Tutorial 2: Image Basics

If we consider an image as a two-dimensional function $f_a(x,y)$ where, for every co-ordinate position *(x,y)* exist a function value, then, a digital image $I = f(x,y)$ corresponds to a two-dimensional discrete function. This digital image can be analysed as a two-dimensional matrix in which every element of the matrix is considered as a picture element or *pixel* (sometimes called *pel*). A typical size of an image is a square of 2^n x 2^n pixels: 128 x 128, 256 x 256, 512 x 512 pixels. In the general case an image can be of size m-by-n, m pixels in the vertical direction and n pixels in the horizontal direction. The value of the pixel can represent light intensity, attenuation, depth information, or intensity of radio wave depending on the type of information contained by the image. This intensity level is the essential information to perform any operation like segmentation in most of the techniques.

Colour images contain important information about the perceptual phenomenon of colour related to the different wavelengths of visible electromagnetic spectrum. In many cases, the information is divided into three primary components *Red, Green* and *Blue* or psychological qualities *hue*, *saturation* and *intensity*. Prior knowledge of the objects colour can lead to classification of pixels but this is not always known.

The intensity level describes how bright or dark the pixel at the corresponding position should be coloured. There are two ways to represent the number that represents the brightness of the pixel: The double class (or data type). This assigns a floating number between 0 and 1 to each pixel. The value 0 corresponds to black and the value 1 corresponds to white. The other class is called *uint8*, which assigns an integer between 0 and 255 to represent the brightness of a pixel. The value 0 corresponds to black and 255 to white. The class *uint8* only requires roughly 1/8 of the storage compared to the class *double*. On the other hand, many mathematical functions can only be applied to the doubles. Matlab supports the following image formats: BMP, HDF, JPEG, PCX, TIFF, XWB, PNG.

To read and display images use *imread* and *imshow*:

```
image1 = imread ('pout.tif');  %Reads an image into a variable image1
image2 = imread ('moon.tif');
image3 = imread ('tire.tif');
image4 = imread ('mri.tif');
image1 = imread ('pout.tif');
image5 = imread ('flowers.jpg');
figure(1)
imshow(image1);                % Displays the image
figure(2)
subplot(2,2,1); imshow(image1);
subplot(2,2,2); imshow(image2);
subplot(2,2,3); imshow(image3);
subplot(2,2,4); imshow(image4);

figure(1); clf;
subplot(3,1,1)
imshow(image5(:,:,1))                % Displays Red component
subplot(3,1,2);imshow(image5(:,:,2)) % Displays Green component
subplot(3,1,3);imshow(image5(:,:,3)) % Displays Blue component
```

2.1 Can you notice the differences of the colour components?

2.2 Try this now:

```
figure(1); imshow(image1(1:100,1:100,:));
figure(1); imshow(image1(100:end,100:end,:));
figure(1); imshow(image1(1:4,1:4,:));
image1(1:4,1:4,:)
```

This is an easy way of cropping and investigating the individual values of the pixels.

The images are located in the directory /package/matlabr12/toolbox/images/imdemos. Notice the differences of size and colour of the images. The images cannot be manipulated when their type is *uint8*, they have to be converted to *doubles*:

```
image4 = double(image4);        % Converts image to a matrix of doubles
image5 = randn(128,128);        % Creates a 128 x 128 normally
                                % distributed image (Gaussian noise)
```

2.3 Now try:

```
surf(image4); shading flat;
colormap(gray);
colormap(jet);
rotate3d on;
```

2.4 Use the mouse to rotate the image and change the viewpoint. This is equivalent to use the *view* command. Change the axis of the image with the *axis* command. Repeat the process for other images.

Once the images are in a double type, they are treated and handled as matrices so that any operations; addition, subtraction, multiplication, inversion, transposition, ... are valid, as long as they follow matrix operations.

2.5 The functions *max, min, mean, std, sort* are very useful when you are interested in the statistics of an image. Find out more of them with the command *help* and apply them to the previous images. What is the response for m x 1, or 1 x n matrices? What happens when the size is m x n? *mean2* and *std2* can be useful in those cases.