

## LAB 3

### 1. Histogram Equalization

In this lab I have tried to implement the code by dividing it in multiple function, to have the code divided in more modules. The first part of the homework is done by the two functions `RGB_histogramEqualization` and `HSV_histogramEqualization`; I have also implemented `computeHistograms` and `showResizedImg`.

The first function it's just a recalling of the function `w1` provided by OpenCV, but with some variables already set that the function `calcHist` needs (this improve the readability of the code): this is done because for our homework only the bins and the upper range value of the value that we want to consider change.

The second function it has been written only to have a output that could be able to fit my screen.

I have test this part of the homework with:

- `countryside.jpg` (dark image)
- `image.jpg` (dark image)
- `overexposed.jpg` (light image)

and in the next pages there are shown the results.

As we can see, the dark images don't present notable histograms in the highest part of the value range in the RGB color space and their equalization produce unnatural results. We obtain same poor results if we consider the HSV color space and we equalize the H and S channel; instead an equalization on the V channel produces in both images a lighter an well exposed (to the light) image, this is due to the fact that the value channel represents the brightness of the image and the equalization tries "to spread the light" all over the pixels in an equal way.

With the overexposed image, instead, we obtain a good results with the RGB color space equalization too: in fact it's hard for me to detect which computed image represents more the reality (in the RGB case I can suppose that the image has been taken during a sunset with a cloudy sunset, instead in the HSV case I might think it has been taken in a simply cloudy dark day).

I have also tried the equalization with other overexposed images (found on internet) and I have obtained the same similar results among the RGB equalization and the HSV – value channel only – equalization that I have obtained with the provided image, `overexposed.jpg`.

It's good to notice the outstanding art result obtained in the saturation channel equalization.

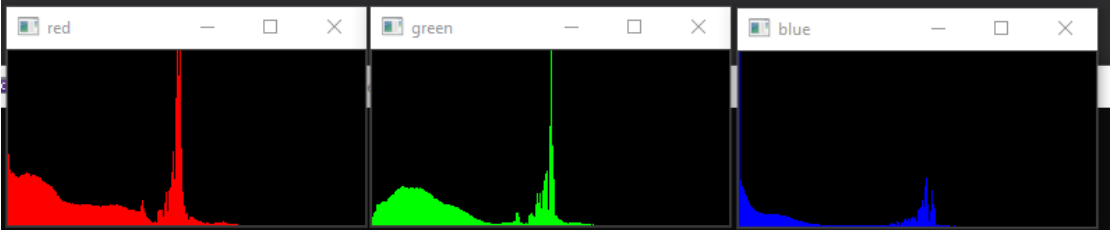
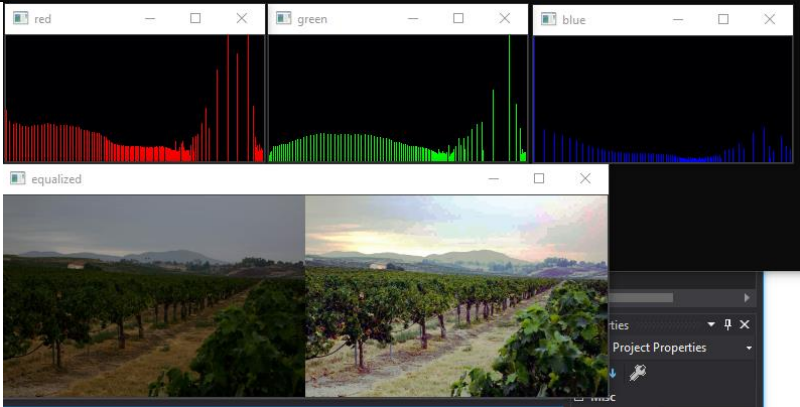
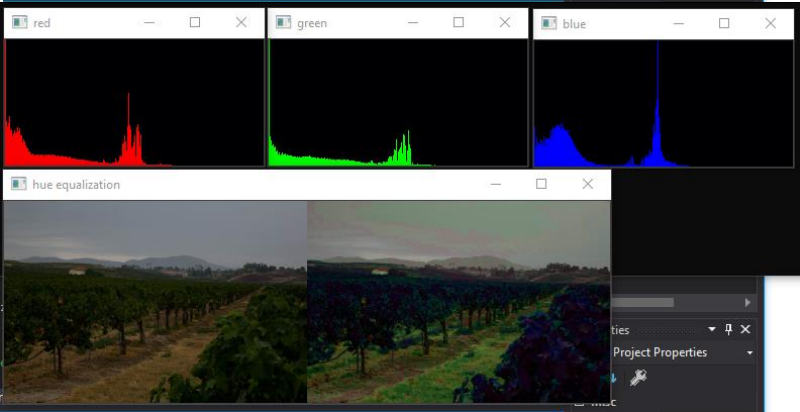
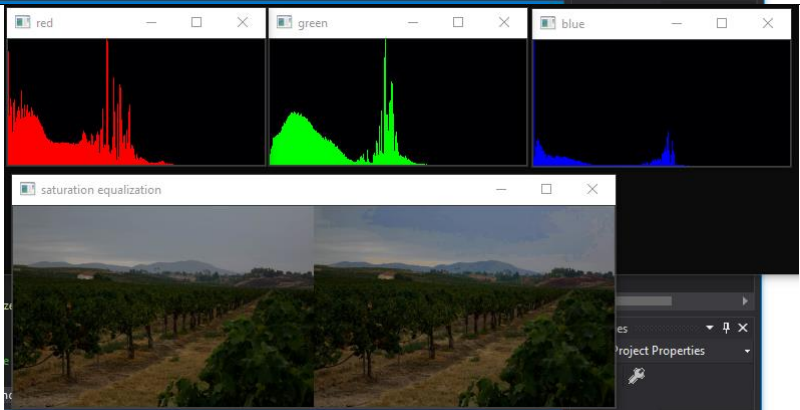

