|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name:            Lab Section: Id No:** | | | | |
| **Pre-lab Session work (5M)** | **In-Lab Session work (15M)** | **Post Lab session work (5M)** | **Viva**  **(5M)** | **Total Marks**  **30M** |
|  |  |  |  |  |
| **Remarks if any:** | | | | |
| **Date: Signature of the Instructor Marks awarded** | | | | |

**2. Arithmetical and Logical Operations on Images**

Image arithmetic applies one of the standard arithmetic operations or a logical operator to two or more images. The operators are applied in a pixel-by-pixel way, i.e. the value of a pixel in the output image depends only on the values of the corresponding pixels in the input images. Hence, the images must be of the same size. Although image arithmetic is the simplest form of image processing, there is a wide range of applications. A main advantage of arithmetic operators is that the process is very simple and therefore fast.

Logical operators are often used to combine two (mostly binary) images. In the case of integer images, the logical operator is normally applied in a bitwise way.

In this lecture we will talk about arithmetic operations such as subtraction and averaging as well as logic operations such as Not, AND, and OR. In this lecture we will see how we can remove noise from an image by using image averaging.

**Objectives:**

To

1. understand the arithmetic operations such as addition, subtraction multiplication, division and averaging.
2. explore the logic operations such as NOT, OR, AND, and XOR etc.

**2.1 Image Arithmetic:**  An image is represented in a matrix format. To perform image arithmetic the size of the two matrices should be same. The operation on two images results in a new image.

2.1.1 Image Addition: To add two images or a constant value, use ‘imadd’ function which is built in function. Imadd method can be used to brighten an image by adding a constant value to each pixel.

Example 2.1: Image addition by a constant value

Refer MIIP\_2\_1.m

**< Type the Matlab codes here >**

**< Plot the figures and type the results here >**

Example 2.2: Addition of two images

Refer MIIP\_2\_2.m

**< Type the Matlab codes here >**

**< Plot the figures and type the results here >**

**2.1.2 Image Subtraction:** ‘imsubtract’ is builtin Matlab function to subtract each pixel value in one of the input images from another, or subtract a constant value from an image. Image subtraction can be used for more complex image processing. Image subtraction is used to detect images in a series of images of the same scene. For this operation the images must be the same size and class.

1. Image subtraction from a constant value.

Example 3:

Refer MIIP\_2\_3.m

**< Type the Matlab codes here >**

**< Plot the figures and type the results here >**

1. Image subtraction from another image.

Example 4:

Refer MIIP\_2\_4.m

**< Type the Matlab codes here >**

**< Plot the figures and type the results here >**

**2.1.3 Image Multiplication:** ‘immultiply’ multiplies each pixel value in one of the input images from the corresponding pixel in the other input image or multiply a constant value and returns the result in the corresponding pixel in an output image. If elements of output image exceeding the range of integer type are truncated, and fractional values are rounded. Image multiplication can be used for more complex image processing. For this operation the images must be the same size and class.

1. Image multiplication with a constant value.

Example 5:

Refer MIIP\_2\_5.m

**< Type the Matlab codes here >**

**< Plot the figures and type the results here >**

1. Image multiplication with another image.

Example 6:

Refer MIIP\_2\_6.m

**< Type the Matlab codes here >**

**< Plot the figures and type the results here >**

2.1.4: Image division:

(a) Divide an Image by a Constant Factor

Example 7:

Refer MIIP\_2\_7.m

**< Type the Matlab codes here >**

**< Plot the figures and type the results here >**

(b) Divide Image by another image

Example 8:

Refer MIIP\_2\_8.m

**< Type the Matlab codes here >**

**< Plot the figures and type the results here >**

**2.2 Logical Operations**: **Logic operations** provide a powerful complement to implementa-tion of image processing algorithms based on morphology. The logic operations are applied for binary images. These operations consist of 4 basic binary operations: AND, OR, and XOR and a unary operator NOT. Secondary operators can be created by combining the three binary operators with the unary operator, yielding: NAND, NOR, and XNOR. Logic operations are performed on a pixel by pixel basis between corresponding pixels of two or more images (except NOT, which operates on the pixels of a single image).

Example 9:

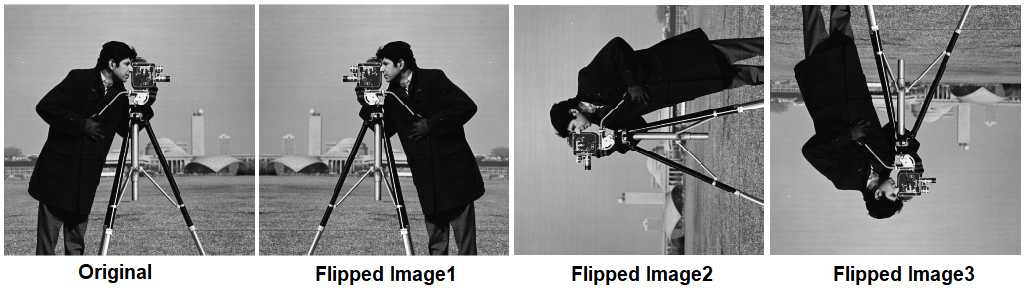
Refer MIIP\_2\_9.m

**< Type the Matlab codes here >**

**< Plot the figures and type the results here >**

**Lab2-Exercise Questions**

**Exercise1:** Write a Matlab code for the following flipped images



**Exercise2:** Consider the following figure. Write a Matlab program to extract the appropriate rectangular portion of the image (shown in a red color rectangular) and display in a separate window. Identify the size of the resultant image and calculate the memory size.

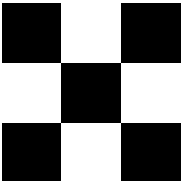


**Exercise3:** Consider the following figure which is a rectangular in size. Write a matlab program to display the image in a suitable square image (2*n* X2*n*).



imtool(J,'DisplayRange',[]);

**Exercise4:** Consider the following image of size 180X180. Develop Matlab code to display the image using arithmetic and logic operations.



**Exercise5:** Write a Matlab code to read and display an image **'board.tif'.** Now display a portion of the image that has the dimensions of 256X256.

**Exercise6:** Enter the following Matlab code and observe the results

**clear; close all; clc;**

**im = imread('cameraman.tif');**

**im1=imdivide(im,64);**

**im2=immultiply(im1,64);**

**figure();**

**subplot(1,3,1); imshow(im,[]); title('Original Image');**

**subplot(1,3,2); imshow(im1,[]); title('Divided by 4');**

**subplot(1,3,3); imshow(im2,[]); title('Multiply by 4');**

Comment on the result. Why is the result not equivalent to the original image?