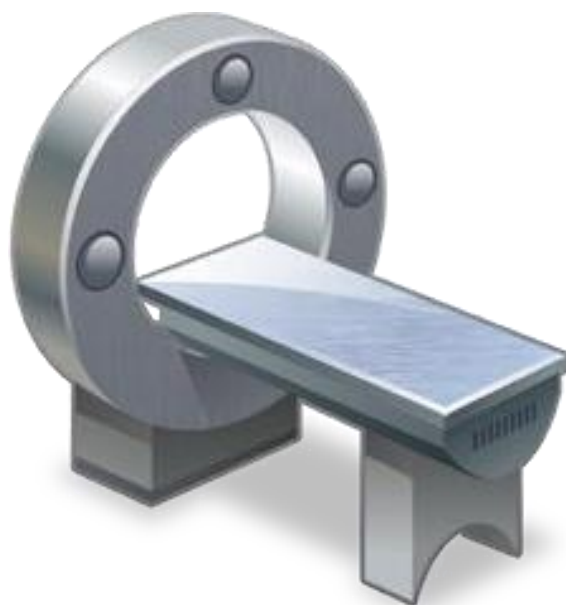


QSPECT

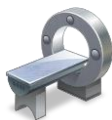


“Reconstruct your own SPECT data”

User's Guide v3.0

 BIOEMTECH

2019, Athens

**Disclaimer:**

QSPECT v3.0 has been developed for research purposes. It is not intended for clinical use. QSPECT is not a clinically approved medical software, and users understand and accept that the developers have no responsibility if reconstructed data obtained using QSPECT are used for clinical purposes.

Acknowledgments:

For any use of this software please refer to the following publication:

G. Loudos, P. Papadimitroulas, P. Zotos, I. Tsougos, P. Georgoulas

[Development and evaluation of QSPECT open-source software for the iterative reconstruction of SPECT images](#)

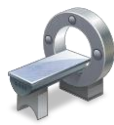
Nuclear Medicine Communications 2010 Jun; 31(6): 558-66.

Contributors / Contact:

Dr. George Loudos (george@bioemtech.com)

Dr. Panagiotis Papadimitroulas (panpap@bioemtech.com)

Dr. Nikos Efthimiou (nikos.efthimiou@gmail.com)



QSPECT User guide

Contents

I. Introduction	3
II. SPECT Image reconstruction.....	3
Data loading and visualization	3
Construction of the probability system matrix.....	5
Reconstruct & export SPECT data.....	6
III. Benchmark examples	7
Benchmark-1: Preclinical pixelated crystal (MOBY – GATE)	7
Benchmark-2: Clinical homogeneous crystal (XCAT – GATE)	8
IV. Notes.....	10
IV. Appendix “Parameters need to be defined in projections .c scripts”	10

I. Introduction

QSPECT is an Image Reconstruction application with graphical user interface, which visualizes images from SPECT data. QSPECT integrates image reconstruction by using the Maximum Likelihood Expectation Maximization – MLEM iterative algorithm. The QSPECT's functions are analytically described.

II. SPECT Image reconstruction

Data loading and visualization

The user may start the procedure by clicking the first button at the toolbar "**Load Project**" (Figure 1).