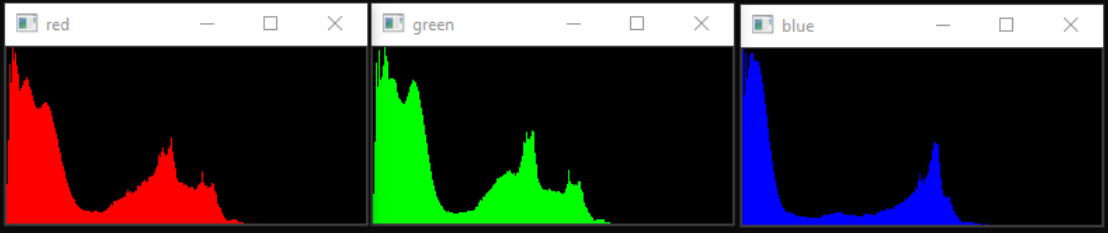

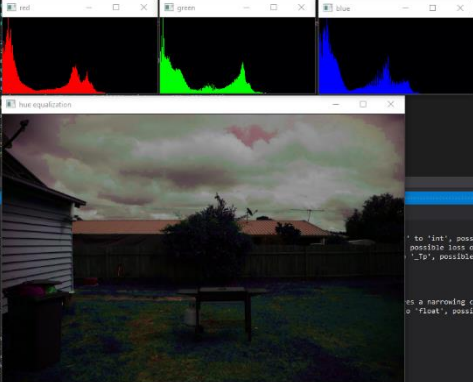
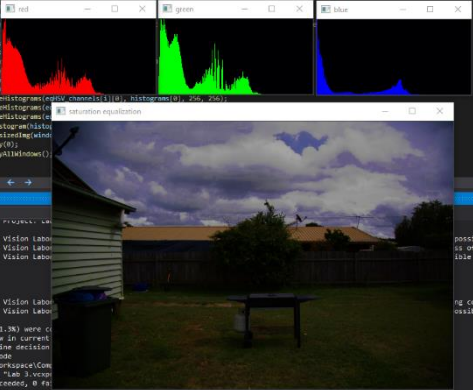
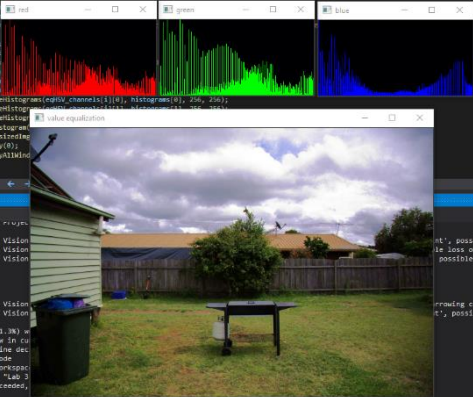
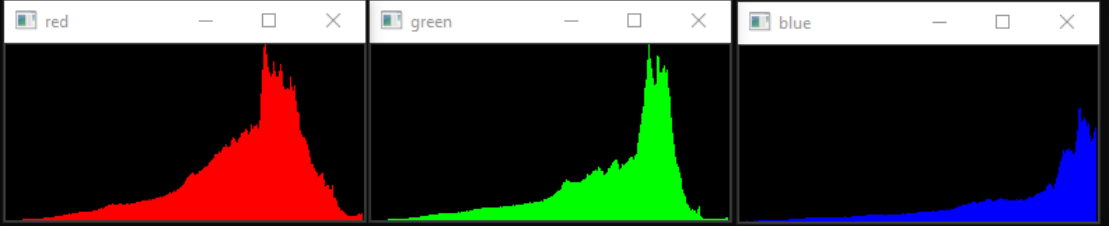
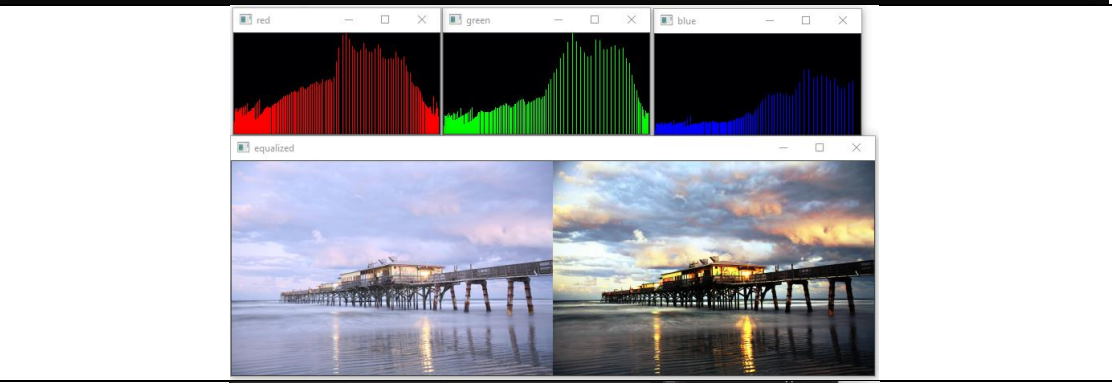
