IPBMA. Exercice 5

Building a basic projection radiography system

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

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

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

Output→ Effective Energy of the photons (eE) and N0

interactor_PR(N0, Object, Projection).- function that simulate 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 shape of the object. Both quantities are included in the object itself.
Both, frontal and lateral projections are feasible to select.

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

Output→ Numpy array (2D), whose values represent the number of photoms per unit area of the quantum image

detectorNoiseP(Image, n).- function that simulate a squared digital detector of n x n cells, following Poisson distribution.

i) Image → Quantum image captured
ii) N → Number of cells per side of the detector. Therefore, the total number of cells in the detector will be n x n.

Output→ Numpy array (2D), whose values represent the amount of electrical charge produced by the incident radiation inside each detector cell.

getNumberPhotons(image): function to calculate the total number of photons represented by the whole image.

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

getNumberPhotonsCell(image, range): function to calculate the distribution of photons per cell with a number of photons detected.

i) Range → Range of values included in the distribution.

Output→ 2D array representing the distribution of photons per cell with a number of photons detected.

getNumberCellsPhoton(image, range): function to calculate the distributions of cells per number of photons detected.

i) Range → Range of values included in the distribution.

Output→ 2D array representing the distribution of cells per number of photons detected.

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.

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).
- A .py file with the python functions created.
- All the necessary files to verify the correct operation of the application.
-

Deadline: Wednesday, December 22, 10:00 AM CET.