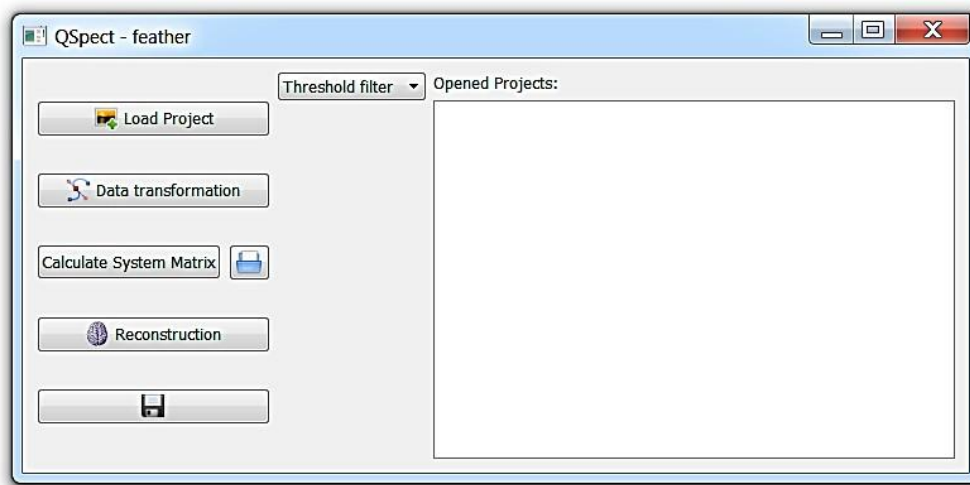


Figure 1: QSPECT's main menu on Windows distribution.

Initially, the data are loaded in the preview area and the user may select one of them by double clicking on it or may select all the images and then press "**Open**" (Figure 2). Then he/she will be asked if the selected data are "**Projections**" or "**Sinograms**" (Figure 3).

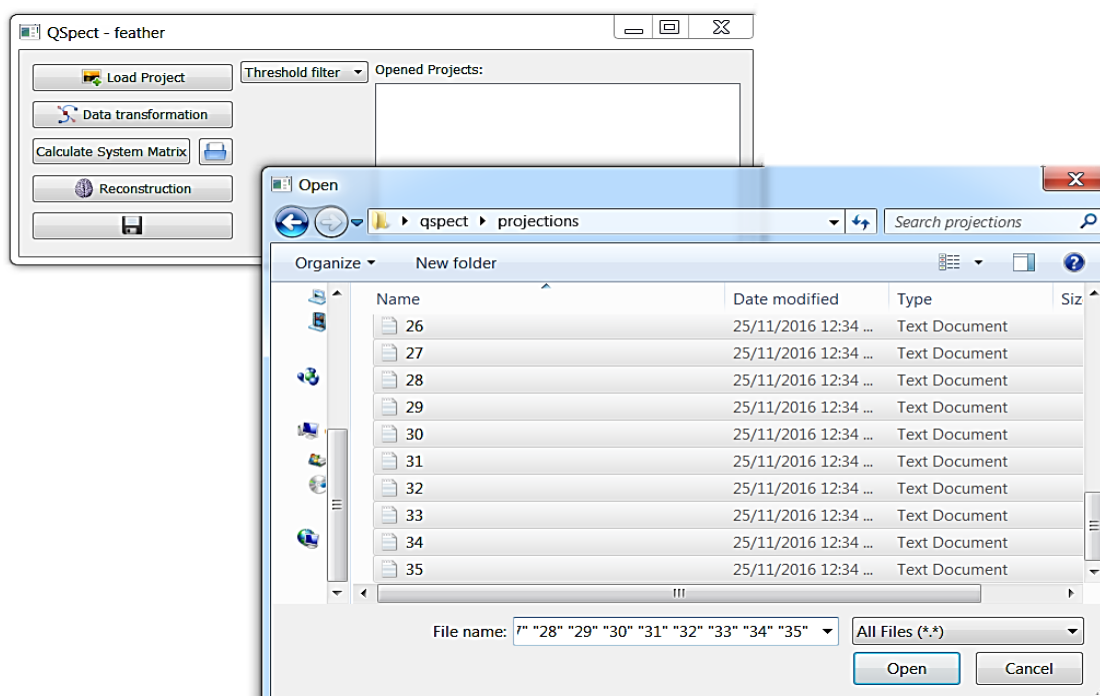


Figure 2. Selection of the files (projections or sinograms) in “Load Project”.

