MATLAB LAB: DIGITAL IMAGE PROCESSING

Introduction

In this Tutorial, we'll scan through the key features/functions of image processing from A to Z. It won't be a comprehensive but a very short while we can grasp what's going on Matlab's image processing.

We'll use the following basic image processing functions:

1.imread()

2.imshow()

3.imwrite()

4.rgb2gray()

5.imhist()

6.imadjust()

7.im2bw()

imread()

The **imread()** command will read an image into a matrix:

```
img = imread('ImageProcessing_1/BerkeleyTower.png');
>> size(img)


ans =
   499   748     3
```

It's a 499x748 matrix with 3 RGB channels. The matrix looks like this:

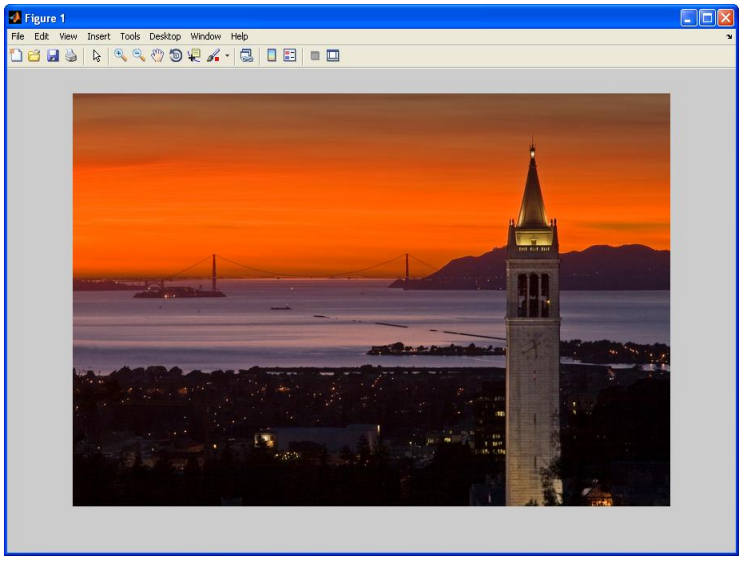
```
   24   40   73  108  129  108   96  100  109  114  108  109
   29   56   97  107  110  104  103  105  106  110  110  111
   ...
    3    2    2    1    0    0    1    1    0    1    2    4
    1    0    1    3    2    0    0    0    1    1    2    1
   ...
```

Input file : Any Image

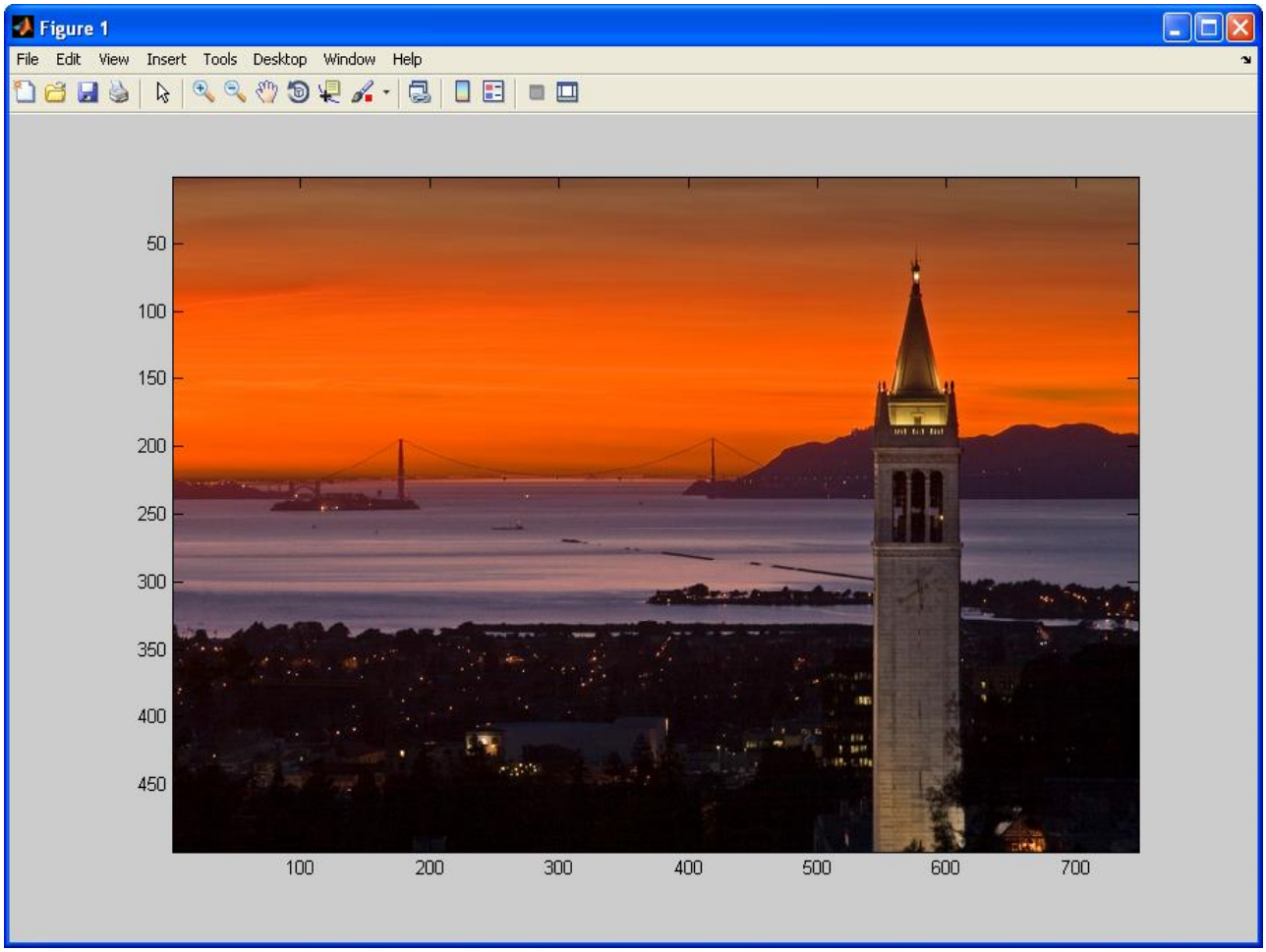
imshow()

