Master Artificial Intelligence
&
Virtual Reality

Practical work n°2

Introduction to Image Processing and Analysis

1. Averaging filter

1.1) Using OpenCV, load the **gray_disc.png** as a grayscale image into a variable named **img** from the ressources provided in this practical work.

To be sure that the image has been well loaded, you can view it with either cv2.imshow('original_image',img) or with PIL.Image.fromarray(img).show() (of course after importing the associated packages).

1.2) Define two averaging filters h1 and h2 of size (5x5) and (11x11) respectively, and use them to filter the image using the OpenCV built-in function **filter2D**. Read its documentation in order to use it !

1.3) What can you notice from the obtained results related to the question 1.2 ?

1.4) Plot the gray levels of the line 400 of the original image and the filtered image with the kernel of size 11x11. What are your remarks? What can you deduce ?

1.5) Load the noised_disc.png in a variable named **noised_img** and visualise it. What kind of noise is that?

1.6) Let's apply the two filters h1 and h2 defined in question 1.2. What can you notice from the resulting filtered images ? What can you deduce ?

2. Median filter

2.1) Using OpenCV, load the **gray2_disc_noised_s&p.png** as a grayscale image into a variable named **noised_sp** from the ressources provided in this practical work. What type of noise has affected this image?

2.2) In order to denoise the latter, what is the appropriate filter to use?

2.3) Let's try to denoise this image using the built-in OpenCV function **medianBlur** after you have read its documentation (try size 9). What can you notice?

2.4) Let's try to denoise the image using the filter h2 defined in the question 1.2. What can you notice/deduce ?

2.5) Let's try to denoise the image affected with the gaussian noise with a median filter of size 9. What can you notice/deduce?

3. Behind the scene

Now after you have well understood the basics of these filters, you are invited to do more !

3.1) Having the free-distorted image named **gray_disc.png**, try to generate its gaussian and Salt&Pepper versions using the function **random_noise** from the **Scikit-image** library. You should have the same images as the ones provided in the ressource folder.

3.2) Above, you have used the **filter2D** OpenCV built-in function to perform local avearge filtering as well as median filtering. Now, the question is how this convolution is made ? What is the technique implemented to perform such convolution? In order to understand this, try to implement a function named myConvolution. Here are some tips that might help you :

- How much and what are the entries the function will be waiting for ?

- As seen in the image analysis and processing course, convolution opearation can be defined simply as an **element-wise operation followed by a sum** :

$$\begin{bmatrix}1&4&7\\2&5&8\\3&6&9\end{bmatrix} * \begin{bmatrix}1/9&1/9&1/9\\1/9&1/9&1/9\\1/9&1/9&1/9\end{bmatrix} = \begin{bmatrix}1/9*1&1/9*4&1/9*7\\1/9*2&1/9*5&1/9*8\\1/9*3&1/9*6&1/9*9\end{bmatrix}$$

$$= \quad 0.31$$

- When performing a convolution of an image using any kind of filter, the image size is reduced ! see below :

$$\begin{bmatrix}1&5&9&13\\2&6&10&14\\3&7&11&15\\4&9&12&16\end{bmatrix} * \begin{bmatrix}1/9&1/9&1/9\\1/9&1/9&1/9\\1/9&1/9&1/9\end{bmatrix} = \begin{bmatrix}6&9.99\\7.11&11.11\end{bmatrix}$$

- As you notice, the image went from a size of 4x4 to a size of 2x2. However, in some applications, such the ones we are dealing with in this practical work, we would like to have a resulting image having the same size of the input image. To do so different strategies can be implemented using image padding. Two of them are widely used :

1- Replicating the pixels along the border of the image : to have an ouput result of size 4x4, the original image is transformed into an image of size 6x6.

2- Zero padding : filling the borders with 0.

Try to implement the first strategy using the skimage.exposure built-in function : **copyMakeBorder.**

- Browse your new transformed image and perform the convolution by sliding the kernel (try to use the h2 filter defiend in question 1.2) over the image.

- Postreat your resulting image and visualise/save it in order to compare with result obtained with cv2.filter2D.

3.3) Try to get a similar result to the one obtained in question 2.3 by implemting the median filtering operation.