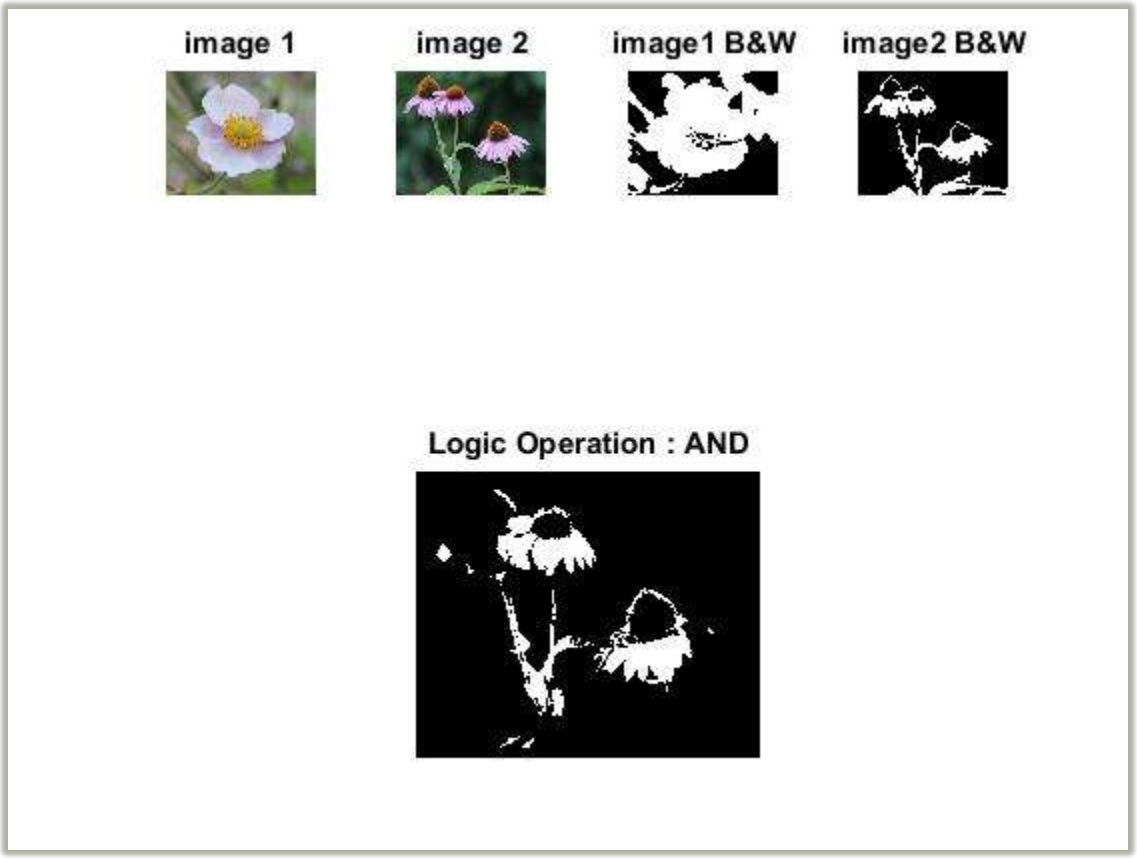


**Figure2.3**

**Matlab implementation example:**

```
clc;
clear ;

im1 = imread('rose1.jpg'); % reading image
im2 = imread('rose2.jpg'); % reading image


im1_bw = im2bw(im1);       % converting RGB image to black and white *
im2_bw = im2bw(im2);       % converting RGB image to black and white

Operation_AND_result = and(im1_bw,im2_bw); % performing AND logic on the images


figure('Name','Logic operation "AND"','NumberTitle','on')

 subplot(4,4,1),
 imshow(im1);            %Display the image
 title('image 1');

 subplot(4,4,2),
 imshow(im2);            %Display the image
 title('image 2');

 subplot(4,4,3),
 imshow(im1_bw);           %Display the image
 title('image1 B&W');

 subplot(4,4,4),
 imshow(im2_bw);           %Display the image
 title('image2 B&W');


 subplot(2,2,[3,4]),
 imshow(Operation_AND_result); %Display the result
 title('Logic Operation : AND')
```

**Comment:**

the operators for AND is ‘&’ whilst the operator for OR is ‘|’ and are applied in infix notation form as A & B, A | B.

## 2.4- Thresholding:

Thresholding produces a binary image from a grey-scale or color image by setting pixel values to 1 or 0 depending on whether they are above or below the threshold value.

Thresholding is commonly used to separate or segment a region or object within the image based upon its pixel values.