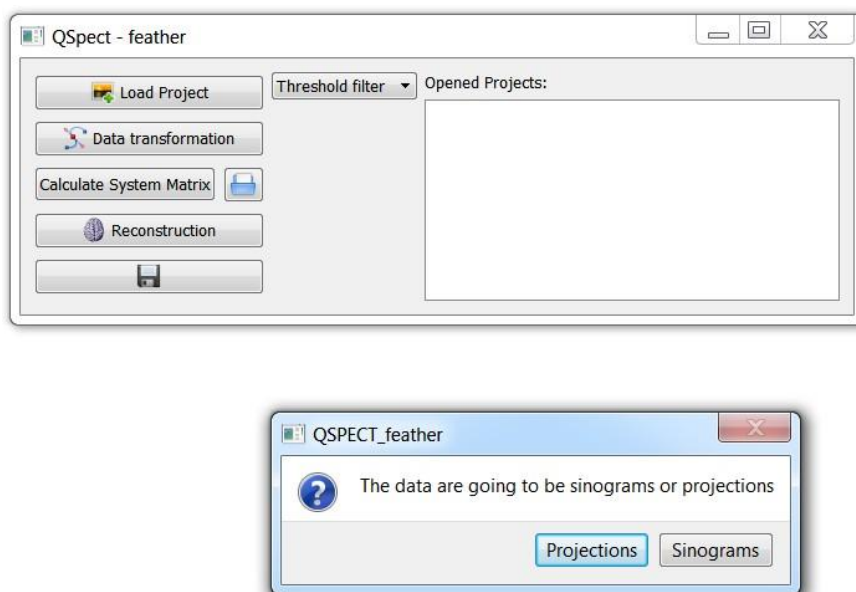


Figure 3. The user has to define if the selected data are Projections or Sinograms.

If the selected data are projections, then the user may select the “**Data transformation**” button from the toolbar to transform them into sinograms (Figure 4). A preview window opens to visualize the loaded projections and/or sinograms (Figure 5).

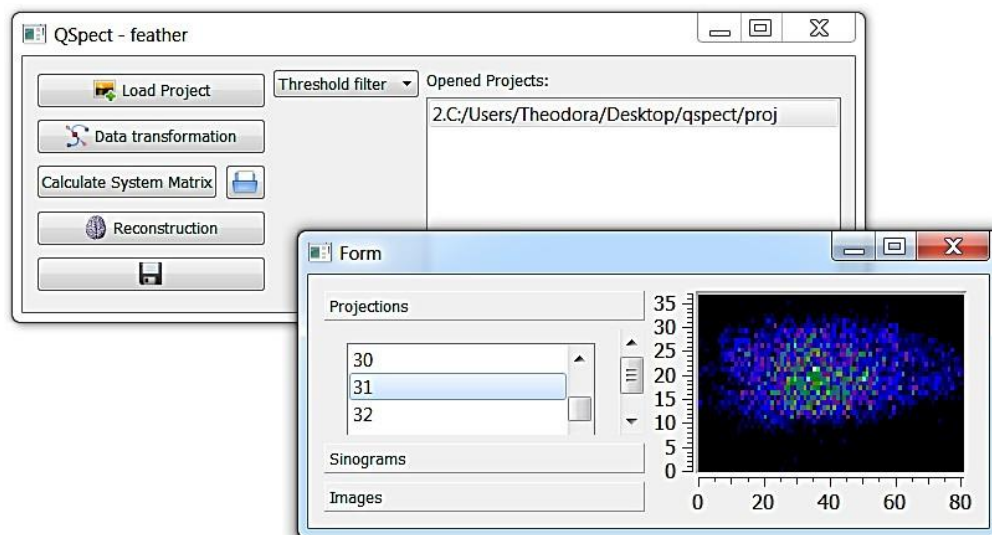


Figure 4. The user has to select “Data transformation” to transform projections into sinograms.

