COMPUTER VISION AND IMAGE PROCESSING

LAB SESSION 2
POINT AND LOCAL OPERATORS

ALESSIO TONIONI - ALESSIO.TONIONI@UNIBO.IT

Image Processing Lab

- Today's lab:
  - Histogram creation and visualization (Exercise 0)
  - Intensity transformation (Exercise 1-2-3)
  - Convolutions (Exercise 4)

- Download '*Laboratories.zip*' from the web page of the course at didattica.arces.unibo.it/

- Extract the zip archive and open '*ElabImage_2010.sln*'

- We are going to work on '*Lab_Session_2*' project.

Image Processing Lab

- Fill the missing code in functions defined in ‘*functions.cpp*’ and execute ‘*Lab_2.cpp*’ to check your work.

- To choose between C or C++ comment or uncomment the first define on top of ‘*functions.h*’.

C++

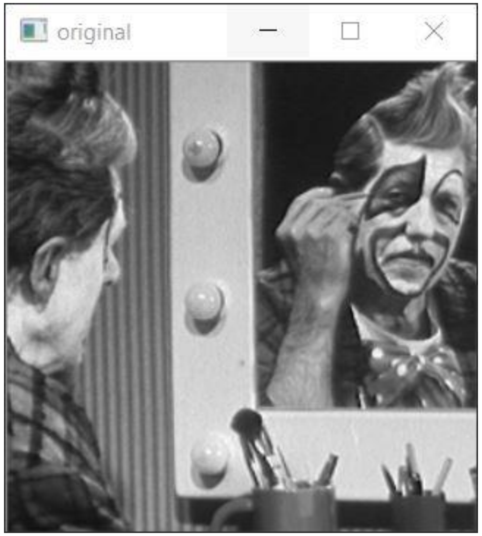
```
#define OPENCV_CPP_INTERFACE
```

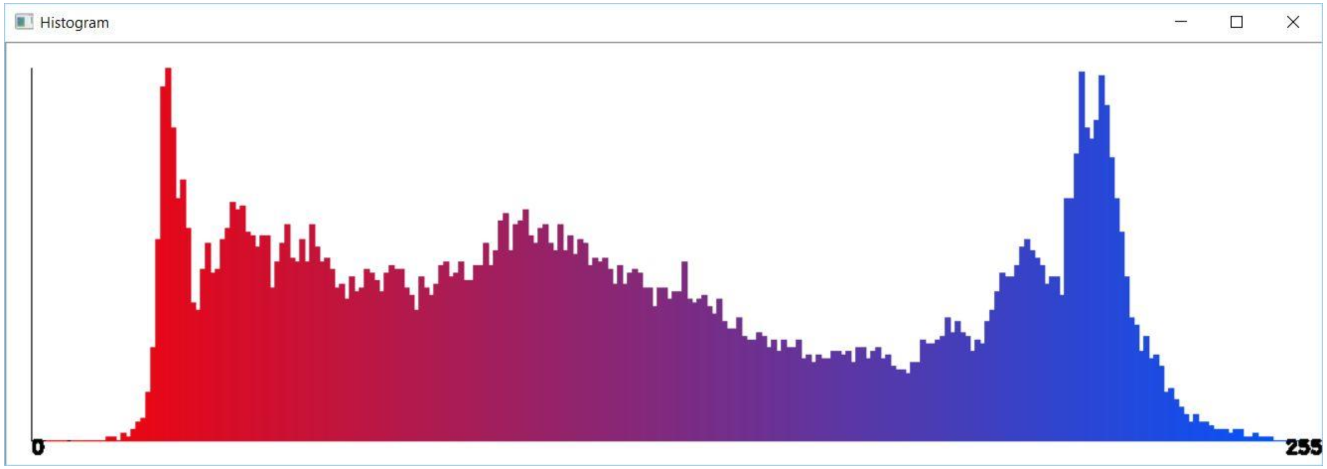
C

```
//#define OPENCV_CPP_INTERFACE
```

Exercise 0 – Histogram creation

- Complete ‘**buildHistogram**’ in ‘*functions.cpp*’ with the proper code to create the histogram of a grey scale image.





<u>Image filename</u>: *data/cln1.bmp*

Exercise 1 – linear contrast stretching

Given P_{in} = pixel intensity in the original image, P_{out} = corresponding pixel intensity in the modified image and the following formulation:

$$P_{out} = \frac{255}{P_{\max} - P_{\min}} \left(P_{in} - P_{\min} \right)$$

Complete '**linearContrastStretch**' function in '*functions.cpp*' using one (or both) the definition for P_{min} and P_{max} :

A. P_{min} and P_{max} are respectively set to the minum and maximum intensity values in the current image.

B. P_{min} and P_{max} are respectively set to [1-5]% and [95-99]% of the image histogram bins.

Exercise 1 – linear contrast stretching

Original Image

Enhanced Image

Image filename: *data/wom1.bmp*

Exercise 2 – gamma correction

Complete '**gammaCorrection**' functions in '*functions.cpp*' implementing the gamma correction operator according to:

$$P_{out} = 255^{(1-r)} \cdot P_{in}^{r}$$

(r is a parameter of the operator and of the gammaCorrection function)

Note:

- The C library "*math.h*" includes a power function: pow(x,y) = x^y
- The C++ '*std*' library includes a power funciton: std::pow(x,y) = x^y

