D2.1

Simulation Framework

Project information

Project full title	MHz rate mulTiple prOjection X-ray MicrOSCOPY
Project acronym	MHz-TOMOSCOPY
Grant agreement no.	101046448
Instrument	EIC Pathfinder Open
Duration	48 months
Website	https://tomoscopy.eu/

Deliverable information

Deliverable no.	D2.1
Deliverable title	Simulation Framework
Deliverable responsible	EuXFEL
Related Work-Package/Task	WP2, Task 2.1
Type (e.g. report; other)	Software
Author(s)	Sarlota Birnsteinova, Ilia Petrov, Pablo Villanueva Perez
Dissemination level	SEN
Document Version	1.0
Date	2023-05-31
Download page	https://github.com/sarlotabirnsteinova/MhzTomoSimulations.git [Private repository]

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No. 101046448

Document information

Version no.	Date	Author(s)	Comment
1.0	2023-05-30	S. Birnsteinova, I. Petrov, P. Villanueva Perez	
1.1	2023-05-31	P. Vagovic, J. Marauska	Review
1.2	2023-06-01	S. Birnsteinova, I. Petrov, P. Villanueva Perez	Review and finalisation

Executive Summary

This report summarizes the realizations of a realistic 4D simulation framework for X-ray multi-projection imaging. The generated datasets will be validated with the experimental measurements of WP1 (using the crystal selection and optical configuration from task 1.1) and will be used to general data for Machine Learning (ML) training in task 2.3.

The framework takes as initial input the XFEL illumination from the XFEL Photon Pulse Database, operated by the EuXFEL. These waves are propagated through the crystal splitters to the sample position. Then, the interaction of those waves and the sample is simulated via the projection approximation. The resulting waves are then propagated to a realistic detector used in the MHz-TOMOSCOPY project (Shimadzu HPV-X2).

The whole simulation accounts for coherent effects, noise, and artifacts arising from the stochastic nature of the XFEL SASE pulses.

This simulation package will be used in conjunction with task 3.2 to generate realistic data coming from the applications of WP3. Such data will be used for ML training.

The rest of this document summarizes this simulation framework's main components and pipeline.

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No. 101046448

Table of Contents

This project has received funding from the European Union’s Horizon Europe research and innovation programme under grant agreement No. 101046448

Simulation Pipeline

1. XFEL illumination

Simulation of the XFEL illumination is done by using XFEL Photon Pulses Database[1], operated by the EuXFEL and wpg library[2], considering the optical configuration specified in WP1.1 and specific of SPB/SFX beamline.

2. Crystals

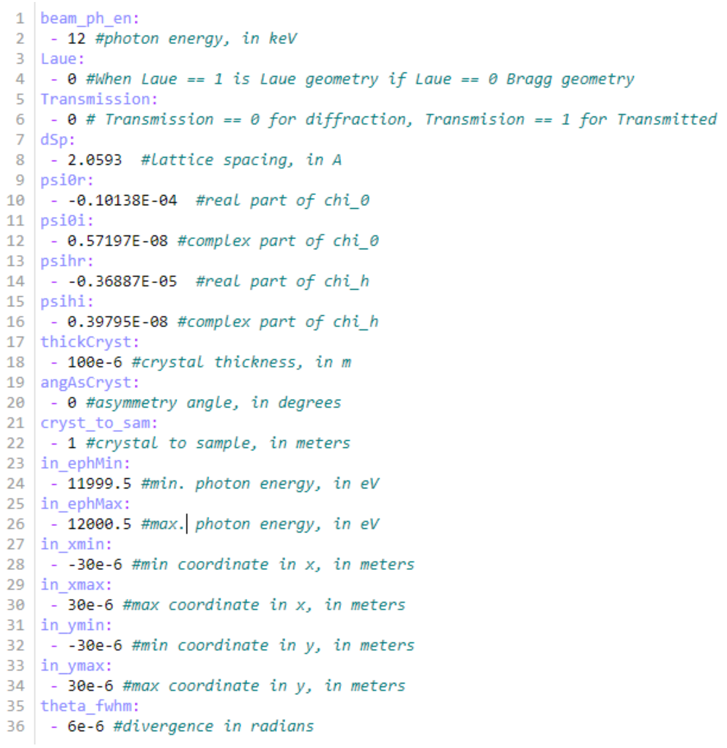
The user defines the crystal parameters in cryst_input.yaml. The susceptibility data can be accessed at the X-ray server[3]. The three-dimensional electric field should be defined as a WPG or SRW wavefront. For each photon energy, a two-dimensional electric field needs to be defined. The package simulates the diffraction of the provided wavefront in a crystal. As a result, the three-dimensional electric field after diffraction is provided.