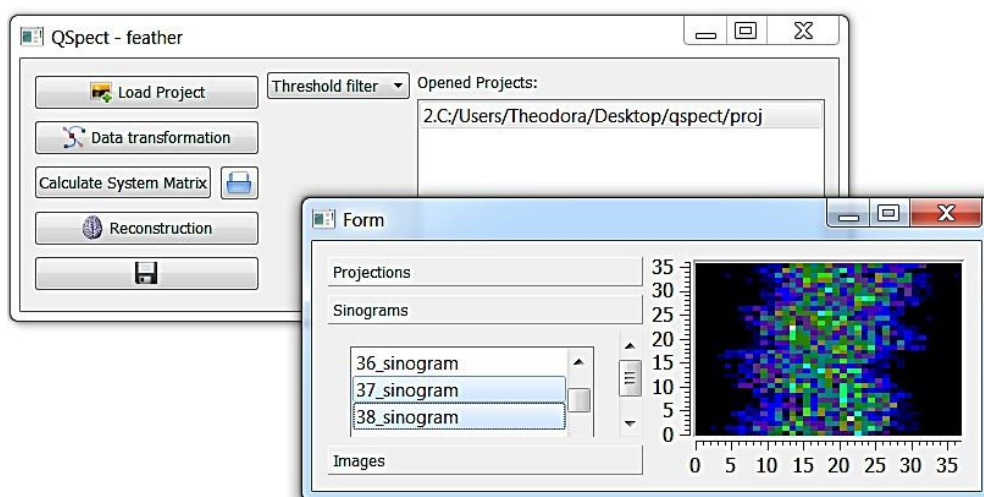


Figure 5. Visualization of the loaded projections and/or sinograms.

Construction of the probability system matrix

The probability system matrix is essential for the implementation of an iterative image reconstruction algorithm such as the MLEM. The user may start this function by selecting the option “**Calculate System Matrix**” from the toolbar (Figure 6).

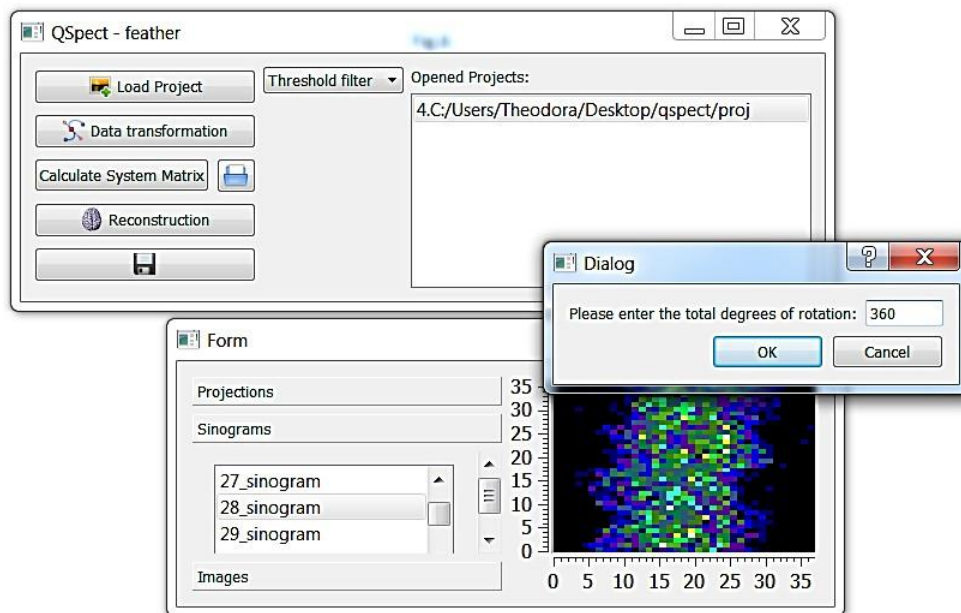


Figure 6. Create the System Matrix and select the total degrees of rotation.

Reconstruct & export SPECT data

After loading a project and defining a System Matrix, the user may proceed to the reconstruction of the data by clicking the button “**Reconstruction**” at the toolbar. The user is asked to enter the number of iterations for the image reconstruction. Additionally, a predefined filter (by the user) can be optionally loaded in this step, by clicking the checkbox “**Use kernel filter**” (Figure 7).

