Università degli Studi di Padova

Department of Information Engineering

Master degree in ICT for Internet and Multimedia

Course: Computer Vision

Lab 3 report

Davide Carta

ID: 1210702

May 3^{rd} 2019

Histogram equalization and Image filtering

This lab's aim was to load an image and show his histograms before and after equalization. Furthermore, working on a different color space and do the same operations is required. In the second part the image had to be displayed in a window featuring a trackbar to slide up and down some values related to three different filter's parameters.

Histogram equalization

An image was loaded and displayed. Since we were working on RGB color space, three different histograms, one each plane, had to be computed using the cv::calclHist() function. To display them, we were provided a specific function showHistogram().

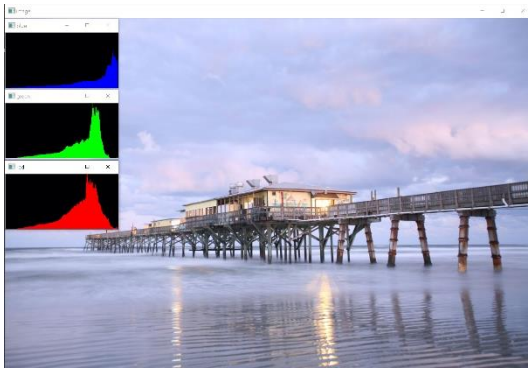

Figure 1. Original image and histograms

Figure 2. Equalized image and histograms

At this point all the three histograms had to be equalized and it was done by mean of the cv::equalizeHist() function. The equalized histograms were merged back together and, together with the equalized image, displayed.

At this point we were required to repeat the above operations working on a different environment: the Lab color space. This time only one channel had to be equalized and it had to be chosen in order to get the best output image. I choose to equalize the "L" channel. To display the equalized image, this one must be taken back to the RGB color space using the function cv::cvtColor() , otherwise a bad output image would appear. Below, the results obtained.

Figure 3. Image in the Lab color space and histograms

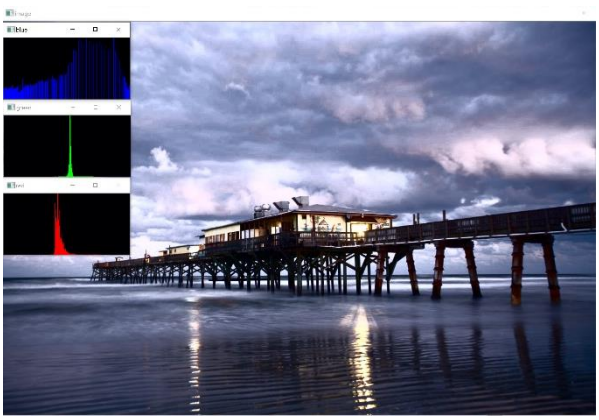

Figure 4. Equalized image in the Lab color space and histograms

Image filtering

The second part of the lab required some workarounds. First thing first, a class filter had to be coded, for further usage, to generate the three filters (Gaussian, Median and Bilateral) to be applied to the image. Second step required to code up a class Parameters for each type of filter, containing the parameters to be passed to the callback function later. At this point the callback functions for the trackbars had to be created, using the cv::createTrackbar() function. Callbacks are called every time the track bar position on the image is moved causing the related parameter to be updated, and the image as well. This was the most demanding yet challenging part of the lab, giving after a lot of debugging some satisfactory results. Below, the image resulting when applying a Gaussian filter with some specific size and sigma values, is displayed.

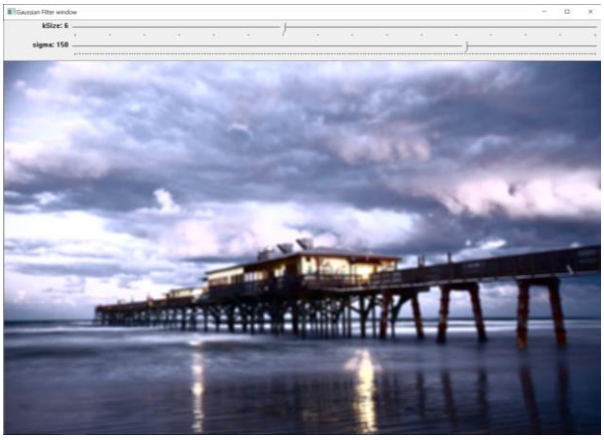

Figure 5. Gaussian filter window with size and sigma trackbars