# LAB 5: MATRIX ARITHMETIC AND IMAGE ARITHMETIC

## Matrix Arithmetic

**Experiment No. 15 Create a matrix**

```
c = [0 2 4 6; 8 10 12 14; 16 18 20 22];
```

**Experiment No. 16 Convert matrix to image.**

```
I = image(c);
```

**Experiment No. 17 Plot a line, and then create an image on top of the line. Return the image Object.**

```
plot(1:3)
hold on
c = [1 2 3; 4 5 6; 7 8 9];
im = image(c);
im.AlphaData = 0.5;
```

**Experiment No. 18 Add a number to a matrix**

```
c = [0 2 4 6; 8 10 12 14; 16 18 20 22];
g = c + 2
```

**Experiment No. 19 Multiply a matrix by a number**

```
c = [0 2 4 6; 8 10 12 14; 16 18 20 22];
f = c * 2;
```

**Experiment No. 20 Subtract a number from a matrix**

```
c = [0 2 4 6; 8 10 12 14; 16 18 20 22];
m = c - 2;
```

**Experiment No. 21 Matrix Addition**

```
c = [0 2 4 6; 8 10 12 14; 16 18 20 22];
v = [1 0 7 3; 4 2 10 1; 4 0 20 9];
x = c + v;
```

**Experiment No. 22 Matrix Subtraction**

```
c = [0 2 4 6; 8 10 12 14; 16 18 20 22];
v = [1 0 7 3; 4 2 10 1; 4 0 20 9];
```

x = c - v;

**Experiment No. 23 Matrix Multiplication**

c = [0 2 4 ; 8 10 12; 16 18 9];

v = [1 0 7; 4 2 10; 4 0 20];

x = c * v;

# Image Arithmetic

Add, subtract, multiply, and divide images

Image arithmetic is the implementation of standard arithmetic operations, such as addition, subtraction, multiplication, and division, on images. Image arithmetic has many uses in image processing both as a preliminary step in more complex operations and by itself. For example, image subtraction can be used to detect differences between two or more images of the same scene or object.

**Functions:**

- imabsdiff Absolute difference of two images
- imadd Add two images or add constant to image
- imcomplement Complement image
- imdivide Divide one image into another or divide image by constant
- imlincomb Linear combination of images
- immultiply Multiply two images or multiply image by constant
- imsubtract Subtract one image from another or subtract constant from image

## Experiment No. 24 imabsdiff

Absolute difference of two images

### Syntax

```
Z = imabsdiff(X,Y)
```

### Description

Z = imabsdiff(X,Y) subtracts each element in array Y from the corresponding element in array X and returns the absolute difference in the corresponding element of the output array Z.

### Example

Display Absolute Difference between Filtered image and Original

1. Read image into workspace.
   ```
   I = imread('cameraman.tif');
   ```
2. Filter the image.
   ```
   J = uint8(filter2(fspecial('gaussian'), I));
   ```
3. Calculate the absolute difference of the two images.
   ```
   K = imabsdiff(I,J);
   ```
4. Display the absolute difference image.
   ```
   figure
   imshow(K,[])
   ```

## Experiment No. 25 imadd

Add two images or add constant to image

### Syntax

```
Z = imadd(X,Y)
```

### Description

Z = imadd(X,Y) adds each element in array X with the corresponding element in array Y and returns the sum in the corresponding element of the output array Z.

### Examples

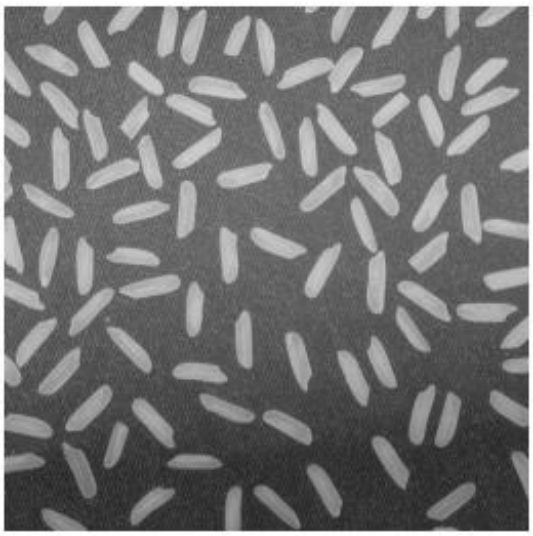
#### a) Add Two Images and Specify Output Class