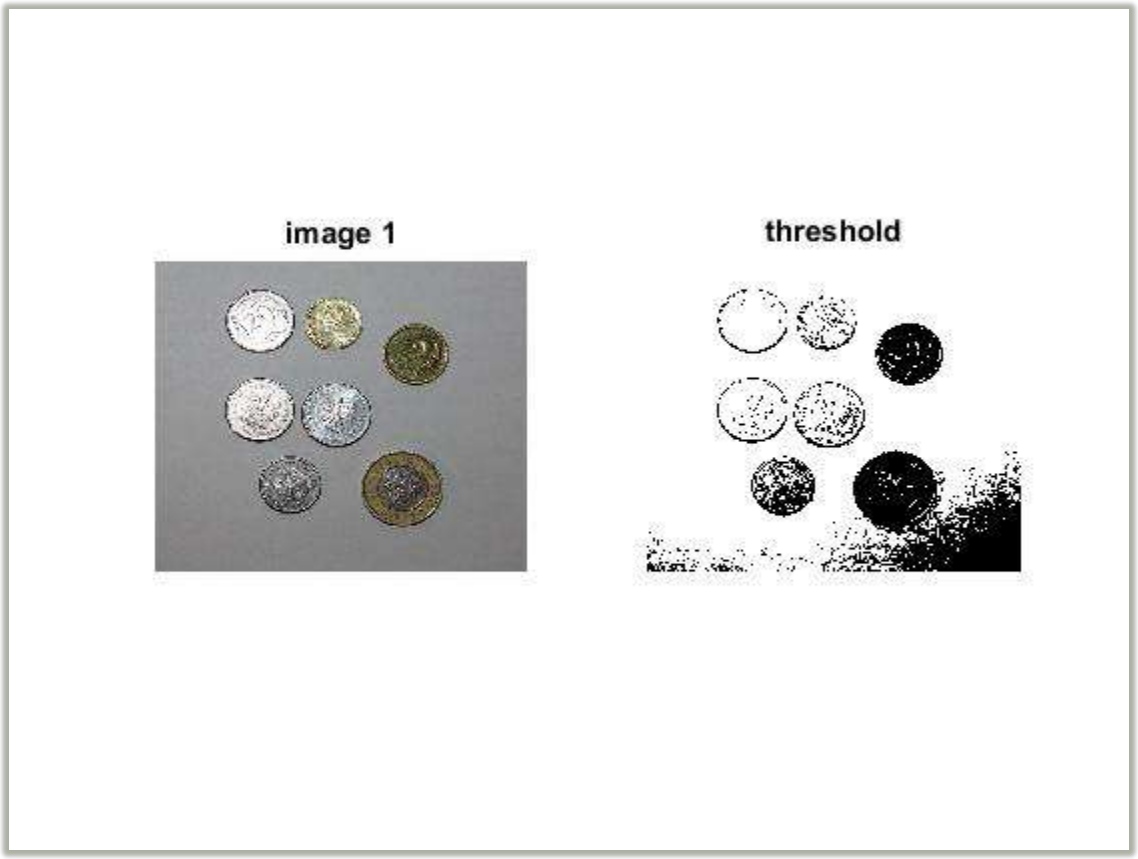


**Figure 2.4**

### Matlab implementation example:

```
clc;
clear ;

im1 = imread('coins3.jpg'); % Reading image

thre_im1 = im2bw(im1,0.5);  % Perform thresholding

figure('Name','Thresholding','NumberTitle','on')
subplot(1,2,1)
```

```
imshow(im1);                %show original image
title('image 1')

subplot(1,2,2)
imshow(thre_im1);           %show image after thresholding
title('threshold')
```

**Comments:**

Common variations on simple thresholding are:

- The im2bw function automatically converts color images (such as the input in the example) to grayscale and scales the threshold value supplied (from0 to 1)
- The use of two thresholds to separate pixel values within a given range;
- The use of multiple thresholds resulting in a labeled image with portions labeled 0 to N;
- Retaining the original pixel information for selected values (i.e. above/between thresholds) whilst others are set to black.

---

---

**For Matlab Usage:**

1. Create a new script in Matlab environment :

   Home >> New Script

2. Save it as a **.m** file under the following path:

   for Windows: My documents >> MATLAB
   for Linux distribution : home/MATLABWORKSPACE

Note that if you want to upload any file to the Matlab environment all of them should be placed in the same directory: MATLAB WORK SPACE folder (e.g. ”pictures”).

3. Now you can easily implement a code in the editor.

Remember that before you start to run your program you should save all the changes you have done in the editor.

**Biblioghraphy:**

1-Digital image processing using Matlab :
Rafael C. Gonzalez
Richard E. Woods
Steven L. Eddins

2-Fundamentals of digital image processing : A Practical Approach with examples in Matlab
Chris Solomon
Tobby breckon