Homework 1 OpenCV

Computer Vision 2021/22

Alessandro Manfè 2041603

March 31st 2022

This homework report purpose is to provide the experimental results of some application of OpenCV library. The first part involves various filters while the second concerns the part relating to the histograms. Comments on problems encountered and concolusions are also provided at the end of the document.

Filtering

For the filtering part it's provided the starting grayscale image of which is highlighted the detail of the cables. (Image 1)

Image 1: Cables detail

maxFilter

First implemented filter is the maxFilter of variable kernel. It simply replace a pixel value with the max value of neighbors pixels inside the kernel centered in it. We can observe the same detail with kernel dimension: 3 (a), 5 (b), 7 (c), 9 (d).

Image 2: maxFilter

minFilter

Same as the maxFilter but considering the smallest value.
Same kernel sizes: 3 (a), 5 (b), 7 (c), 9 (d).

Image 3: minFilter

We can notice how the maxFilter successfully removes the cables from the image but corrupts it making it more pixelated, while the minFilter, for the nature of its behaviour, tends to emphasize more pixel with low brightness value so dark objects become even more darker. It's clearly not the right filter for this task.

medianBlur

Using medianBlur() function of OpenCV library.
Kernel sizes: 3 (a), 7 (b), 11 (c), 15 (d).

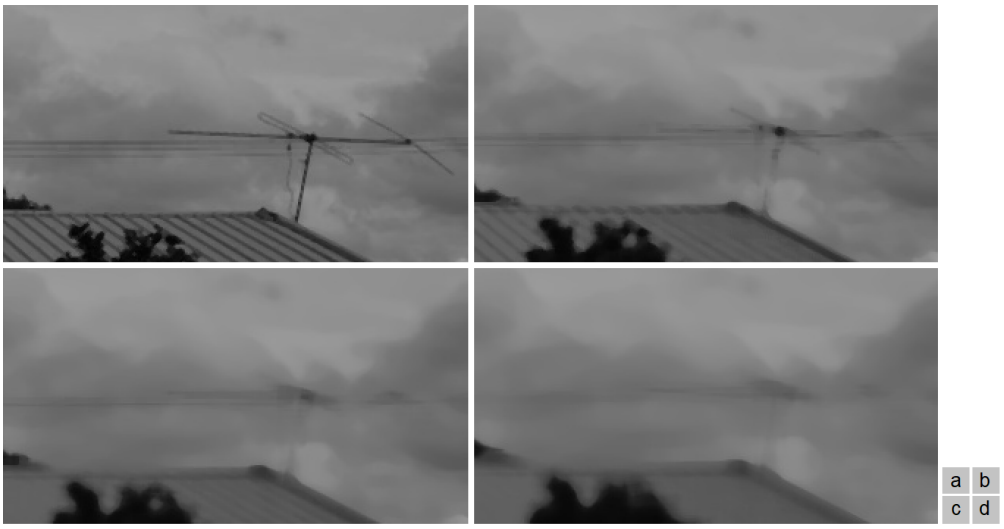

Image 4: medianBlur

GaussianBlur

Using GaussianBlur() function of OpenCV library with auto-computed standard deviations. Kernel sizes: 3 (a), 7 (b), 11 (c), 21 (d).

In this case both filters apply a smooth blur to the image, which effects are quite similar. GaussianBlur seems to better preserve high variations in brightness like for the rod of the antenna, which is still visible with kernel of size 21. Overall this kind of blur is not suitable for eliminating small details like the cables because the image becomes completely indistinguishable before these are effectively removed.

Image 5: GaussianBlur

Histograms

Using a function specifically implemented for this purpose (See "histToImg()" in "Lab2.cpp") we plot the histogram of the original grayscale image with different number of bins.
In this case we have: 256 (a), 100 (b), 50 (c), 20 (d).