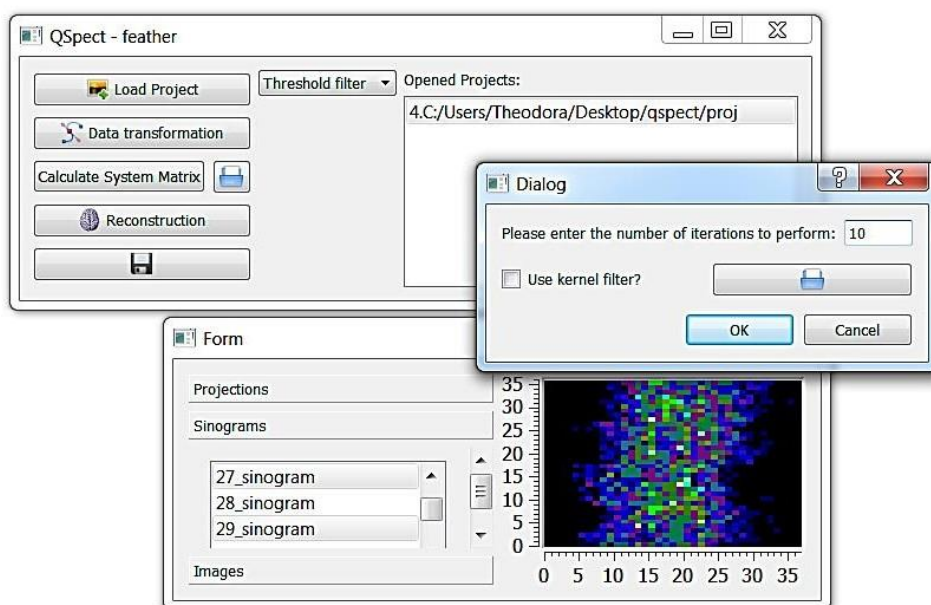
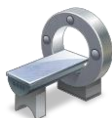



Figure 7. When proceeding to the reconstruction, the number of iterations need to be defined.



After finishing the reconstruction process, the user may save the data by clicking the last button  at the toolbar. Then, he/she will be asked to save the data as Projections / Sinograms / Images in a selected output directory.

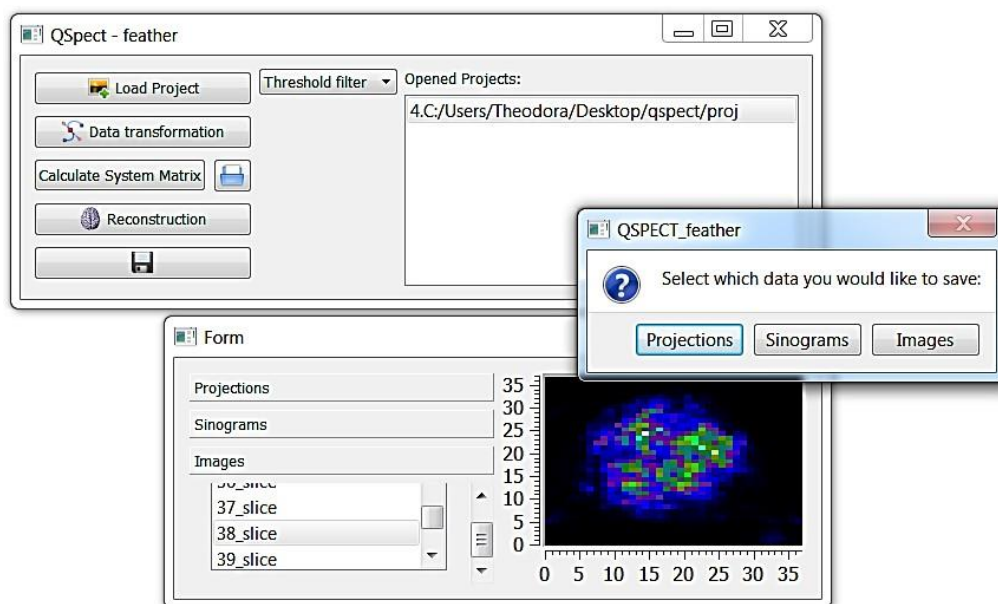


Figure 8. Save your selected data in an output directory as Projections / Sinograms / Images.

