# EGR 111
# Image Processing

This lab shows how MATLAB can represent and manipulate images.

New MATLAB Commands: imread, imshow, imresize, rgb2gray
Resources (available on course website): secret_image.bmp

## 1. Loading and Displaying Grayscale Images

A grayscale image (also called a black and white image) is represented in MATLAB by a two-dimensional matrix. Each element in the matrix indicates the brightness of a pixel in the image. (Pixel is short for picture element.) The higher the pixel value, the brighter or whiter the pixel appears in the image. Many images use 8-bit numbers to represent the pixel brightness, in which case the pixel values range from 0 (black) to 255 (white), and numbers in between appear as shades of gray.

MATLAB includes several sample images. In this section we will load and display one of these files called moon.tif.

1. Load the image file moon.tif, which is installed with MATLAB, using the following command:
```
I = imread('moon.tif');
```

2. Display the image:
```
figure(1), imshow(I)
title('Original image')
```

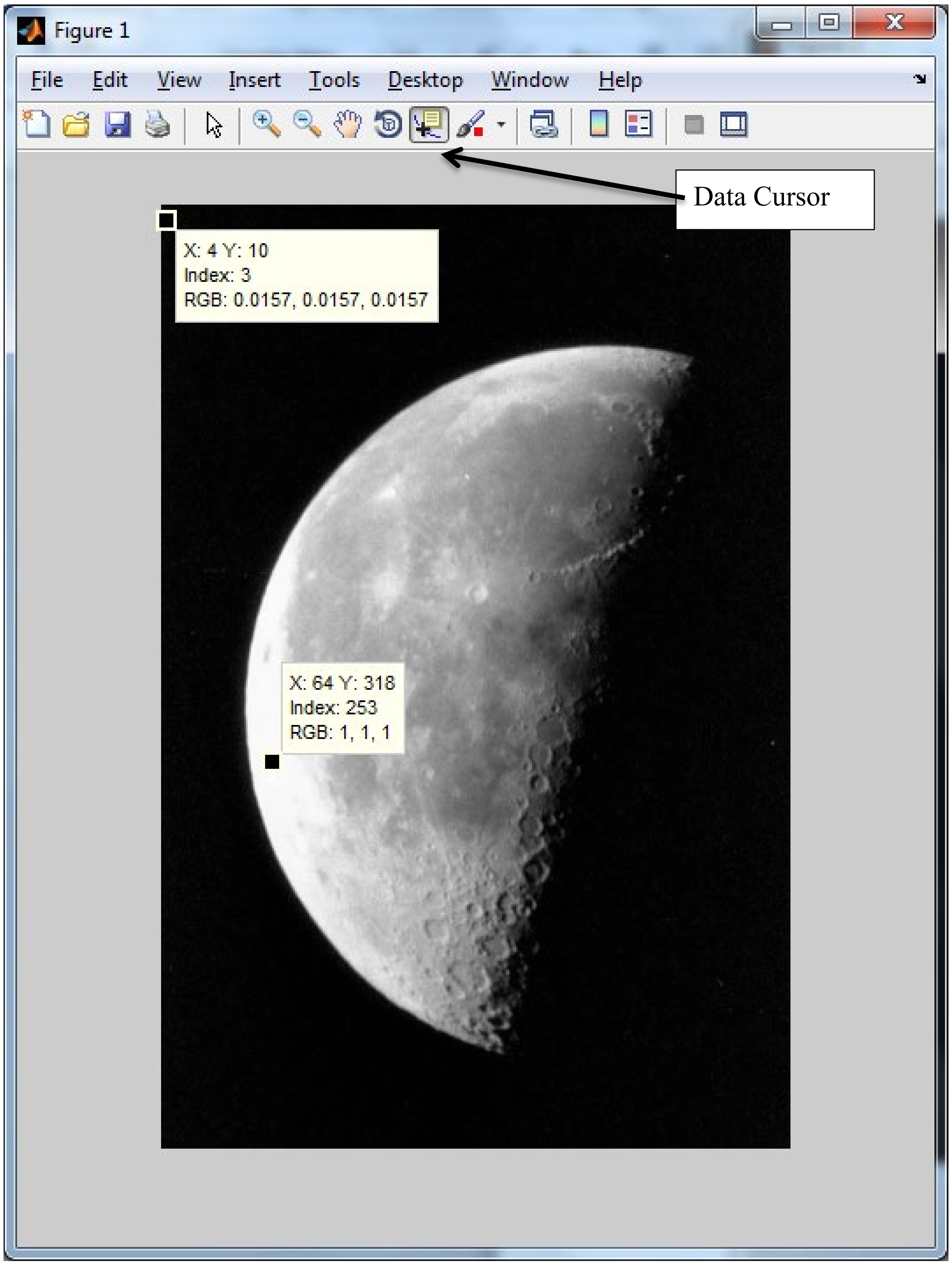
3. Find the size of the image by typing:
```
size(I)
```

MATLAB should display two numbers like: ans = 537   358. The first number is the number of rows and the second number is the number of columns.

4. View some of the image values from the upper-left hand corner of the image:
```
I(1:10,1:10)
```

Note that these values are small (compared to 255), which indicates that the pixels in the upper left-hand corner are dark.

5. Click on the Data Cursor button in the figure window, and click on a dark area in the image (see the figure below). The value of "Index" is the value of the pixel (0 for black, 255 for white). The value of "X" is the x-axis value (which is the column number in the matrix) and the value of "Y" is the y-axis value (which is the row number of the pixel in the matrix).