[1] [citePPD] Maurizio Manetti, Alexey Buzmakov, Liubov Samoylova, Evgeny Schneidmiller, Harald Sinn, Janusz Szuba, Krzysztof Wrona, Mikhail Yurkov; FAST-XPD: XFEL photon pulses database for modeling XFEL experiments. *AIP Conference Proceedings* 15 January 2019; 2054 (1): 030019.

[2] [citeWPG] Samoylova, L., Buzmakov, A., Chubar, O. & Sinn, H. WavePropaGator: Interactive framework for X-ray FEL optics design and simulations. // Journal of Applied Crystallography 08/2016; 49(4) pp. 1347-1355.

[3] [stepanov] https://x-server.gmca.aps.anl.gov/

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No. 101046448



```
beam_ph_en:
 - 12 #photon energy, in keV
Laue:
 - 0 #When Laue == 1 is Laue geometry if Laue == 0 Bragg geometry
Transmission:
 - 0 # Transmission == 0 for diffraction, Transmision == 1 for Transmitted
dSp:
 - 2.0593  #lattice spacing, in A
psi0r:
 - -0.10138E-04  #real part of chi_0
psi0i:
 - 0.57197E-08 #complex part of chi_0
psihr:
 - -0.36887E-05  #real part of chi_h
psihi:
 - 0.39795E-08 #complex part of chi_h
thickCryst:
 - 100e-6 #crystal thickness, in m
angAsCryst:
 - 0 #asymmetry angle, in degrees
cryst_to_sam:
 - 1 #crystal to sample, in meters
in_ephMin:
 - 11999.5 #min. photon energy, in eV
in_ephMax:
 - 12000.5 #max. photon energy, in eV
in_xmin:
 - -30e-6 #min coordinate in x, in meters
in_xmax:
 - 30e-6 #max coordinate in x, in meters
in_ymin:
 - -30e-6 #min coordinate in y, in meters
in_ymax:
 - 30e-6 #max coordinate in y, in meters
theta_fwhm:
 - 6e-6 #divergence in radians
```

Figure 1. A sample input with the crystal parameters.

The current installation of WPG available at DESY maxwell server at */afs/desy.de/group/exfel/software/wpg/openmp/* allows simulating Bragg diffraction with any asymmetry for the crystal that is oriented such that Bragg's condition is fulfilled at *bem_ph_en* .

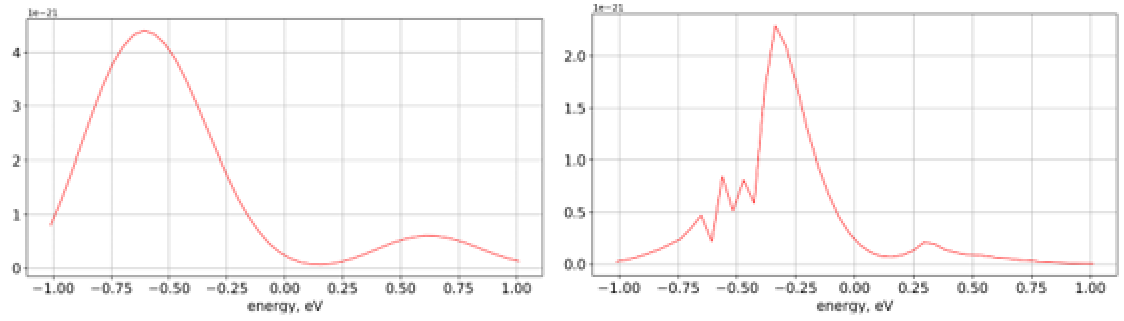


Figure 2. Pulse intensity at various photon energy before (left) and after (right) diffraction in C*(220) 100 um crystal at 12 keV in Bragg case.