III. Benchmark examples

Alongside the source code of QSPECT, 2 benchmark example projects are provided, to test the reconstruction procedure. The projections, the sinograms and the reconstructed images are given. Thus, test these 2 datasets following these steps:

- i. Load the projections from the folder “Projections”
- ii. Transform the projections into sinograms
- iii. Create the System Matrix with 360°
- iv. Reconstruct the data with 15 iterations
- v. Save the new data in your own directory
- vi. Compare your outputs with the provided benchmarks' images

Benchmark-1: Preclinical pixelated crystal (MOBY – GATE)

Benchmark-1 includes the projections of a mouse model resulted from simulated imaging acquisition (projections=36, total rotation= 360° , step= 10°) with GATE Monte Carlo simulation toolkit. The MOBY computational phantom was used to simulate the biodistribution of ^{99m}Tc -HMPAO radiopharmaceutical and a small animal pixelated SPECT camera (head: 81×37 crystals). Figures 9, 10 and 11 shows indicative pictures of the projections, sinograms and images respectively.

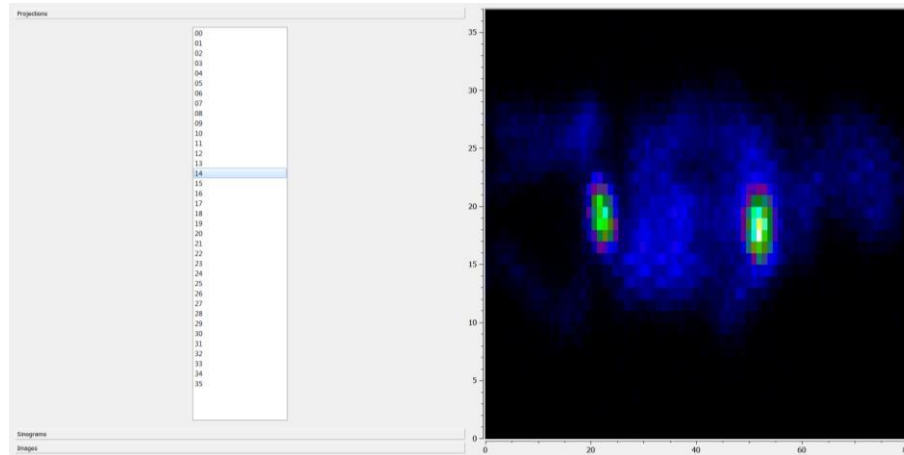
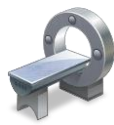


Figure 9. Visualization of the 14th loaded projection.

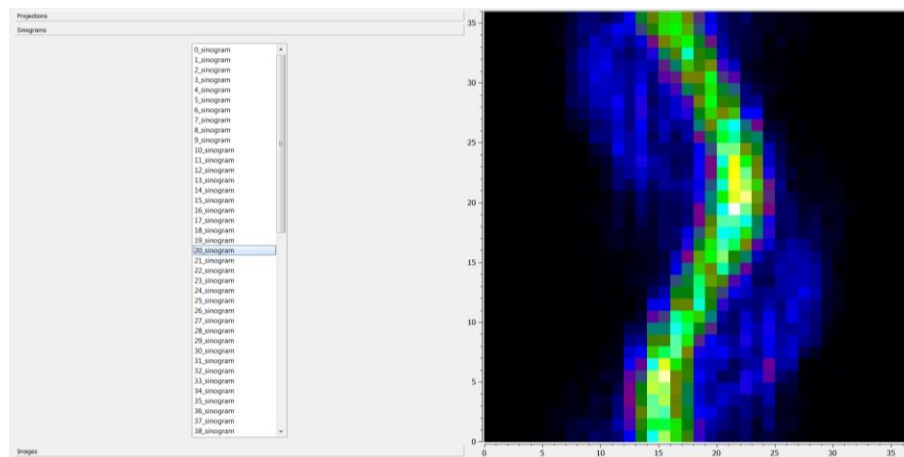


Figure 10. Visualization of the 20th sinogram.

