Lab 1. Hands-on Image Processing. Operations of Colorspaces.

Xabier Morales and Pritam Mishra

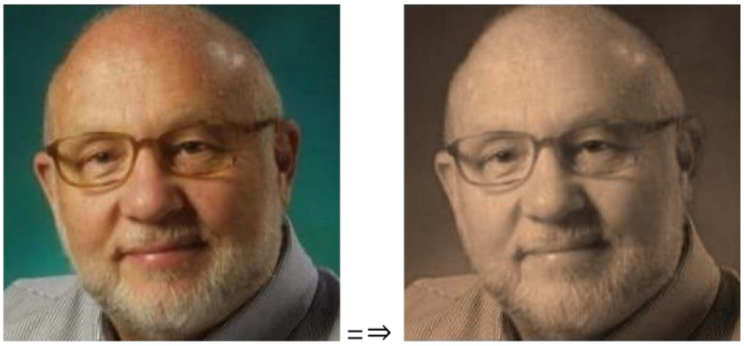
Course 2021-2022

Deadline: April 22nd (23:59)

Preliminaries. Check the Live Script included in the zip folder (+ the main tutorial) in order to get ready for this lab.

Exercise 1.

Sepia effect. Write a Matlab program to make the following transformation: given a color image, create a new image of the same size but in sepia color, like an old photograph.



Exercise 2.

Skin detector. Write a Matlab program able to detect pixels with skin color. Follow these instructions:

1. Take two images from the exact same position, with the same lightning conditions and a plain background: the first one must contain only background, and the second one must contain your face (or a head-shoulder image). In order to obtain better results, you should disable as many auto-settings from your camera as possible, such as white balance.
2. Detect, from the Foreground image, which are the pixels corresponding to skin.
3. Substitute those skin pixels with the color value of the same exact pixel in the background image.

Potentially, the result should filter properly the vast majority of the skin (it is difficult to have a perfect segmentation) whilst keeping clothes and parts of the face (lips or hair for instance). If you are facing difficulties with your own pictures, try with *BG1P, FG1P, BG2P* and *FG2* images attached in the .zip file.

Once tested with your own pictures (or the default ones), test your program with *BG3P, FG3P, BG4P* and *FG4* images. Does is still work? Why? If it does not, try to adapt it to the new pictures (*hint: set as inputs of the function the ranges you want to threshold*). Do you have problems with certain colors? Why?

Exercise 3. (Optional)

Average Image. Write a program in Matlab to compute an average image from two input images (such as `lena.png` and `barbara.png`). The average image can be computed with the following operations:

$$\frac{x+y}{2}$$

and

$$\frac{x}{2}+\frac{y}{2}$$

Use both operations using both *bytes* (uint8) and doubles images. This is: (1) uint8 - first equation, (2) uint8 - second equation, (3) double - first equation and (4) double - second equation. Do we obtain the same resulting image in the four cases? Why? Which are the main differences?
Hint: function `im2uint8` converts images to 8-bit unsigned integers.