1. Read two grayscale uint8 images into the workspace.
   ```
   I = imread('rice.png');
   J = imread('cameraman.tif');
   ```
2. Filter the image.
   ```
   K = imadd(I,J,'uint16');
   ```
3. Display the result.
   ```
   imshow(K,[])
   ```

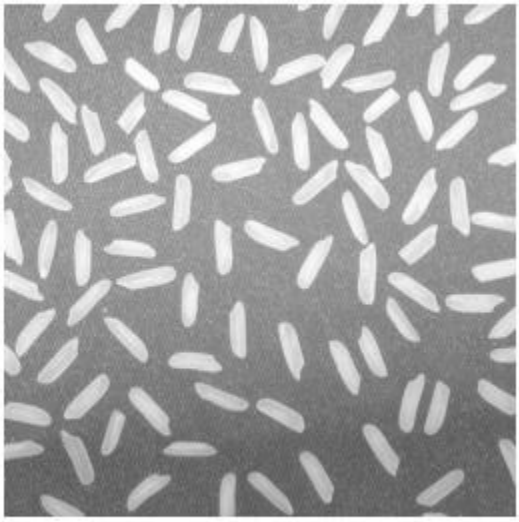
#### b) Add a Constant to an Image

1. Read an image into the workspace.
   ```
   I = imread('rice.png');
   ```
2. Add a constant to the image.
   ```
   J = imadd(I,50);
   ```
3. Display the original image and the result.

```
imshow(I)
```



```
figure
imshow(J)
```



## Experiment No. 26 imcomplement

Complement image

### Syntax

```
J = imcomplement(I)
```

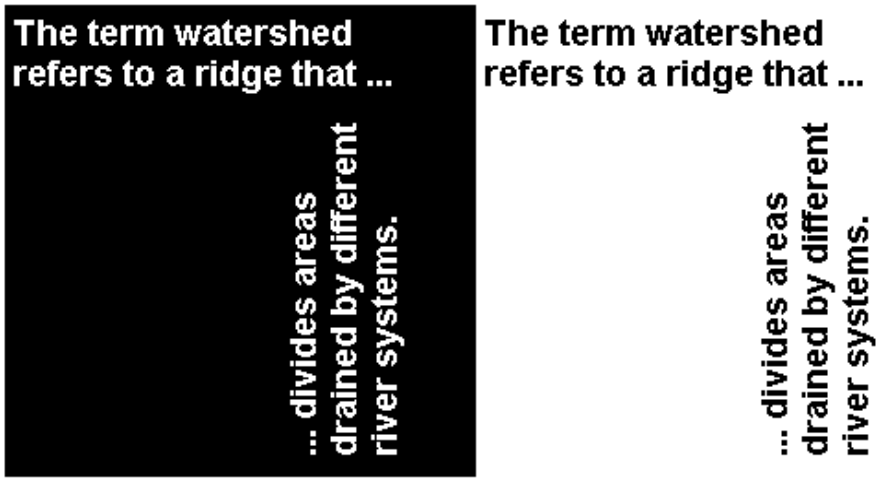
### Description

J = imcomplement(I) computes the complement of the image I and returns the result in J.

