Digital Image Processing - Lab Session 1
UPEC - Optics, Image, Vision and Multimedia

Farnaz Majidzadeh

1. Image Processing and Analysis: getting started

1.1 Download the single channel image *'crossroad.dat'* using the command read function in Python. The data consist of unsigned integers 8 bits in size, so use *uint8* type.

1.2 Is it possible to display the image using *imshow* or any other showing image function in Python? What information is missing? Then assume that the image is 435 lines long and use the function *reshape* to organize data in memory.

1.3 Here is the interest of image file format, isn't it easier using the file *'crossroad.bmp'*? Use the command *read* for *('crossroad.bmp')* directly.

1.4 See the gray levels in the workspace and analyze the first pixel. Row? Column? Value?

1.5 Display the value of this first pixel in the command window. And the bottom right corner pixel? Use the *size* function.

1.6 Create a vector L1 as a copy of the first row of this image. Then a vector C1 for the first column. Display them using *bar* or *plot*. Choosing some columns of *'sonnet'*, what can we conclude about the acquisition step?

1.7 Display the entire image.

1.8 Store this image with different formats (png, tif, jpg).

2. Resolution

2.1 Using *'crossroad'* image, change the resolution with boxes 2, 4, 8 and 16 pixels on a side and observe the results. What features do you lose at each resolution size?
i) Using basic subsampling method, where only one pixel over 2 for example is stored each line and column, see the colon *(:)* operator,
ii) Using *resize* function.

2.2 Same questions with images *'test pattern'* and *'patterns'*.

3. Quantization

3.1 Considering the previous images with data of type uint8, the data consists of eight bits for each pixel. The gray levels are from 0 to 255, ie 256 classes. Create new images with 128, 64, 32, 16, 8, 4, and 2 gray levels. If necessary, *dec2bin* convert decimal to binary number in string, *bitget* get bit at specified position and *bitset* set bit at specific location while *bitshift* shift bits specified number of places.

3.2 How many bits are needed to preserve image quality? Does it change from place to place in the image? How so?

4. Indexed color

4.1 Open and display the single channel image *'chro'* without any map.

4.2 Use its own, or *'jet'*, *'hsv'*, *'hot'* and *'lines'* color maps. Explain and comment. According to you, discuss about the context where these maps can be used.

4.3 Repeat question1 with the image *'glucose'*. Use the function *ind2rgb* to store the image as a RGB one. Store this image with and without compression and compare the different sizes.

4.4 Display the three channels image *'spectrum'*. Transform this image to a single channel one using the color map *'map.txt'* and *rgb2ind*. Comments.

4.5 Using again this map, transform the RGB image *'umbrella'* in a single channel image. Comments.