Exercise Chapter 2 – Basic tools for image processing - LUT

This exercise is mainly an observation exercise (there are few programming jobs) during which you will be not only familiarized with the use of Matlab for image processing (which will help you for the next exercises) but also with the use of basic image processing tools.

Launch Matlab and update the path list in the path browser.

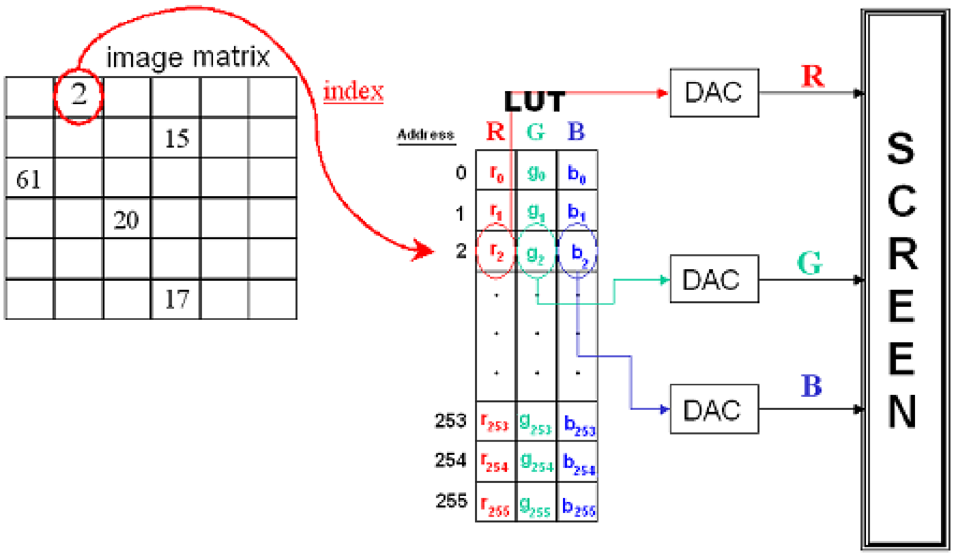
To build a Look-up Table (*LUT)*

1 – Starting session
From the Matlab command window, open a new file "M-File" in which you will type your commands and of which each line (finishing with “;”) will be then interpreted.

2 – Open an image
Start by loading the grayscale image *CLOWN_LUMI.BMP* with ***imread*** (take a look at the Matlab help). Observe the type of the data.

3 – Display and conversion
The image display can be done with the commands ***image***, ***imagesc*** et ***imshow***. Observe the differences; also carry out a test by converting your data (cf. Exercise « *Introduction to Matlab* » of chapter 1: try *image*=***double***(*image*)).

4 – Examples of LUT
To display a grayscale image (2D array) Matlab uses a default LUT, which associates each element of the array with a color. This default LUT has 64 different output colors (use the command ***colormap*** to display the colors of the default LUT).



A color corresponds to each index “i”. This color is created by the combination $\{r_i, g_i, b_i\}$ of primary colors Red-Green-Blue. The three values $\{r_i, g_i, b_i\}$ are sent to three “digital-to-analog converters (DAC)” to modulate the screen Red-Green-Blue electron guns. By assigning a different image band to each gun, we can create a false color composite image to aid in visual interpretation.

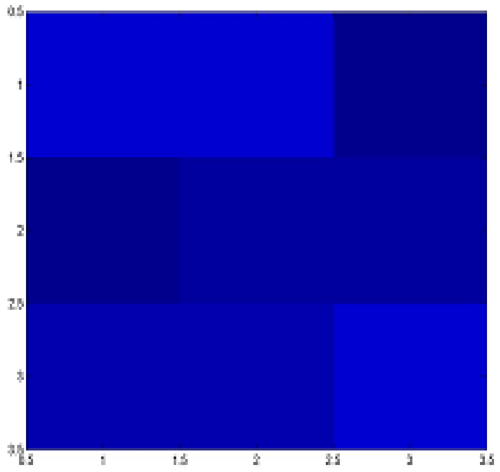
Matlab allows you to build a specific LUT and to use it with the ***colormap*** function. Colormap must have values in [0,1]. Let us consider, for example, the simple case of a 3×3 matrix M:

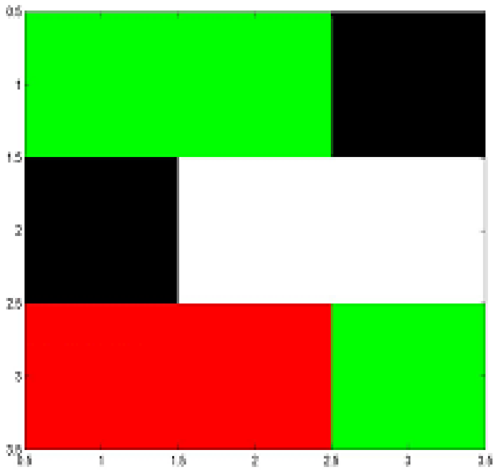
$$M = \begin{bmatrix} 5 & 5 & 1 \\ 1 & 2 & 2 \\ 3 & 3 & 5 \end{bmatrix}$$

This matrix has 4 distinct values. We want to visualize: “1” in black, “2” in white, “3” in red, and “5” in green. To do this, we build a LUT called “map4C”: the LUT outputs are the 4 required colors. Type the command ***colormap(map4C)*** to apply this LUT:

```
M = [5 5 1;1 2 2;3 3 5] ;
% LUT
r = [0 1 1 0];
g = [0 1 0 1];
b = [0 1 0 0];
map4C = [r’ g’ b’];
image(M)
colormap(map4C)
```

Results before and after the use of the command ***colormap( )***:

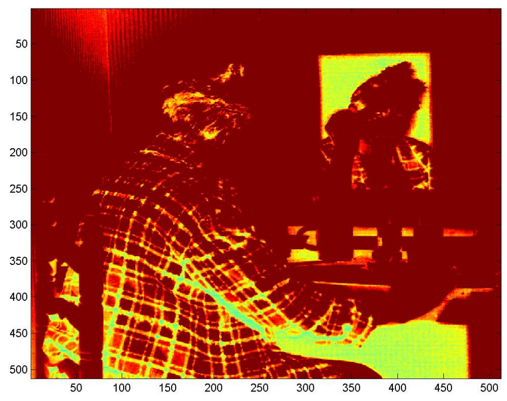
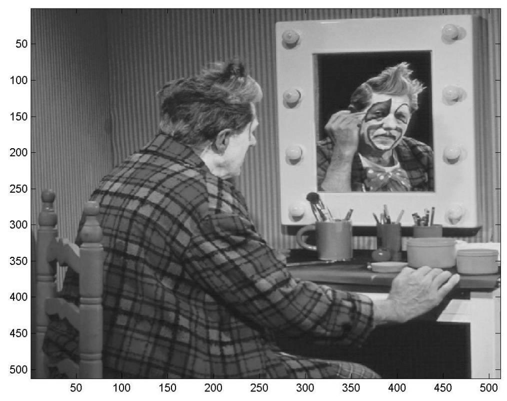




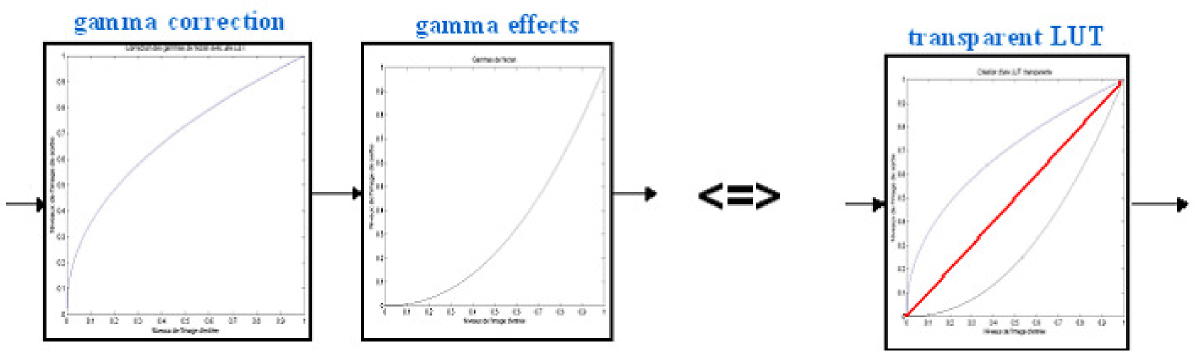
To display an achromatic image in gray levels you must use a LUT so that all output colors are gray scales: $\forall i \in [0, 255], r_i = v_i = b_i$ (typically for a 8-bit pixel coding there are 256 gray levels)

ACTION: Load the script ***lutndg.m*** in your working folder. Open and analyze this script. From this script, visualize your achromatic image in gray levels with the command ***lutndg.m***.