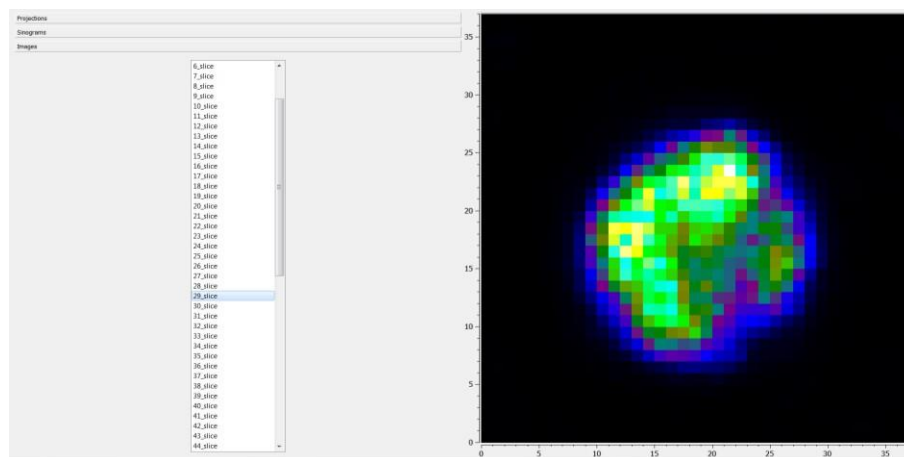
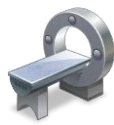


Figure 11. Visualization of the 29th image.

Benchmark-2: Clinical homogeneous crystal (XCAT – GATE)

Benchmark-2 includes the projections of a human male model, resulted from a simulated imaging acquisition (projections=36, total rotation=360°, step=10°) with



GATE Monte Carlo simulation toolkit, using the XCAT computational phantom. The SPECT scanner had a homogeneous crystal ($44.5 \times 59.1 \text{ cm}^2$). Figures 12, 13 and 14 shows indicative pictures of the projections, sinograms and images respectively.

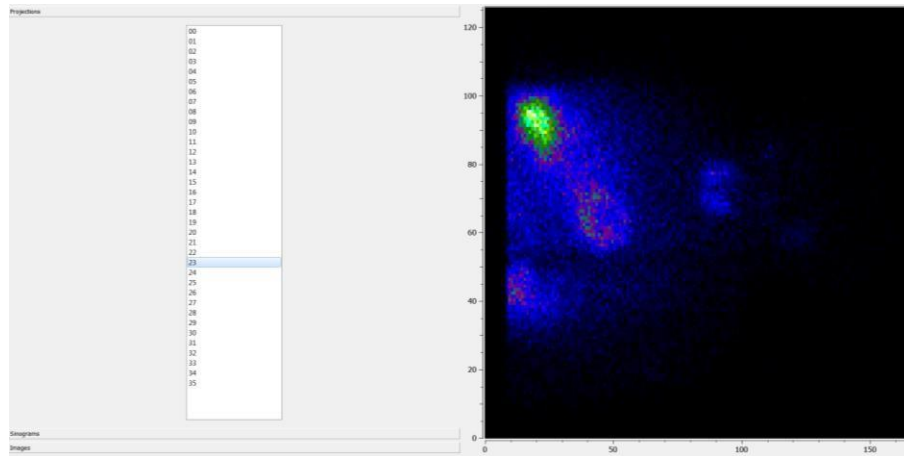


Figure 12. Visualization of the 23rd loaded projection.

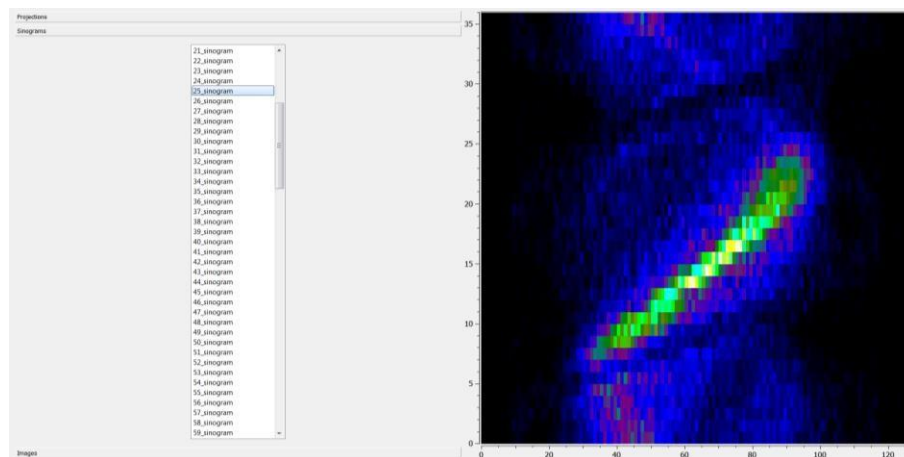


Figure 13. Visualization of the 25th sinogram.

