IPBMA. Exercise 5

Building a basic projection radiography system

Built a projection radiography system using Python functions and tried to check its degree of operation using the cube phantom (see the example of the enclosed html file). The functions will be called from the main program. The functions to be implemented will be: ***source(), interactor_PR(), detectorNoiseP_1_1(), getNumberPhotons(), getNumberPhotonsCell(), getNumberCellsPhoton(), plotDistribution(), getContrast() and getSNR()*** and will include the following parameters:

source(kVp, N0).- a function that simulates an x-ray tube

i) kVp → Peak voltage difference between anode and cathode.
ii) N0 → Number of photons generated per unit area.

Note.- Suppose the unit area is the size of each point of the quantum image generated after the x-ray beam cross the object.

Output→ Effective Energy of the photons (eE) and N0. Suppose 40% of the maximum value.

interactor_PR(N0, Object, Projection).- a function that simulates the interaction between the x-ray beam and the object, in this case, a phantom. This interaction depends on the linear attenuation coefficient and the object's shape. Both quantities are included in the object itself.
Both frontal and lateral projections must be feasible to select.

i) N0 → Number of photons generated per unit area.
ii) Object → Object to be radiographed.
iii) Projection → Frontal, lateral

Note.- Suppose the object is cube whose edges measure 1 cm.

Output→ Numpy matrix (2D), whose values represent the number of photons per unit area of the quantum image obtained after the object is radiographed.

detectorNoiseP_1_1(Image).- function that simulates a squared digital detector, following Poisson distribution (same as exercise 4).

i) Image → Quantum image to be captured.

Output→ Numpy array (2D), whose values will be the image's pixel values of the image detected.

getNumberPhotons(image): function to calculate the image's total number of photons.

Output→ Total number of photons represented by the whole image.

getNumberPhotonsCell(image, maxValue): function to calculate the distribution of photons captured with respect to the cells of the image that capture a specific number of photons.

i) Image → Image to be analyzed.
ii) maxValue → maximum value of photons per cell.

Output→ Histogram of the distribution of the photons in the image.

getNumberCellsPhoton(image, maxValue): function to calculate the distributions of cells of the image with respect to the cells of the image that capture a specific number of photons..

i) Image → Image to be analyzed.
ii) maxValue → maximum value of photons per cell.

Output→ Histogram of the distribution of the cells in the image.

plotDistribution(data, xLabel, yLabel): function to plot the distribution of an x-y input data.

i) data → 2D array including x-y data.
ii) xLabel → Label of the x-axis of the plot
iii) yLabel → Label of the y-axis of the plot

Output→ 1D plot of the distribution x-y data.

getContrast(image, x0, y0, x1, y1, w): function to calculate the contrast between two squared regions of size *w* x *w*, located at (x0, y0) and (x1, y1), respectively (see Figure 1 in the appendix).

img → Image from which the contrast will be calculated.
(x0, y0) → (x, y) coordinates of the center of the first region.
(x1, y1) → (x, y) coordinates of the center of the first region.
w → Length of the side of the square area.

Output→ Value of the contrast (in the interval [0, 1]) between the two selected regions.

getSNR(image, x0, y0, w): function to calculate the SNR (signal to noise ratio) of a squared region of size *w* x *w*, located at (x, y)) (see Figure 1 in the appendix).

img → Image from which the contrast will be calculated.
(x, y) → (x, y) coordinates of the center of the region.
w → Length of the side of the square area.

Output→ Value of the SNR of the selected region.

Note.- each student has to bring a zip file called *lastName_Name_P5.zip*, to the following address: pablogtahoces@gmail.com. The subject of the e-mail should be IPBMA_P5. Inside the zip should be included:

- A jupyter notebook showing how the software works (see the example).
- An html file of the notebook.
- A .py file with the Python functions created.
- A pdf file explaining the coherence/non-coherence of the experimental results obtained.
- All the necessary files to verify the correct operation of the application.

Deadline: Wednesday, October 23, 10:00.

APPENDIX

The functions ***getcontrast()*** and ***getSNR()*** compute the contrast (between 2 regions) and the SNR (in one of the regions), respectively.

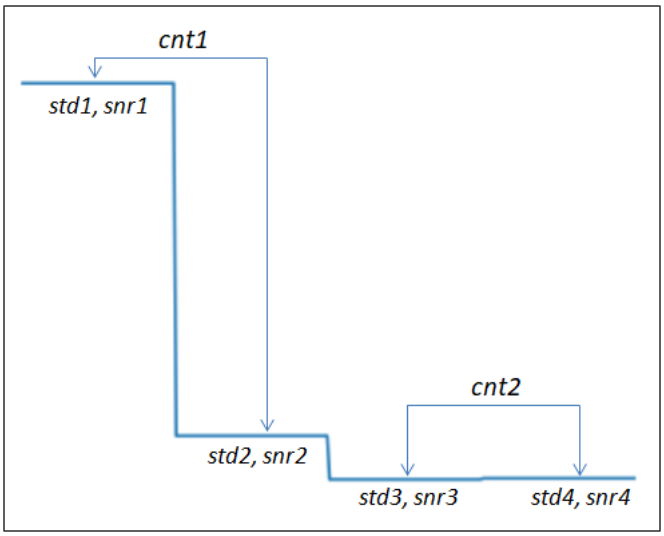

Figure 1