Exercise 2 – gamma correction

Original Images (over-exposed)

Enhanced Images (r = 1.5)

Image filename: *data/fce4.bmp*

Exercise 3 – equalization

Complete ‘**imageEqualization**’ function in ‘functions.cpp’ according to this formulation:

$$j = T(i) = \frac{255}{M \cdot N} \cdot \sum_{k=0}^{i} h(k)$$

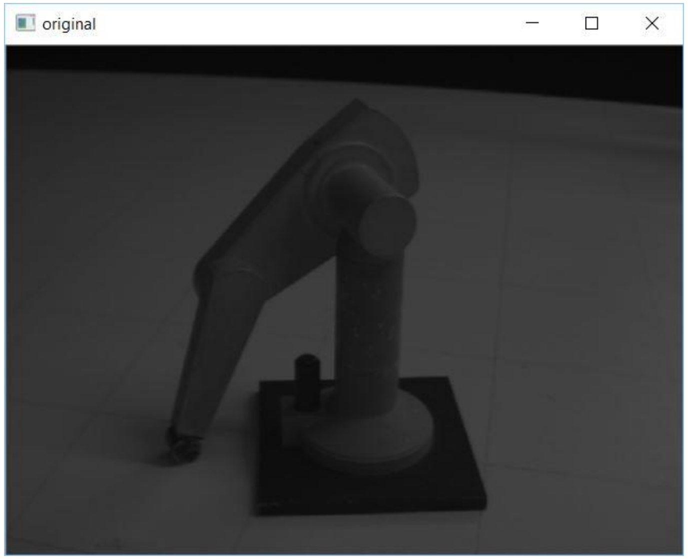
Where:

- ***M***, ***N***: number of rows and columns of the input image
- **i:** i-th greylevel of the input image
- **J**: j-th greylevel of the output image
- **h(k):** samples in the k-th bin of the input image histogram

NOTE: one way to implement equalization is to first compute the cumulative distribution of the input histogram, then compute each output pixel based on that.

Exercise 3 – equalization

Original Image



Enhanced Image

Image filename: *data/pum1dim1.bmp*

Exercise 4 - convolution

Complete **'convolution'** in '*functions.cpp*' according to this formulation for the convolution operator:

$$O(i, j) = \sum_{m=-k}^{k} \sum_{n=-k}^{k} K(m,n) \cdot I(i-m, j-n)$$

Where:

- **O**: output image;
- **K**: square matrix with side *k*2+1* defining the convolution kernel;
- **I**: input image

Exercise 4 – convolution (cont.)

Test your implementation using as kernel:

a) Denoising kernel

$$K(m,n) = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

b) High pass filter (edge-enhancement)

$$K(m,n) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Note:
For case **(b)** the output image could contain values with a negative sign and/or greater than 255. These cases have to be handled accordingly to avoid «overflow» errors.

Exercise 4 – convolution (cont.)

Original Image

Kernel A

Enhanced Image

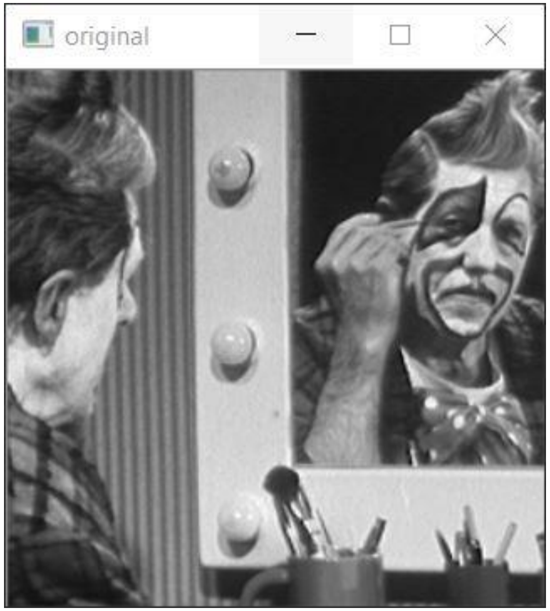
Image filename: *data/fce5moregaussnoise.bmp*

Exercise 4 – convolution (cont.)

Original Image

Kernel B

Enhanced Image



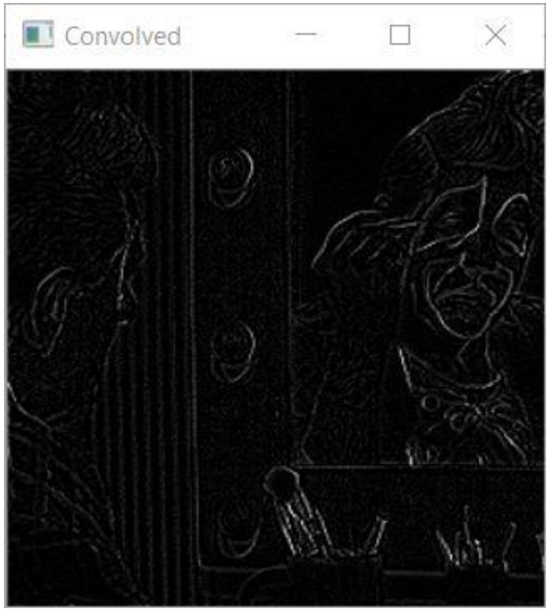


Image filename: *data/cln1.bmp*