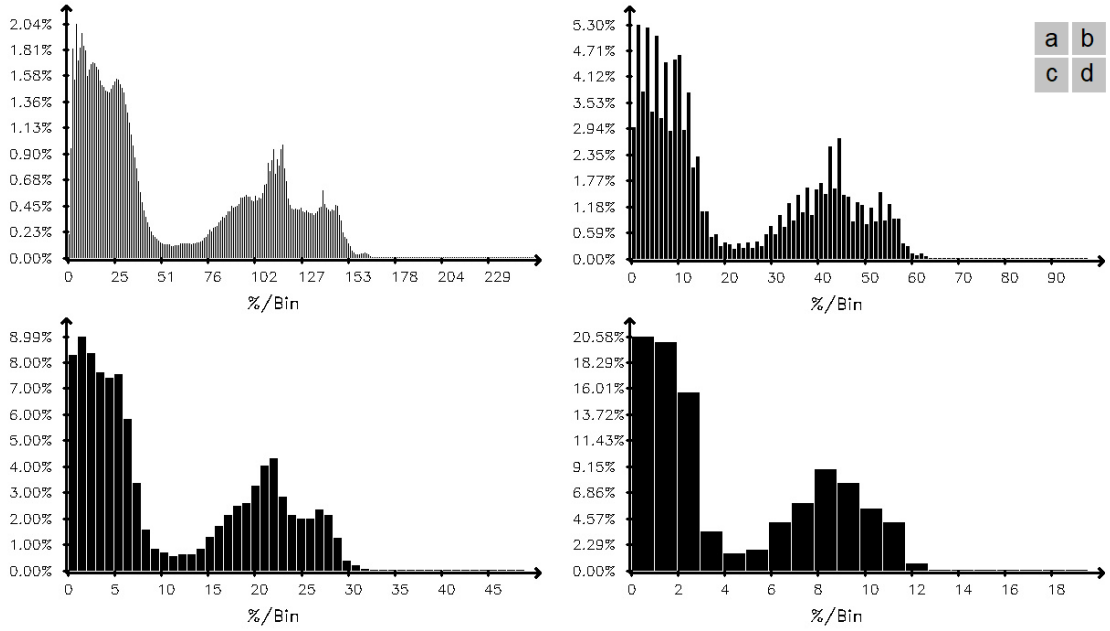

Image 6: Histograms

Lastly we equalize the histogram of the original function to obtain a new normalized image using equalizedHist() function of OpenCV. We can observe the original and new image and their histograms.

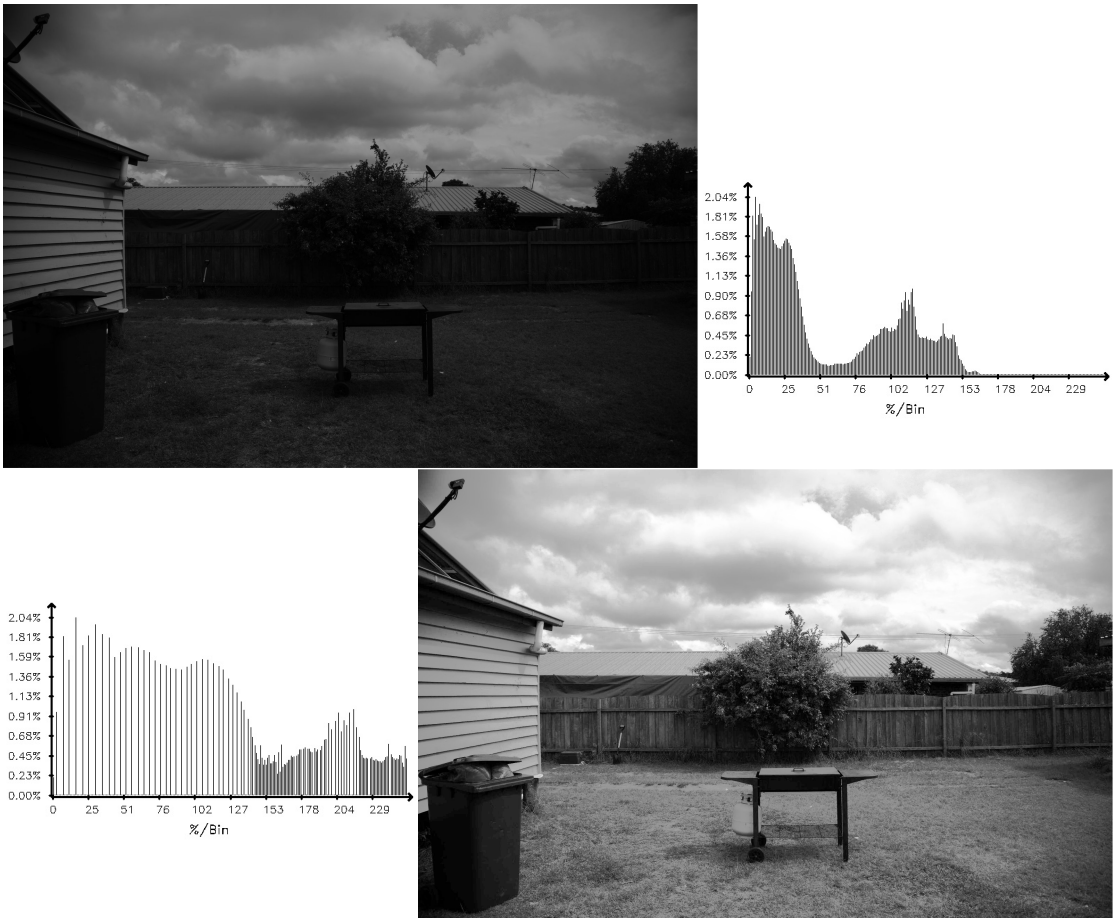

Image 7: Original and equalized image

Conclusions

The main problems encountered in this homework mainly concerned the installation and the correct setting of OpenCV, cause i found it difficult to set up the debugger correctly. Furthermore having the pre-compiled libraries does not allow you to easily recognize the exceptions thrown by the program (For example i struggled a lot with filters before figuring out that at() function is not range checked :c). Apart from this the documentation of OpenCV is very clear and useful, so i didn't encounter particular problems.