COMPUTER VISION AND IMAGE PROCESSING

LAB SESSION 2
POINT AND LOCAL OPERATORS

DR. FEDERICO TOMBARI

Exercise 1 – linear contrast stretching

a) Implement the **contrast stretching** operator using the following formulation:

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

Note: to apply this operator, the minimum (P_{min}) and maximum (P_{max}) intensity values in the current image have to be computed

b) Implement the contrast stretching operator in its more general formulation, setting (P_{min} , P_{max}) respectively to [1-5]% and [95-99]% of the image histogram bins.

Note: finding the 2 percentiles requires the computation of the image intensity histogram. Also, in this case pixels won't fall within the [0,255] range.

Exercise 2 – gamma correction

Implement the gamma correction operator according to:

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

r **being the parameter of the operator.**

Note:

The C library "math.h" includes a power function: pow(x,y) = x^y

Exercise 3 – equalization

Implement the image equalization operator according to the following formulation:

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

where

- *M*, *N* are, respectively, the number of rows and columns of the input image
- i is the i-th bin of the output image histogram
- h(k) is 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 4 - convolution

Formula of the Convolution operator:

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

- O: output image;
- K: matrix defining the convolution kernel;
- I: input image

a) Implement the convolution operator using, as kernel, that typically employed to perform low-pass filtering (useful e.g., for image denoising):

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

Exercise 4 – convolution (cont.)

b) Modify the convolution operator implemented in the previous step to implement a typical high-pass filter (useful, eg. for edge-enhancement) by using the following kernel:

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

Note:

In this case, 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.