To show our image, we the **imshow()** or **imagesc()** command. The **imshow()** command shows an image in standard format, like it would appear in a web browser. The **imagesc()** command displays the image on scaled axes with the min value as black and the max value as white.

```
imshow(img)
```

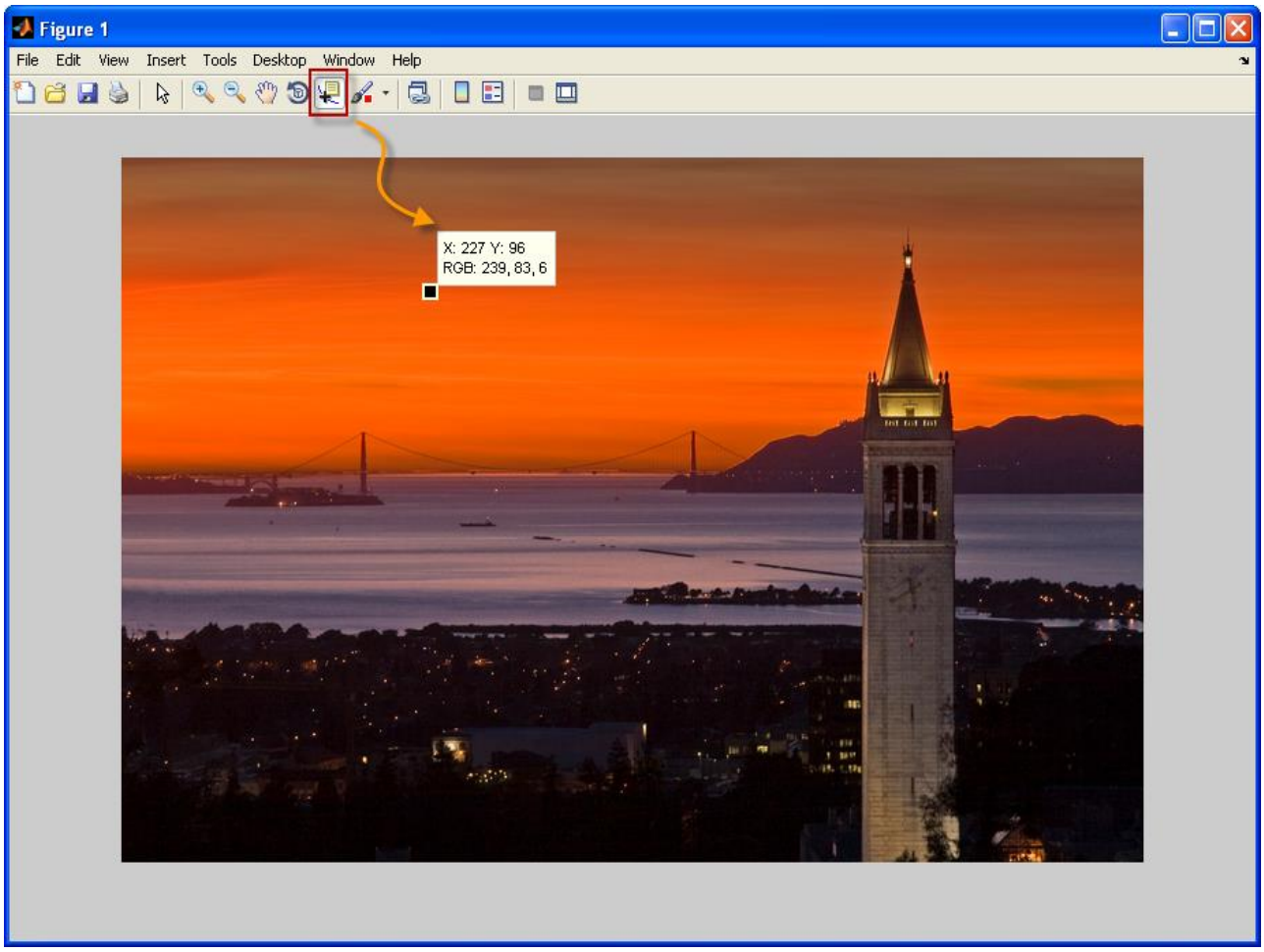