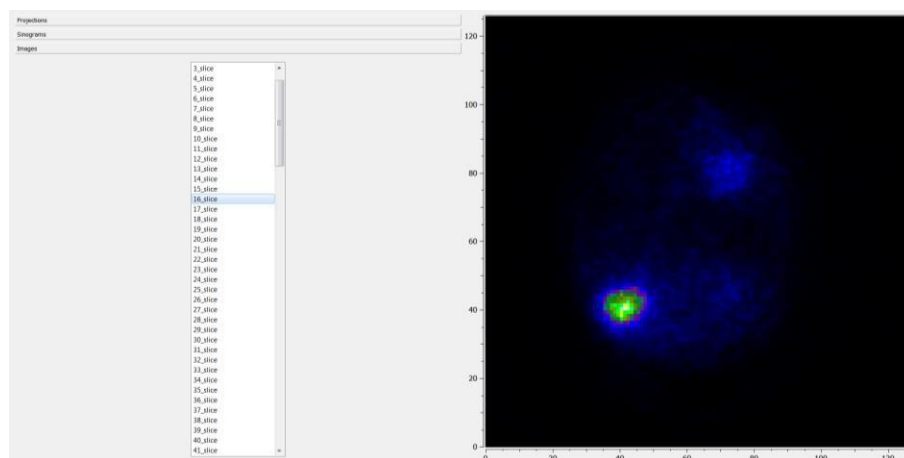
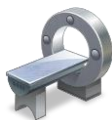


Figure 14. Visualization of the 16th image.



IV. Notes

1. All the data may be in txt file format and may have sequential numbering and the same number of digits.
2. The sinograms may be perpendicular to the preview, otherwise the user may rotate them before insert them.
3. Two extra scripts are provided for converting directly the root files into projections (.txt files). "**Projections_homogeneous_crystal.c**" is useful for simulations with SPECT scanners with homogeneous crystal as a detector, while the "**Projection_pixelated_crystal.c**" stands for simulations with pixelated detectors (notes are included inside the script).
4. Known bug which will be corrected in the next release: After the insertion of the projections, the user may click on the path on the window named "**Opened Projects**" before transforming the data to sinograms.

IV. Appendix "Parameters need to be defined in projections .c scripts"

Before compiling and executing the "**Projections_XX_crystal.c**" files, the user must define the following parameters according to his/her simulation characteristics.

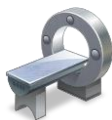
I. For Pixelated Crystals

```
int number_of_heads = X; // Define the number of head
int number_of_steps = X; // Define the number of rotation steps (Total Time
                          / Time Slice)
int number_of_pixels_x = X; // Define the number of pixels in x dimension
int number_of_pixels_z = X; // Define the number of pixels in z dimension
float min_energy = X; // Define the minimum energy (in MeV)
float max_energy = X; // Define the maximum energy (in MeV)
```

In the "main loop" the steps of the rotation are defined using the rotation angles.

II. For Homogeneous Crystals

```
int number_of_heads = X; // Define the number of heads
int number_of_steps = X; // Define the number of rotation steps (Total Time
                          / Time Slice)
int head_x_dimention = X; // Define the x-length of the crystal (in mm)
int head_y_dimention = X; // Define the y-length of the crystal (in mm)
float pixel_x_size = X; // Define the size of the pixel in x dimension (mm)
float pixel_y_size = X; // Define the size of the pixel in y dimension (mm)
float min_energy = X; // Define the minimum energy (in MeV)
float max_energy = X; // Define the maximum energy (in MeV)
int energy_peak_centroid = X; // Define the centroid energy
int energy_window = X; // Define the energy window
```



- 1) Compile using this command line in the terminal:
g++ (name of file) `root-config --cflags --libs
- 2) After compiling the code execute using the following command line:
./a.out 'directory name of root files'