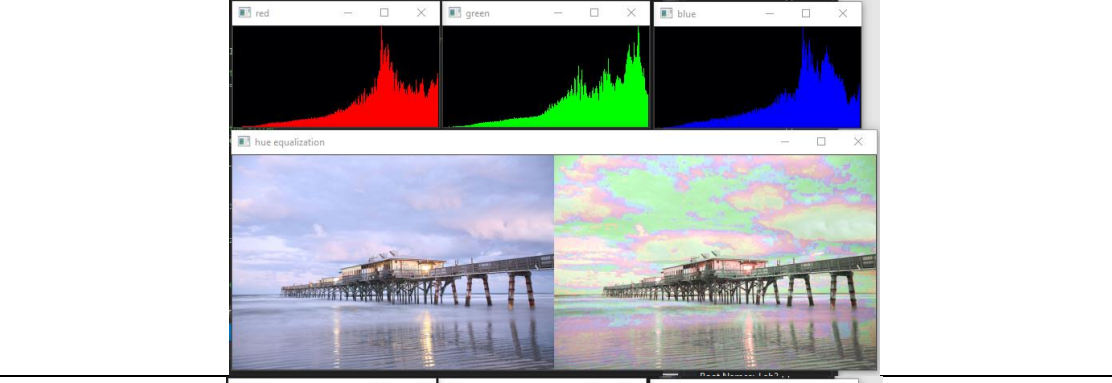
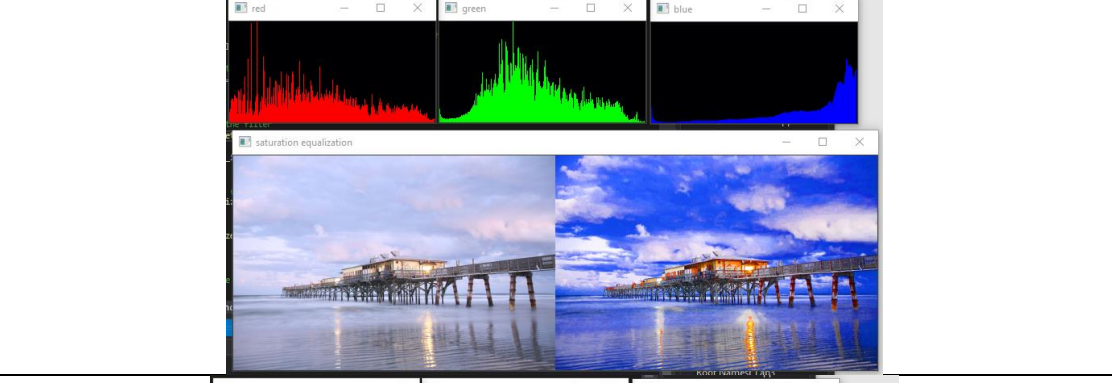
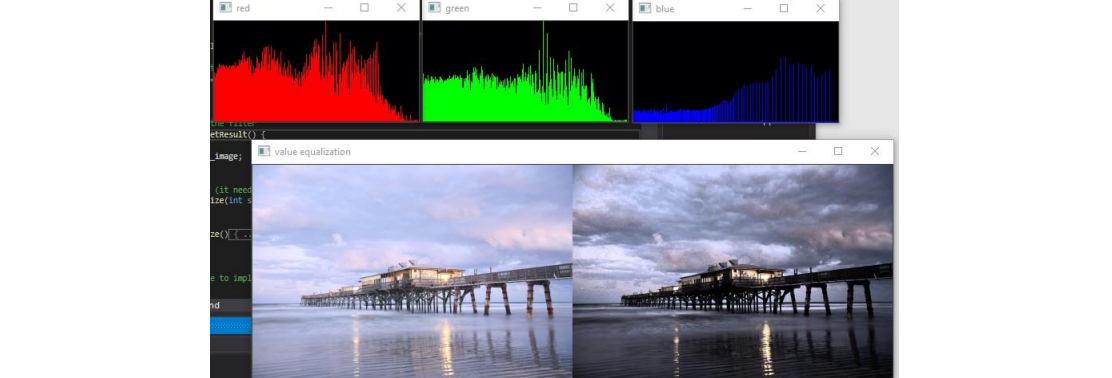
<p>RGB Histograms</p>	
<p>Equalized Histograms</p>	
<p>Equalized Hue Histogram</p>	
<p>Equalized Saturation Histogram</p>	
<p>Equalized Value Histogram</p>	