X: 4 Y: 10
Index: 3
RGB: 0.0157, 0.0157, 0.0157

X: 64 Y: 318
Index: 253
RGB: 1, 1, 1

## 2. Manipulating Grayscale Images

MATLAB makes it easy to manipulate the image. First let's make a copy of the image.
```
I2 = I;
```

We can access the value of the pixel in the 10th row and 4th column by typing the following.
```
I2(10,4)
```

The value of the pixel in the 10th row and 4th column should be 3, which is the same as the Index value shown in the Data Cursor in the figure (see the figure above).

We can change the pixel in the 10th row and 4th column to white by typing the following:
```
I2(10,4) = 255;
figure(2)
imshow(I2)
```

You should now see a white dot in the upper left hand corner of the image. You may need to zoom in to see it.

Note that the image in the figure window is updated only when you type the `imshow` command, not when you change the values in the matrix.

We can make a horizontal white line by replacing an entire row with white pixels.
```
I2(100,:) = 255;
figure(2)
imshow(I2)
```

Recall that when using the colon operator to index matrices, the colon by itself means all of the rows or all of the columns.

We can brighten the image by adding a constant to every element:
```
I2 = I + 100;
figure(2)
imshow(I2)
```

Compare the two images (original and brightened) side-by-side.

Invert the image (convert light areas to dark and vice versa) as follows:
```
I2 = 255 - I;
figure(2)
imshow(I2)
```

Two images can be superimposed by adding the matrices as follows.
```
I2 = imread('liftingbody.png');
I2 = imresize(I2, size(I));
I3 = I + I2;
figure(3)
imshow(I3)
```

A color image can be converted to grayscale as follows.
```
I4 = imread('onion.png');
figure(4)
imshow(I4)
I5 = rgb2gray(I4);
figure(5)
imshow(I5)
```

**Exercise 1:** Load an image of your choice into MATLAB, make some changes to it, and display the original image and the changed image. You can use your own image, or if you want, you can use one of the MATLAB sample images listed below.

bag.png
coins.png
concordorthophoto.png
liftingbody.png
rice.png
testpat1.png
westconcordorthophoto.png
AT3_1m4_01.tif
cameraman.tif
cell.tif
circuit.tif
eight.tif
forest.tif
kids.tif
moon.tif
mri.tif
pout.tif
shadow.tif
spine.tif
tire.tif
trees.tif

***Checkpoint 1:*** Show your instructor the original and modified images from Exercise 1.

**Exercise 2:** Download the file secret_image.bmp from the course website and display the image. This image has been purposely modified in an attempt to conceal a hidden image. First the pixel values were all divided by 10, so the range of the data was reduced to the range 0 to 26. Then the value 200 was added to about half of the pixels which were selected at random. This processing covers the original image with random noise, however, it is possible to remove the processing and recover the original image. The pixels that were randomly selected will have values greater than 26, so in order to recover the original image, a program needs to check each pixel value, and for those pixels that have a value greater than 26, the program should subtract 200. Then the program should multiply all pixel values by 10. Write a script file that loads secret_image.bmp, recovers the original image, and displays the result.

***Checkpoint 2:*** Show your instructor your script file and the recovered image from Exercise 2.

---

**OPTIONAL Exercises**. The optional exercises below show how to manipulate color images.

MATLAB can also load and display color images. Type the following commands to load the image file onion.png which is a MATLAB example image:

```
I=imread('onion.png');
imshow(I);
```

A color image is composed of three separate images, one for the intensity of red, one for green, and one for blue. You can see that MATLAB represents this color image by a three dimensional matrix by typing the following command:

```
size(I)
```

The size of this image is 135 by 198 by 3. There are 135 rows, 198 columns, and 3 colors (red, green, and blue).

You can display each of the 3 images separately using the following commands:

```
figure; imshow(I(:,:,1)), title('Red')
figure, imshow(I(:,:,2)), title('Green')
figure, imshow(I(:,:,3)); title('Blue')
```

Note that the white onion appears fairly bright in all three images because white is composed of red, green, and blue. The red pepper appears fairly bright in the red image, but dark in the green and blue images. The yellow pepper appears bright in the red and green images because yellow is made up of a mixture of red and green.

You can generate images with various colors by making matrices with different values for each of the three images. For example, to make an 8-bit image with 100 by 100 pixels where all the pixels are black, you set all the values to zero as follows:

```
I = uint8(zeros(100,100,3));
imshow(I)
```

You can make an image with all red pixels by setting the first color value to the maximum value of 255 as follows:

```
I = uint8(zeros(100,100,3));
I(:,:,1) = 255;
imshow(I)
```

The following commands generate an image with all green pixels by setting the second color value to the maximum value of 255 as follows:

```
I = uint8(zeros(100,100,3));
I(:,:,2) = 255;
imshow(I)
```

And a blue image can be generated as follows:

```
I = uint8(zeros(100,100,3));
I(:,:,3) = 255;
imshow(I)
```

You can add different amounts of the primary colors (red, green, and blue) to get other colors. For example, it you mix red and green you get yellow.

```
I = uint8(zeros(100,100,3));
I(:,:,1) = 255;
I(:,:,2) = 255;
imshow(I)
```

Make an image where all three primary colors are added together (red, green, and blue). What is the resulting color?

Load the image file onion.png as shown above and make a yellow stripe across the image.

One way to change the colors in an image is to swap the images from two of the colors. For example, the commands below swap the red and blue images.

```
I=imread('onion.png');
tmp = I(:,:,1);  % save red image in a temporary variable
I(:,:,1) = I(:,:,3);  % put blue image into red
I(:,:,3) = tmp;  % put red image into blue
figure(2),imshow(I)
```

**Optional Exercise 3:** Write a script file to load an image of your choice, make modifications to the image to make it more interesting, and display the result.