In Figure 2, a sample spectrum before and after diffraction is shown. Here, the exact Bragg condition is fulfilled at a photon energy in the valley between the spectral peaks. Therefore, there is diffraction only at the tails of the diffraction curve. By calculating the ratio of the integrals of spectra, one can estimate the effectivity of the given splitter crystal. The obtained wavefronts are then propagated to the sample positions given the geometry from task 1.1.

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No. 101046448

3. Creating a phantom

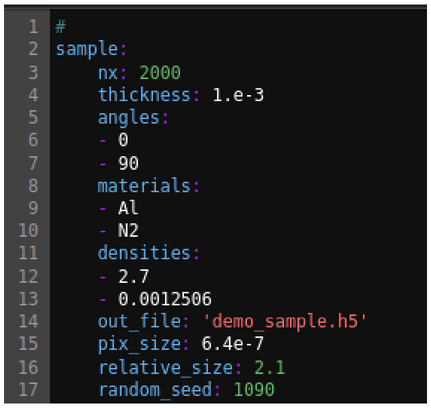
There are two possibilities for how one can approach defining a sample to propagate. The first one is that the user will produce a file with an array of projected thicknesses of all materials at given projection angles. Specification of materials and densities is necessary and will be used later to produce the transmission function of the sample. Additional information, such as pixel size, should be included.

Another option is to use the provided function for the generation of foam samples. The following library should be installed [foam_ct][1], [foam_ct_doc][2]. Files provided in the foam_changes directory should replace the same-named counterparts in [foam_ct_git][3], which enable to return projected thickness of the phantom instead of already propagated data in the original library. After replacing files with provided changed versions, installation of the library follows the instruction [foam_ct_git][3].

For the generation of foam phantoms, several parameters need to be given in the yaml config file. Array size, labeled as *nx*, and projection *angles* need to be specified. It will lead to producing square arrays containing projections of a cylinder with voids for given angles. Parameter *thickness* here refers to the diameter of the cylinder.

Materials and *densities* are for projections in the following order: cylinder and voids with densities in g/cm^3. Materials are passed as a string with the xraydb library format [xraydb][4]. Produced phantom will be saved in file *out_file.* The pixel size is given by the parameter *pix_size* in meters*.* The relative size will lead to the cylinder taking 2/relative_size-th of the width of an image *nx.* Random seeds can be specified for the generation of void positions. All details concerning the foam phantom itself can be found [foam_ct][1].

As an example, we show parameters, Fig. 3, for making a foam sample using the foam_ct_phantom library used in the *make_foam_sample* function, shown in Figure 5. In Figure 4, the projection of the cylinder is shown for parameters values specified in Figure 3.



```yaml
#
sample:
    nx: 2000
    thickness: 1.e-3
    angles:
    - 0
    - 90
    materials:
    - Al
    - N2
    densities:
    - 2.7
    - 0.0012506
    out_file: 'demo_sample.h5'
    pix_size: 6.4e-7
    relative_size: 2.1
    random_seed: 1090
```

Figure 3. An example of input parameters needed for creating a foam sample.

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No. 101046448

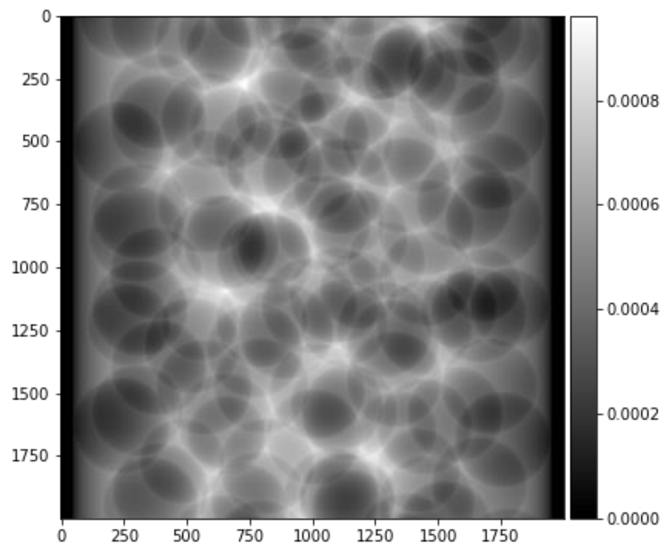
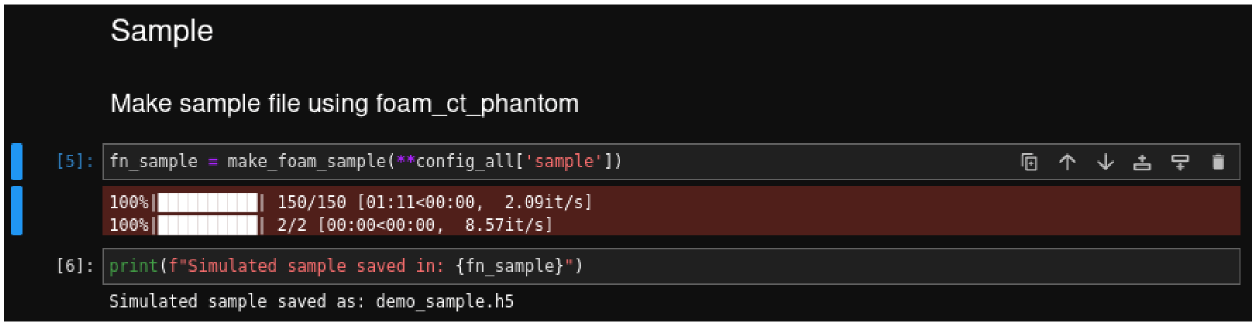


