# pyXIT: Python GUI Software for CT Imaging, Learning and Experimental Algorithms; Example: Multi Material Phase Retrieval

M. Ullherr[a,*], S. Zabler[a,b], J. Graetz[b] and R. Hanke[a,b]

[a]Lehrstuhl für Röntgenmikroskopie, Universität Würzburg, Josef-Martin-Weg 63, 97074 Würzburg, Germany
[b]Fraunhofer Development Center X-Ray Technology EZRT, Josef-Martin Weg 63, 97074 Würzburg, Germany

software download: www.physik.uni-wuerzburg.de/en/lrm/research/software

*maximilian.ullherr@physik.uni-wuerzburg.de

## Overview

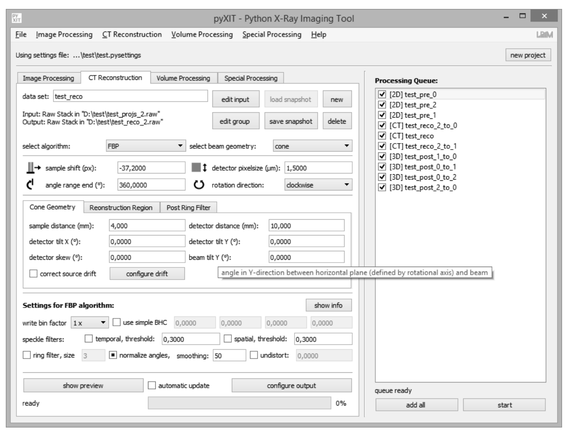
pyXIT (the Python X-Ray Imaging Tool) is a software for data processing in x-ray computed tomography (CT) imaging, intended for use in a scientific or development context.
It was developed because no CT imaging software existed that combined a GUI, fast preview and the ability to integrate existing Python code.
The software is ideally suited for experimental algorithms, hardware or software development and teaching/learning.
The programming language Python is easy to learn and suited for rapid development. Here, it is used as a common interface for algorithms implemented in different ways.
pyXIT is not intended to replace commercial software, as the use cases are different.

Comparable software: ASTRA* [1], pyHST* [2], tomopy* [3] and MuhRec [4] (* no GUI).

## Features

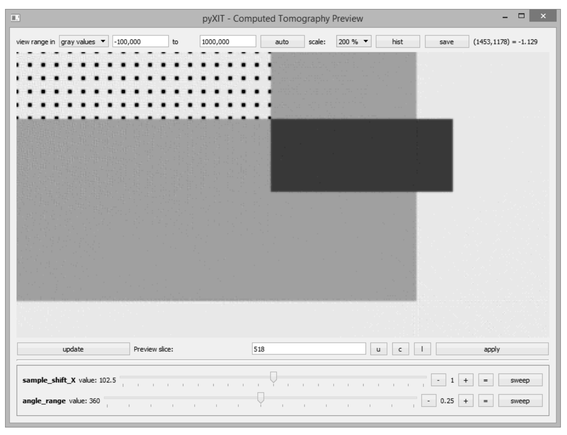
- Computed tomography reconstruction with cone FBP
- Plugin-based image filters for 2D and 3D (see right side), extendable by users
- Fast slice preview and parameter sweep for CT and image/volume processing
- Written in Python/PyQt and OpenCl, high CT reconstruction speed
- Multiple input formats (tif, raw, ...)
- Free/libre software: can be inspected and changed by users (FreeBSD license)
- Software parts can be used independently (e.g. OpenCl to Python API layer)

## User Interface




Main window, Cone FBP CT panel with the processing queue enabled

- Control hub, manage multiple data sets
- Set parameters for plugins/algorithms
- Tool tips explain user interface




CT preview window (simulated test data)

- Shows result preview
- Selectable area (slice or subvolume)
- Prior results are cached
- Automatic or manual update

## Plugin Example: Multi Material Phase Retrieval

This is an example of a custom algorithm which is made easily usable with a pyXIT plugin. It applies phase retrieval to inline phase contrast volume images with multiple materials present in the sample [5].
The sample is a rose stem; ESRF ID19, 1.1 µm pixelsize, 300x600 µm shown.