Here are the images before and after using the LUT of the script ***lutndg.m***:

Let us consider another example of LUT creation from Matlab: a screen generates “gamma effects” which can be compensated (non-linear transformation of the input image gray levels). To compensate the gamma effects we create a LUT which is the inversion of the gamma effects so that the concatenation “LUT o gamma effects” is a identity transformation.

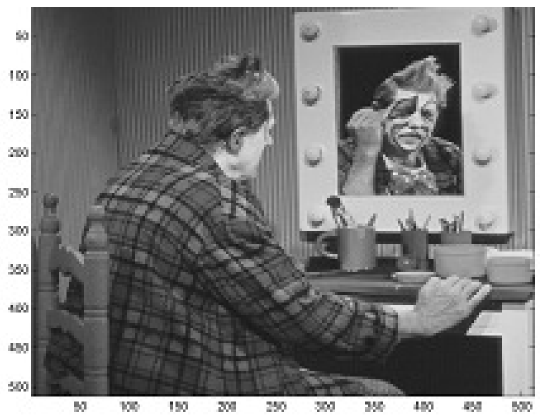


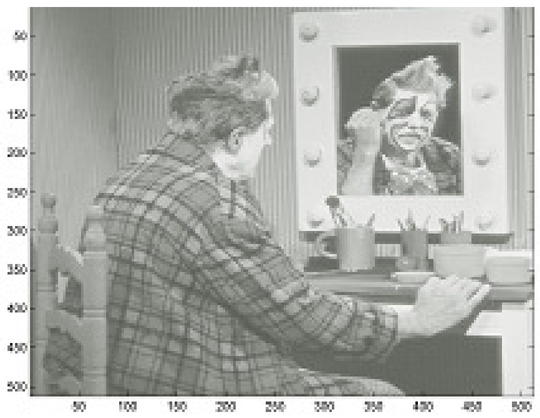
Here is an example of a script to compare the images before and after the gamma compensation:

```
I = imread(‘CLOWN_LUMI.BMP’);
% LUT to display image clown_lumi in gray levels
r=0:1/255:1;
g=0:1/255:1;
b=0:1/255:1;
map=[r’ g’ b’];
image (I);
colormap(map)
% LUT to compensate gamma effects
vect = 0:1/255:1;
gamma_r = 2.2;        % output=input^2.2 (red)
gamma_g = 2.3;        % output=input^2.3 (green)
gamma_b = 2.1;        % output=input^2.1 (blue)
r = vect.^(1/gamma_r);
g = vect.^(1/gamma_g);
b = vect.^(1/gamma_b);
```

```
% we build the LUT with these 3 vectors
map_gamma_inv = [r' g' b'];
% to apply the LUT
figure
image(I)
colormap(map_gamma_inv);
```

Here are the results before (image on the left) and after (image on the right) using the LUT to compensate the gamma effects:





ACTION: From the preceding examples, synthesize a LUT to make the video inversion of the image *CLOWN_LUMI*. This image processing consists of carrying out the inversion (well-known in photography) of each color plane i.e. levels 0 become 255 and conversely, levels 1 become 254, etc.

5 – To display the R-G-B components

Load the color image *CLOWN* in your working folder. Observe the type of the data and visualize the image. Visualize the color image *CLOWN* plane by plane by creating adequate LUTs for the planes Red, Green, and Blue (base this on the exercise "Breaking up a color image" from chapter 1).

Exercise solution: Look-up Table

1 – Open a script editor using the method of your choice (cf. exercise "Matlab initiation")

2 – In your working folder type the command:

```
            I = imread('CLOWN_LUMI.BMP');
```

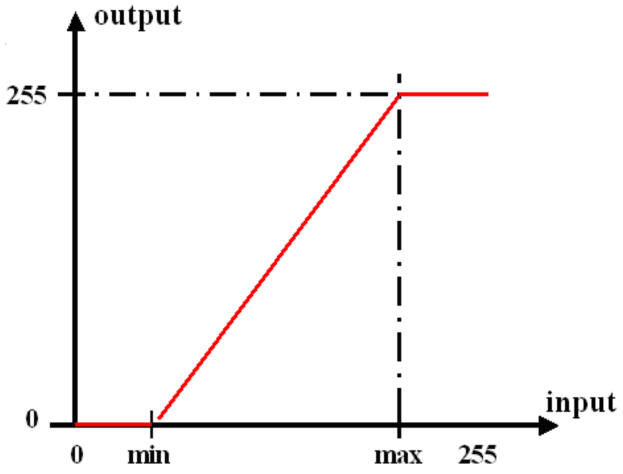
I is a 2D array for a grayscale image. The data type of I is uint8 (unsigned integer).