Figure 4. An example of cylinder projection of foam sample with input parameters is displayed in *Figure 3*.



Sample

Make sample file using foam_ct_phantom

```
[5]: fn_sample = make_foam_sample(**config_all['sample'])

100%|██████████| 150/150 [01:11<00:00,  2.09it/s]
100%|██████████| 2/2 [00:00<00:00,  8.57it/s]

[6]: print(f"Simulated sample saved in: {fn_sample}")

Simulated sample saved as: demo_sample.h5
```

Figure 5. An example of creating a foam sample using the *make_foam_sample* function with the input parameters shown in *Figure 3*.

The function *make_foam_sample* returns the file name, in which the results of projections of a simulated sample are saved. Such a file contains additional information such as pixel size, materials, densities, etc. This file will later be used as an input to the propagation class.

4. Propagation of a sample

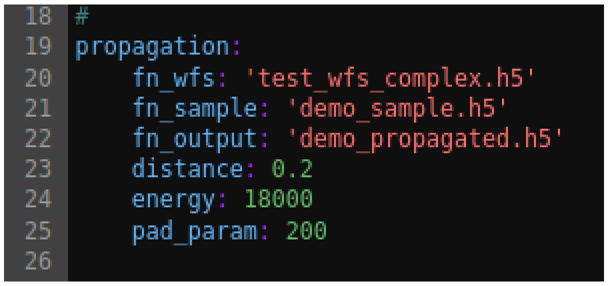
Propagation of the sample is realized in the SamplePropagation class. Several inputs are needed, specified in the YAML file. Three file names are passed as parameters: i) file containing saved wavefronts, named *fn_wfs*, ii) file with sample information *fn_sample*, described in Section 3, and iii) file name under which propagated results will be saved, *fn_output*. One needs to specify propagation distance in meters and photon energy in eV. However, if *fn_wfs* contains information about photon energy, this value will be used in propagation and will rewrite energy from the YAML file. However, this parameter can be changed using the *change_energy()* method of the class. Parameter *pad_param* gives information about padding used in the propagation of arrays.

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No. 101046448

Input wavefronts are in the form of complex arrays, and the information about the pixel size and energy can also be stored. If this is not the case, settings from the sample file will be used. Moreover, if wavefronts are stored in WPG format, the function *read_save_wpg()* is included to save wpg wavefronts directly as desired complex array. The function will also save information from the WPG file about pixel size and photon energy.