Top right: Single material phase retrieval produces long range blurring artifacts around the material with the higher absorption (black = calcium oxalate).
Bottom right: Our own algorithm.



## Image Enhancement and Artifact Correction

Enhancing the image quality or removing image artifacts makes the CT images easier to evaluate and use. Artifact correction is necessary in some cases.
The following plugins/features of pyXIT are especially useful for high resolution and inline phase contrast x-ray imaging:

- Wiener deconvolution
- Phase retrieval, multi material [5]
- Ring filters, partial and full rings
- Background variation removal

For the cone FBP CT reconstruction:

- Drift correction from rewind images
- Detector lens distortion correction
- Field of view extension (half-scan)

## Processing Examples

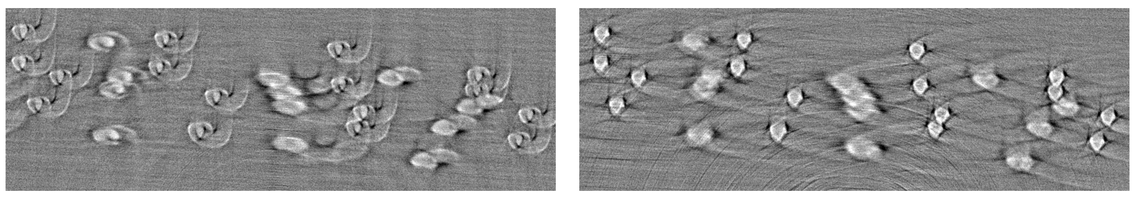
### Image enhancement for inline phase contrast CT (Click-CT scanner).

Volume slices of a polymer granulate for 3D printing.
0.62 µm voxelsize (2x binned), 400x400 µm shown.

Step 1: CT reconstruction (with simple ring filter) | Step 2: Phase retrieval + wiener deconvolution | Step 3: Slice ring filters (can be applied in CT)

### Drift correction (LRM-LMJ CT setup).

Volume slices of a polymer fiber mesh sample.
Extreme case, up to 30 pixels thermal drift and sample holder defect.



Blur from drifting sample or source | With drift correction from rewind images

## Implementation Details

### Reconstruction Speed for FBP

(GeForce 980 Ti, without file access)

| size | parallel | cone |
|---|---|---|
| 1000 | 105 $s^{-1}$ | 64 $s^{-1}$ |
| 2000 | 930 $min^{-1}$ | 540 $min^{-1}$ |
| 4000 | 125 $min^{-1}$ | 70 $min^{-1}$ |

Size is detector width and nprojs, speed is given in slices per time.

### Blockwise Volume Processing

Basic problem: The volume image is too large to fit into RAM due to memory overhead.
Solution: Divide the volume into overlapping blocks, process them in parallel. Speed up the FFT by choosing side lengths in $2^k 3^m 5^n$ ($k \geq 3$). Algorithms must have a known range, but need not be adjusted otherwise.

For both, file access is efficient and concurrent, up to 110 MB/s for Gbit LAN NAS.

## References

[1] W. van Aarle et al. In: *Ultramicroscopy* 157 (2015). DOI: 10.1016/j.ultramic.2015.05.002.
[2] A. Mirone et al. In: *NIM B* 324 (2014). DOI: 10.1016/j.nimb.2013.09.030.
[3] D. Gürsoy et al. In: *J. Synchr. Rad.* 21.5 (2014). DOI: 10.1107/S1600577514013939.
[4] A. P. Kaestner. In: *NIM A* 651.1 (2011). DOI: 10.1016/j.nima.2011.01.129.
[5] M. Ullherr and S. Zabler. In: *Optics Express* 23.25 (2015). DOI: 10.1364/OE.23.032718.

Presented at the DIR2019, International Symposium on Digital Industrial Radiology and Computed Tomography, 2 – 4 July 2019 in Fürth, Germany |  | Generated using LaTeX/tikzposter