3 – There are three Matlab function for displaying images: ***imshow, image***, and ***imagesc***.

+ ***imshow*** displays the image stored in the graphics file or the corresponding matrix. ***imshow*** calls ***imread*** to read the image from the file, but the image data is not stored in the Matlab workspace. The file must be in the current directory or on the MATLAB path.

+ ***image*** displays the matrix in parameter as an image. Each element of the matrix specifies the color of a rectangular segment in the image. This function does not read the image from the file

+ ***imagesc*** is the same as ***image*** except that it scales image data to the full range of the current color map and displays the image. This function uses the following default LUT:



To visually compare these 3 functions you can also type the command ***figure*** before each of the ***image***, ***imagesc***, and ***imshow*** commands. Matlab will create graphic figure objects. Figure objects are the individual windows on the screen in which MATLAB displays graphical output.

```
% To visualize the differences between the functions
figure
image(I) ;
figure
imagesc(I)
figure
imshow(I)
```

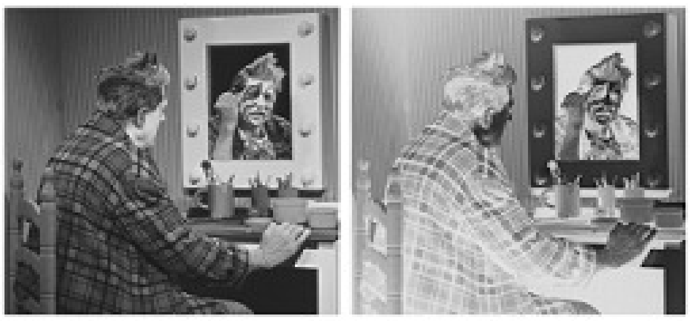
Example of conversion:

```
I = imread('CLOWN_LUMI.BMP');
J = double(I);
image(J);
```

4 – Here is an example to build the LUT which allows us to make the inversion video of the grayscale image *CLOWN_LUMI*:

```
% Open an image
Im = imread('CLOWN_LUMI.BMP');
% To build the LUT
r=1:-1/255:0;
g=1:-1/255:0;
b=1:-1/255:0;
lut=[r' g' b'];
image(Im)
colormap(lut)
```

Here are the images displayed:



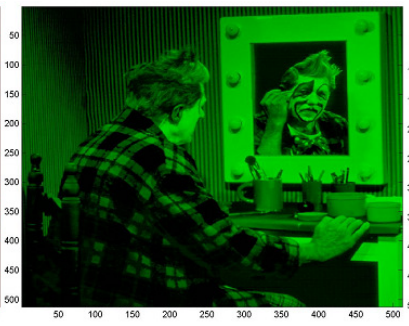
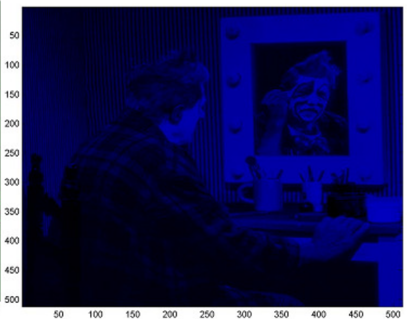
Thanks to an adequate LUT you can display the inverse video image without storing “image” data in the Matlab workspace and without transforming the original image.

5 – For a color image the data are three-dimensional and the type is uint8. The three planes of the image color are achromatic planes. To display the red one you must use a LUT whose outputs are only red levels
Here are the commands to visualize the three RGB planes of the color image *CLOWN*:

```
Im=imread('CLOWN.BMP') ;
% To build the LUT
vector=0:1/255:1;
r=vector;
g=vector*0;
b=vector*0;
lutr=[r' g' b'];
r=vector*0;
g=vector ;
lutg=[r' g' b'];
v=vector*0;
b=vector ;
lutb=[r' g' b'];
% Display the RGB planes
image(Im(:,:,1));       % Red plane
colormap(lutr);
figure
image(Im(:,:,2));       % Green plane
colormap(lutg);
```

```
figure
image(Im( :, :,3));     % Blue plane
colormap(lutb);
```

Here are the images displayed:

<u>Note</u>: Matlab is a matrix visualization tool which proposes specific LUTs. These LUTs are presented in the “***Supported Colormaps***” section of the Matlab help on the colormap. However you cannot build a LUT for color images with Matlab: it is not possible to create a LUT for each color plane Red, Green, Blue.