### Examples

**a) Reverse Black and White in a Binary Image**

```
bw = imread('text.png');
bw2 = imcomplement(bw);
imshowpair(bw,bw2,'montage')
```




**b) Create the Complement of an Intensity Image**

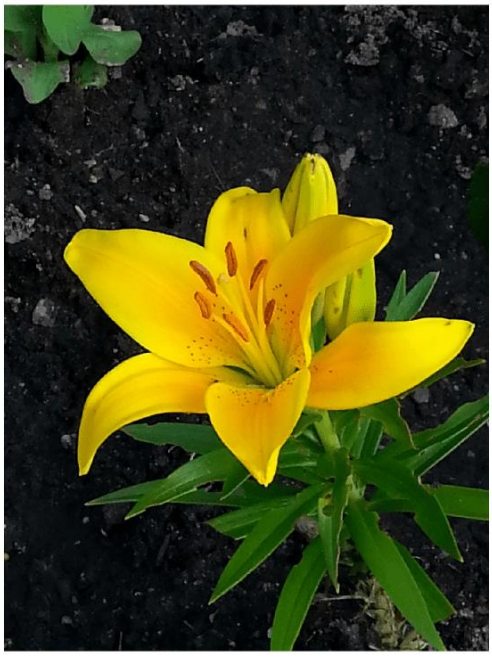
```
I = imread('cameraman.tif');
J = imcomplement(I);
imshowpair(I,J,'montage')
```

## c) Create the Complement of a Color Image

1. Read a color image into the workspace.

```
rgb = imread('yellowlily.jpg');

imshow(rgb)
```



2. Display the complement of the image.

```
c = imcomplement(rgb);
```

```
imshow(c)
```

## Experiment No. 27 imdivide

Divide one image into another or divide image by constant

### Syntax

```
Z = imdivide(X,Y)
```

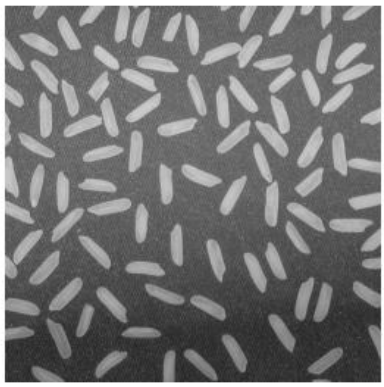
### Description

Z = imdivide(X,Y) divides each element in the array X by the corresponding element in array Y and returns the result in the corresponding element of the output array Z.

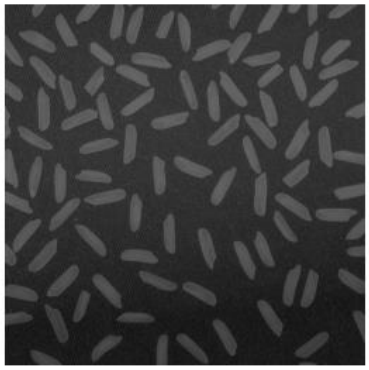
### Examples

#### a) Divide an Image by a Constant Factor

1. Read an image into the workspace.
   ```
   I = imread('rice.png');
   ```
2. Divide each value of the image by a constant factor of 2.
   ```
   J = imdivide(I,2);
   ```
3. Display the original image and the processed image.
   ```
   imshow(I)
   ```



```
figure
imshow(J)
```



## Experiment No. 28 imlincomb

**Examples**

### a) Scale an Image Using Linear Combinations

1. Read an image into the workspace.
   ```
   I = imread('cameraman.tif');
   ```
2. Scale the image using a coefficient of 1.5 in the linear combination.
   ```
   J = imlincomb(1.5,I);
   ```
3. Display the original image and the processed image.
   ```
   imshow(I)
   ```

```
figure
imshow(J)
```

## b) Add Two Images and Specify Output Class Using Linear Combinations

1. Read two grayscale uint8 images into the workspace.
   ```
   I = imread('rice.png');
   J = imread('cameraman.tif');
   ```
2. Add the images using a linear combination. Specify the output as type uint16 to avoid truncating the result.
   ```
   K = imlincomb(1,I,1,J,'uint16');
   ```
3. Display the result.
   ```
   imshow(K,[])
   ```

## Experiment No. 29 immultiply

Multiply two images or multiply image by constant

### Syntax

```
Z = immultiply(X,Y)
```

### Description

Z = immultiply(X,Y) multiplies each element in array X by the corresponding element in array Y and returns the product in the corresponding element of the output array Z.