Using **imagesc()**:



We can check the RGB values with (x,y) coordinates of a pixel:

1. Select "Data Cursor" icon from the top menu
2. Click the mouse on the image



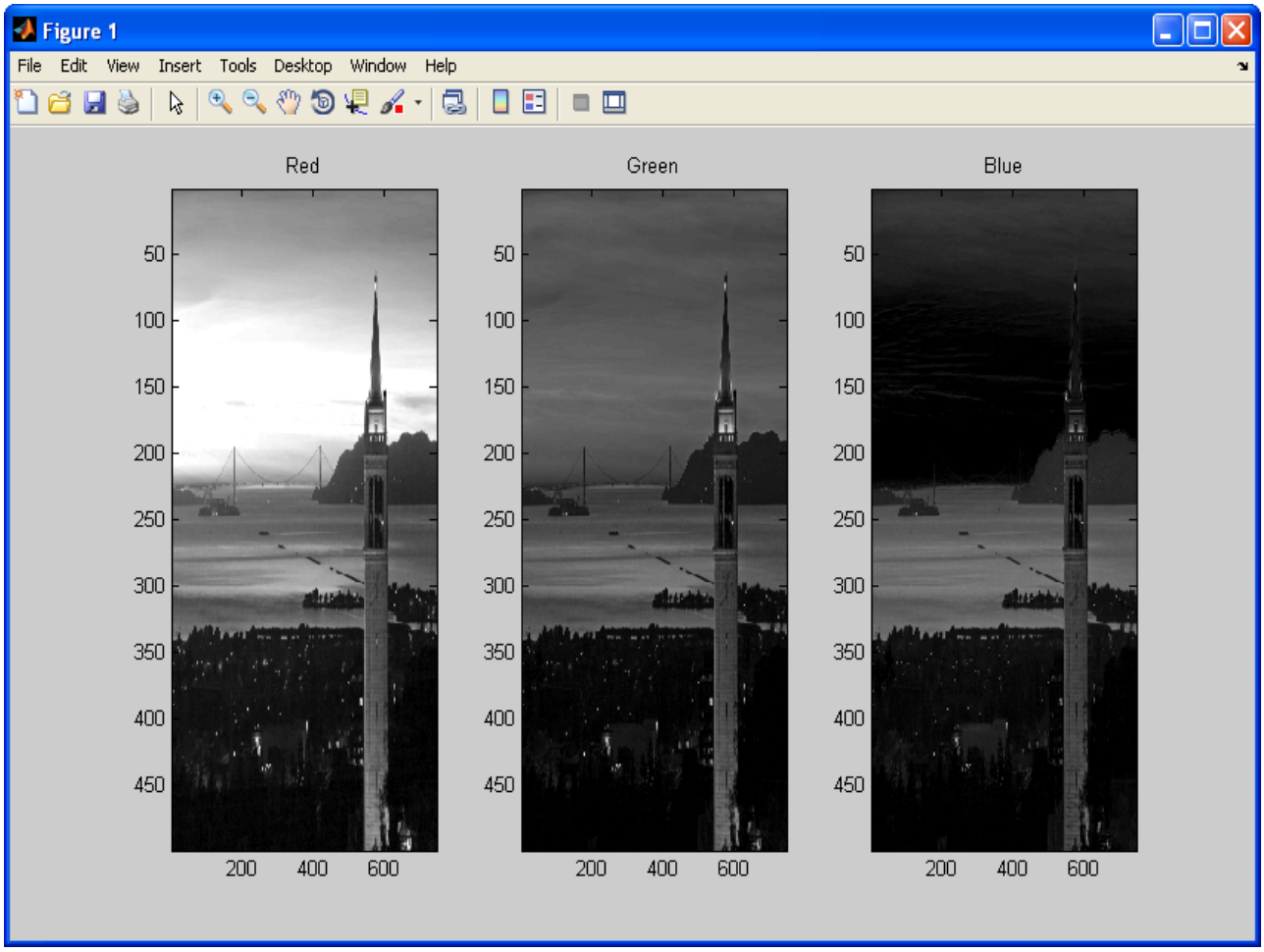
Notice each pixel is a 3-dimensional vector with values in the range [0,255]. The 3 numbers displayed is the amount of RGB.