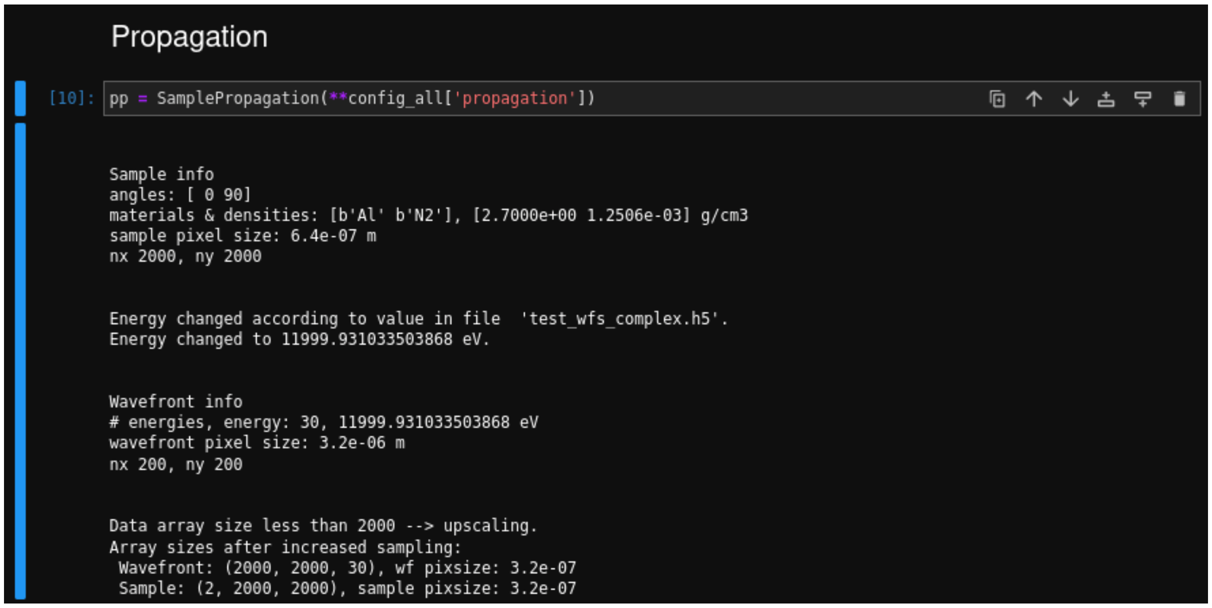
After creating an instance of the Sample Propagation class, a sample transmission function is generated using the xraydb library [xraydb]. Arrays containing transmission functions and wavefronts are adjusted according to values of corresponding pixel sizes, and a combined complex array is created for further propagation. If the combined array is smaller than size (2000,2000), the array will be upscaled to that size, leading to an adjustment of pixel size as can be seen in Figure 6.

Figure 6 shows parameters used for creating an instance of SamplePropagation class, whereas creation of such instance is captured in Figure 7.



```
#
propagation:
    fn_wfs: 'test_wfs_complex.h5'
    fn_sample: 'demo_sample.h5'
    fn_output: 'demo_propagated.h5'
    distance: 0.2
    energy: 18000
    pad_param: 200

```

Figure 6. An example of input parameters needed for creating an instance of SamplePropagation Class.

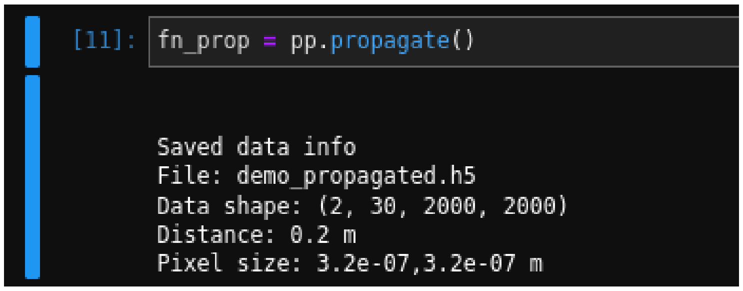


Propagation

```
[10]: pp = SamplePropagation(**config_all['propagation'])


Sample info
angles: [ 0 90]
materials & densities: [b'Al' b'N2'], [2.7000e+00 1.2506e-03] g/cm3
sample pixel size: 6.4e-07 m
nx 2000, ny 2000


Energy changed according to value in file  'test_wfs_complex.h5'.
Energy changed to 11999.931033503868 eV.


Wavefront info
# energies, energy: 30, 11999.931033503868 eV
wavefront pixel size: 3.2e-06 m
nx 200, ny 200


Data array size less than 2000 --> upscaling.
Array sizes after increased sampling:
 Wavefront: (2000, 2000, 30), wf pixsize: 3.2e-07
 Sample: (2, 2000, 2000), sample pixsize: 3.2e-07
```

