# 1st report : equalization and denoising (lab 3)

## Aim of the lab:

In this laboratory experience we aim to learn how to equalize some image histograms using OpenCV and how to apply denoising/smoothing filters in a controlled way. This will also be the occasion to understand how to extend classes in C++ how to us trackbars of OpenCV. More precisely we were first required to perform the equalization on the single components of an image represented in the RGB space and to find another colour space where the equalization of only one component could give good results. Then we had to extend a class Filter in order to implement Gaussian Filtering, Median Filtering and Bilateral Filtering. It was in the end required to build a trackbar controlled graphic interface to vary the parameters of the filters and see the result in real time.

## Practical implementation difficulties and lessons learned:

- I had compilation problems ("undefined references" error for all methods) that I solved by fixing the library paths passed to the compiler by command line.
- Method cv::calcHist() required as parameters pointers and constant pointers, I had a hard time to define and initialize them in the right way, but with the help of the compilation error logs the mechanism behind pointers and references in C++ has become clearer.
- As a answer to the asked question, I would have guessed that by transforming the image into the HSV space and equalizing only the channel V ( Value ) we would have obtained a very good result, indeed in this way colours should stay at their place, and so their saturation : the only thing to be equalized should be the brightness of the scene. However by implementing it practically I realized that the colours become colder by equalizing the V component (if my implementation is correct at least). The better looking result was obtained through the RGB equalization in my case (see annexed images).
- Trackbars usage wasn't so easy for me: initially I wasn't sure about how many Callback functions I would have needed to create, the need to use only int as variables is also a problem, but it can be solved with proper castings.
- Filter class extension was immediate once I understood that the .h file defined the interface while the .cpp file contained the implementations

## Additional notes:

The C++ algorithm used to obtain all the previous results can be found in the folder "CV_Lab3_code", the main file that need to be compiled and executed run is "LAB3.cpp", it is essential that the main file is kept together with all its dependencies contained in the other files and to pass as a parameter the name of the image that we want to process when we run the program. Figures will appear quickly press any keystroke to pass to the successive one and in the end, to terminate the program.

*Figure 1:: Original image*

*Figure 2 : the equalization of the RBG components have been done individually and this is the recomposed image .*

*Figure 3: equalization done only on the V component, note the dark and colder atmosphere…*