Actually, a color image is a combined image of 3 grayscale images. So, we can display the individual RGB components of the image using the following script:

```
subplot(131);
imagesc(img(:,:,1));
title('Red');

subplot(132);
imagesc(img(:,:,2));
title('Green');
```

```
subplot(133);
imagesc(img(:,:,3));
title('Blue');
```



Want to play more? Then, let's make a new image that has more blue by quadrupling the component. The standard image format is uint8 (8-bit integer), which may not like arithmetic operation and could give us errors if we try mathematical operations. So, we may want to do image processing by casting our image matrix to **double** format before we do our math. Then we cast back to uint8 format at the end, before displaying the image.

```
subplot(211);
imshow(img);
```

```
title('Normal RGB');


subplot(212);
blue_img = double(img);
blue_img(:,:,3) = 4*blue_img(:,:,3);
blue_img = uint8(blue_img);
imshow(blue_img);
title('RG 4*B');
```

imwrite()

To save your new blue image, use the **imwrite()** command:

```
imwrite(blue_img, 'Blue4_BerkeleyTower.png', 'png');
```

rgb2gray()

```
img = imread('ImageProcessing_1/BerkeleyTower.png');
gray = rgb2gray(img);
imshow(gray);
```

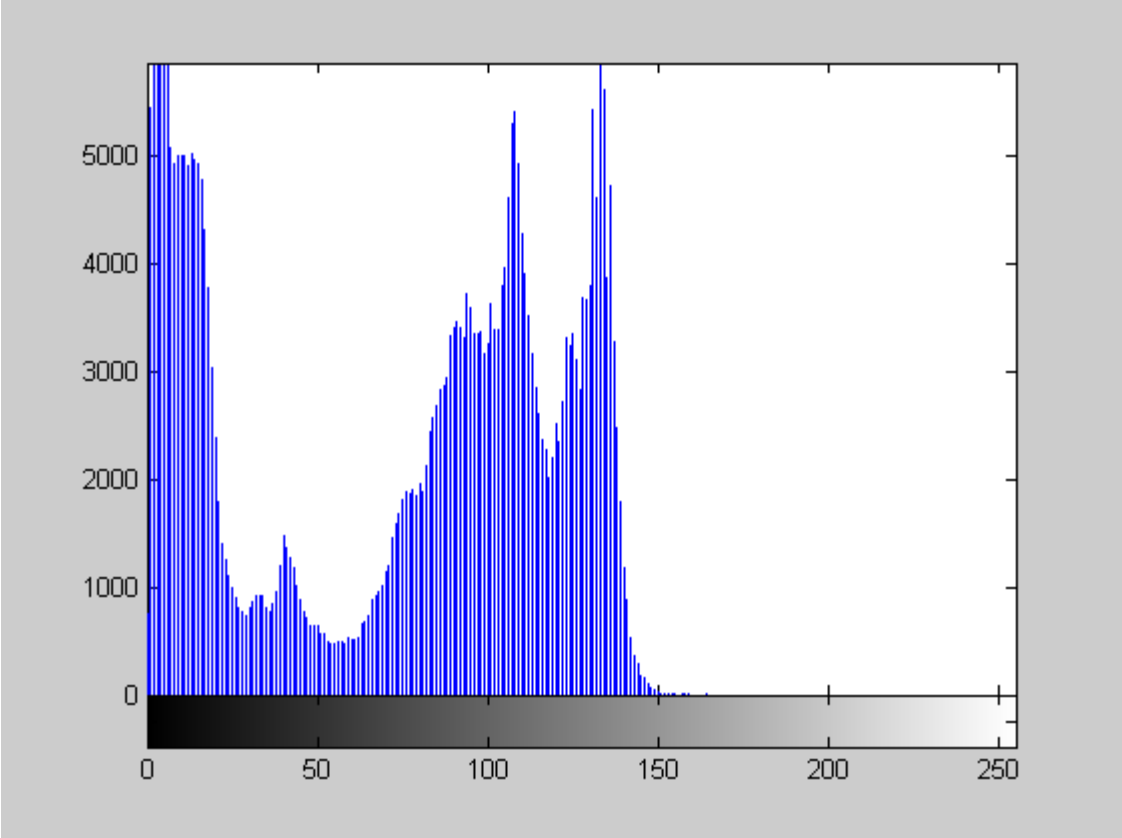
```
>> size(gray)
ans =
   499   748
```