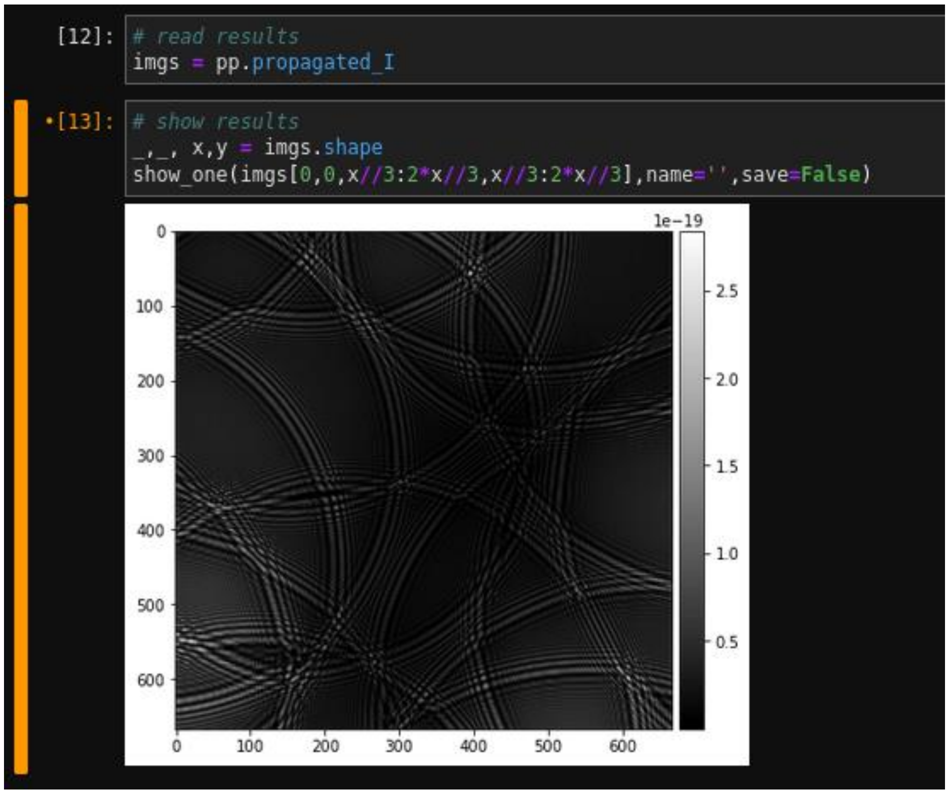
Figure 7. An example of creating an instance of SamplePropagation class.

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No. 101046448



```
[11]: fn_prop = pp.propagate()

      Saved data info
      File: demo_propagated.h5
      Data shape: (2, 30, 2000, 2000)
      Distance: 0.2 m
      Pixel size: 3.2e-07,3.2e-07 m
```

Figure 8. An example of using propagate method.



```
[12]: # read results
      imgs = pp.propagated_I

•[13]: # show results
      _,_, x,y = imgs.shape
      show_one(imgs[0,0,x//3:2*x//3,x//3:2*x//3],name='',save=False)
```

Figure 9. An example of accessing propagated data and plotting.

The next step is propagating the combined wave using the method *propagate()*, shown in Figure 8, which automatically saves the result to the *fn_output file*.

Propagated data can be accessed through a class instance, and a simple plotting function is provided, *show_one()*, as shown in Figure 9.

5. Detector

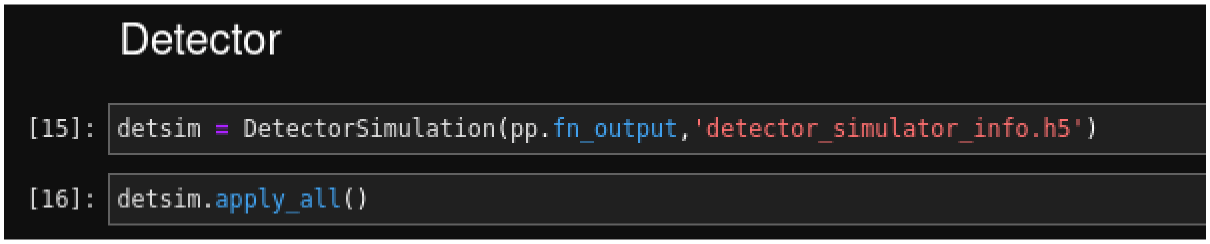
The last part contains a simulation of the detector effects. Two input parameters are needed for the creation of an instance of DetectorSimulation class. The first input contains the file name, named *fn_propagated*, with propagated intensities and other information stored in the SamplePropagation output file. The second input parameter is the file name, *fn_detector_info*,

This project has received funding from the European Union’s Horizon Europe research and innovation programme under grant agreement No. 101046448

with parameters of the simulated detector. The file is provided and includes data for Shimadzu HPV-X2 high frame-rate camera. This file stores data with the pixel size of the detector, image dimensions, and parameters from dark and flat measurements. The last parameter to pass to SamplePropagation class is the optional file name in which simulated data will be saved.