image.jpg

RGB Histograms			
Equalized Histograms			
Equalized Hue Histogram			
Equalized Saturation Histogram			
Equalized Value Histogram			

overexposed.jpg

RGB Histograms	
Equalized Histograms	
Equalized Hue Histogram	
Equalized Saturation Histogram	
Equalized Value Histogram	



## 2. Image Filtering

In this part I have implemented three filters in `filter.cpp`, that are derived class of the provided class `Filter`. In each class I have re-implemented the constructor and the function `doFilter`, in which it's recalled the related OpenCV filtering function, and I have added the `set` and `get` functions for the additional parameters. In the bilateral filter I have decided to use the  $6 \cdot \text{sigma\_space}$  rule to set the filter kernel size.

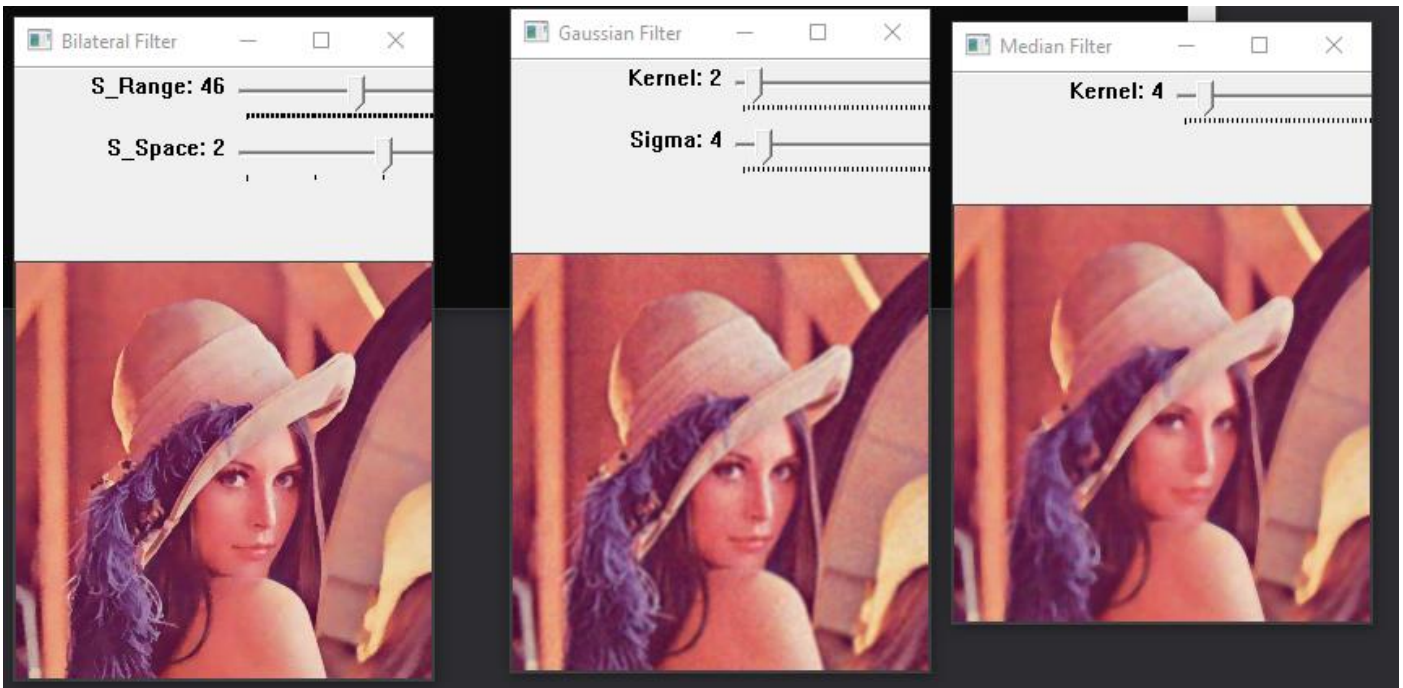
For each filter I have created a window with one trackbar for each parameter needed to change the filter behaviour and each trackbar can set the related value in a range between 0 and 100.

For the only case of the bilateral filter: `sigma_range` has a range between 0 and 200 (to allow to create a "cartoonish" effect"); `sigma_space` has a range between 0 and 10, as suggested in the OpenCV documentation.

In the code, the window names are defined as global variables to allow changes on the related windows through the trackbars.

Also, the input image is resized for viewing purpose. If `scale` is set to 1.0, the image has the original size.

This part has been tested with `lena.png` image and we can notice that the median filter doesn't provide any kind of improving (in fact it's only blurring the image, not denoising). We have also poor result in denoising with the gaussian filter, instead the bilateral filter is providing good results with the parameters shown in the below pictures.



I want to let notice that for the median filtering it has been used the function `medianBlur` of the OpenCV library, instead of `medianFilter`, because I wasn't able to find the requested function.