Now, we have only one value for each pixel not 3 values. In other words, the **rgb2gray()** converted RGB images to grayscale by eliminating the hue and saturation information while retaining the luminance.

imhist()

Display a histogram of image data. **imhist()** uses a default value of n = 256 if I is a grayscale image, or n = 2 if I is a binary image.

```
img = imread('ImageProcessing_1/BerkeleyTower.png');
gray = rgb2gray(img);
imhist(gray);
```



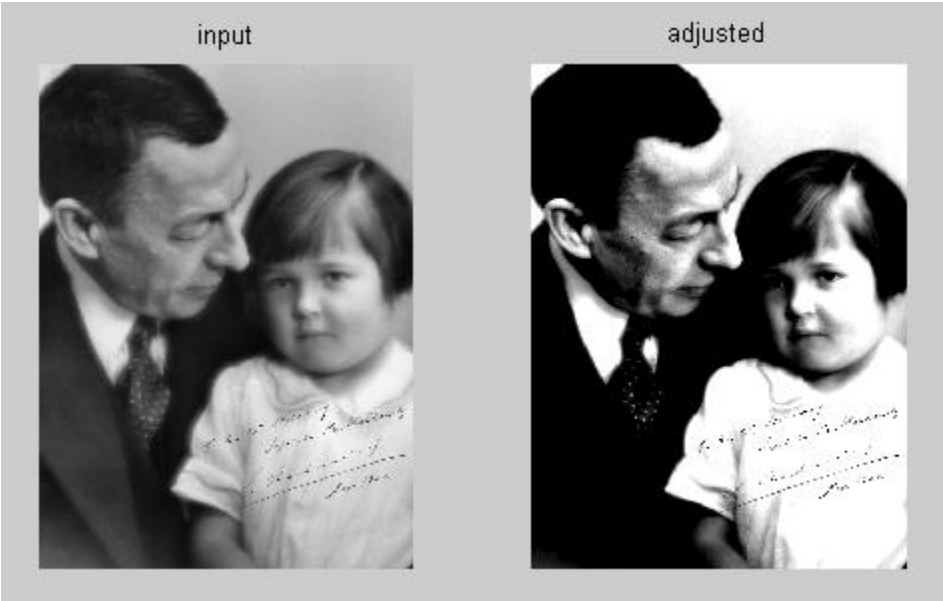
imadjust()

imadjust() adjust image intensity values. It maps the values in intensity image of an input to new values in output image. This increases the contrast of the output image.

```
img = imread('ImageProcessing_1/Rachmaninoff.jpg');
gray = rgb2gray(img);
adj_img = imadjust(gray, [0.3,0.7],[]);

subplot(121);
imshow(gray);
title('input');

subplot(122);
imshow(adj_img);
title('adjusted');
```



Input file : Rachmaninoff.jpg.

```
J = imadjust(I)

J = imadjust(I,[low_in high_in])

J = imadjust(I,[low_in high_in],[low_out high_out])
```

stretchlim(I)

computes the lower and upper limits that can be used for contrast stretching grayscale or RGB image I. The limits are returned in low high.

- I = imread('pout.tif');
- figure
- imshow(I)
- J = imadjust(I,stretchlim(I),[]);
- figure
- imshow(J)

im2bw()

im2bw() converts the grayscale image to a binary image. We'll use the adjusted image.

```
img = imread('ImageProcessing_1/Rachmaninoff.jpg');
gray = rgb2gray(img);
adj_img = imadjust(gray, [0.3,0.7],[]);
bw_img = im2bw(adj_img);

subplot(121);
imshow(adj_img);
title('input image');

subplot(122);
imshow(bw_img);
title('binary image');
```