### Examples

#### a) Multiply an Image by Itself

1. Read a grayscale image into the workspace, then convert the image to uint8.
   ```
   I = imread('moon.tif');
   I16 = uint16(I);
   ```
2. Multiply the image by itself. Note that immultiply converts the class of the image from uint8 to uint16 before performing the multiplication to avoid truncating the results.
   ```
   J = immultiply(I16,I16);
   ```
3. Show the original image and the processed image.
   ```
   imshow(I)
   ```

```
figure
imshow(J)
```

**b) Scale an Image by a Constant Factor**

1. Read an image into the workspace.
   ```
   I = imread('moon.tif');
   ```
2. Scale each value of the image by a constant factor of 0.5.
   ```
   J = immultiply(I,0.5);
   ```
3. Display the original image and the processed image.
   ```
   imshow(I)
   ```

```
figure
```

```
imshow(J)
```

## Experiment No. 30 imsubtract

Subtract one image from another or subtract constant from image

### Syntax

```
Z = imsubtract(X,Y)
```

### Description

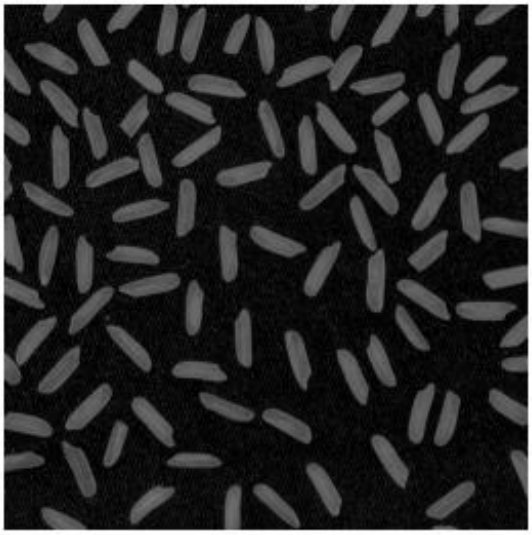
Z = imsubtract(X,Y) subtracts each element in array Y from the corresponding element in array X and returns the difference in the corresponding element of the output array Z.

### Examples

**a) Subtract Image Background**

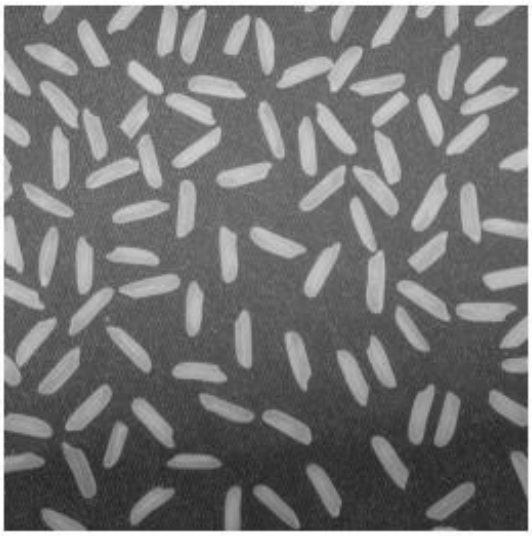
1. Read a grayscale image into the workspace.
   ```
   I = imread('rice.png');
   ```
2. Estimate the background.
   ```
   background = imopen(I,strel('disk',15));
   ```
3. Subtract the background from the image.
   ```
   J = imsubtract(I,background);
   ```
4. Display the original image and the processed image.
   ```
   imshow(I)
   ```

```
figure
imshow(J)
```



**b) Subtract a Constant from an Image**

1. Read an image into the workspace.
   ```
   I = imread('rice.png');
   ```
2. Subtract a constant value from the image.
   ```
   J = imsubtract(I,50);
   ```
3. Display the original image and the result.
   ```
   imshow(I)
   ```



```
figure
```

```
imshow(J)
```