To simulate the effect of the detector method apply_all() is used. It contains the application of a point spread function, estimation of dark field and flat field using Poisson distribution, and the stored parameters. The final step is rescaling the data to 16-bit depth, the same as the saved data from the Shimadzu HPV-X2 camera. Simulated data will be saved in parameter fn_output.

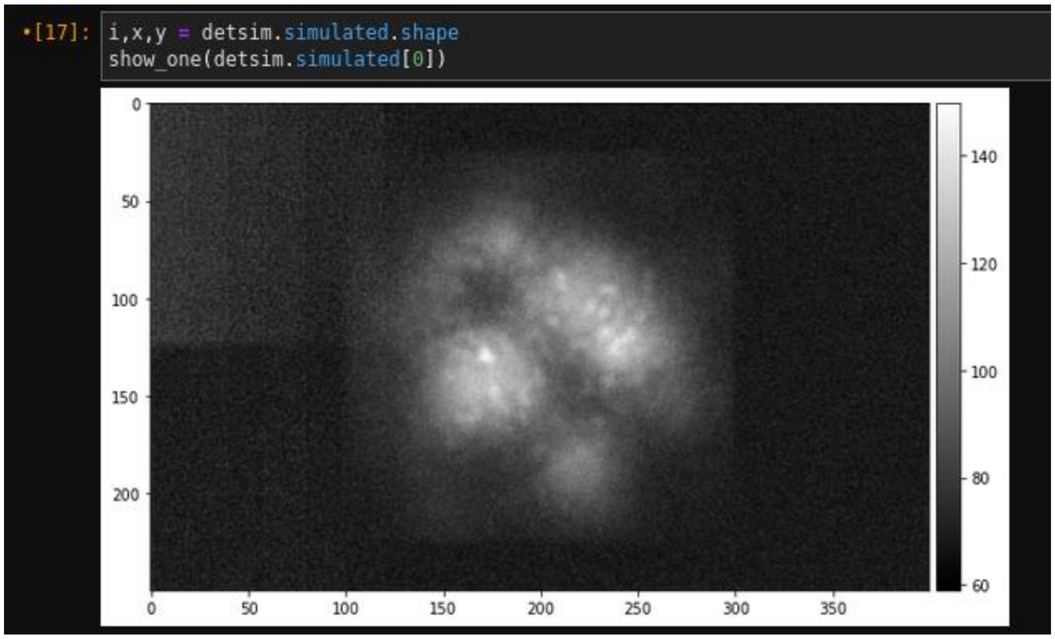
In Figure 10Figure 1 the creation of a detector simulation instance is shown, and Figure 11 contains the application of simulation functions and examples of simulated data in the detector.



Detector

```
[15]: detsim = DetectorSimulation(pp.fn_output,'detector_simulator_info.h5')
[16]: detsim.apply_all()
```

Figure 10. An example of creating an instance of DetectorSimulation class and applying functions to simulate detector effects.



```
•[17]: i,x,y = detsim.simulated.shape
       show_one(detsim.simulated[0])
```

Figure 11. An example of accessing and plotting data after applying detector effects.

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No. 101046448

Bibliography

1. [foam_ct] Pelt, D. M., Hendriksen, A. A., & Batenburg, K. J. (2022). Foam-like phantoms for comparing tomography algorithms. Journal of Synchrotron Radiation, 29(1), 254-265.
2. [foam_ct_doc] https://dmpelt.github.io/foam_ct_phantom/
3. [foam_ct_git] https://github.com/dmpelt/foam_ct_phantom
4. [xraydb] https://xraypy.github.io/XrayDB/, https://github.com/xraypy/XrayDB
5. [shimadzu] https://www.shimadzu.com/an/products/materials-testing/high-speed-video-camera/hyper-vision-hpv-x2/index.html

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No. 101046448