Pore3D Documentation

Release 0.1.1

Sincrotrone Trieste S.C.p.A.

May 12, 2016

Contents

pore3d provides ..... Here is how to add a link to your documentation Docs and here is how to add a reference *[A1]*

CHAPTER 1

Features

- Example of how to write documentation

CHAPTER 2

Contribute

- Documentation: https://github.com/decarlof/pore3d/tree/master/doc
- Issue Tracker: https://github.com/decarlof/pore3d/docs/issues
- Source Code: https://github.com/decarlof/pore3d/

CHAPTER 3

Content

3.1 About

This section describes what the project is about pore3d.

3.2 Install

This section covers the basics of how to download and install pore3d.

Contents:

3.2.1 Installing from source

Clone the pore3d from GitHub repository:

```
git clone https://github.com/decarlof/pore3d.git pore3d
```

then:

```
cd project
python setup.py install
```

3.3 API reference

project Modules:

3.3.1 pore3d.io.tdf

Module .....

Functions:

parse_metadata(f, xml_command)	Fill the specified HDF5 file with metadata according to the DataExchange initia
read_tomo(dataset, index)	Extract the tomographic projection at the specified relative index from the HDF
parse_metadata(f, xml_command)	Fill the specified HDF5 file with metadata according to the DataExchange initia
read_sino(dataset, index)	Extract the sinogram at the specified relative index from the HDF5 dataset.
write_tomo(dataset, index, im)	Modify the tomographic projection at the specified relative index from the HDF
write_sino(dataset, index, im)	Modify the sinogram at the specified relative index from the HDF5 dataset with
get_nr_projs(dataset)	Get the number of projections of the input dataset.
get_nr_sinos(dataset)	Get the number of sinograms (or slices) of the input dataset.
get_det_size(dataset)	Get the width of the detector (nr of pixels) of the input dataset.
get_dset_shape(det_size, fov_height, nr_proj)	Get the shape of the dataset by arranging the input parameters.
get_dset_chunks(det_size)	Get a good chunk combination.

pore3d.io.tdf.**parse_metadata**(*f*, *xml_command*)

Fill the specified HDF5 file with metadata according to the DataExchange initiative. The metadata in input are described in a XML format.

Parameters

- **f** (*HDF5 file*) – HDF5 file open with h5py API
- **xml_command** (*string*) – Immaginary part of the complex X-ray refraction index.

pore3d.io.tdf.**read_tomo**(*dataset*, *index*)

Extract the tomographic projection at the specified relative index from the HDF5 dataset.

Parameters

- **dataset** (*HDF5 dataset*) – HDF5 dataset as returned by the h5py API.
- **index** (*int*) – Relative position of the tomographic projection within the dataset.

pore3d.io.tdf.**parse_metadata**(*f*, *xml_command*)

Fill the specified HDF5 file with metadata according to the DataExchange initiative. The metadata in input are described in a XML format.

Parameters

- **f** (*HDF5 file*) – HDF5 file open with h5py API
- **xml_command** (*string*) – Immaginary part of the complex X-ray refraction index.

pore3d.io.tdf.**read_sino**(*dataset*, *index*)

Extract the sinogram at the specified relative index from the HDF5 dataset.

Parameters

- **dataset** (*HDF5 dataset*) – HDF5 dataset as returned by the h5py API.
- **index** (*int*) – Relative position of the sinogram within the dataset.

pore3d.io.tdf.**write_tomo**(*dataset*, *index*, *im*)

Modify the tomographic projection at the specified relative index from the HDF5 dataset with the image passed as input.

Parameters

- **dataset** (*HDF5 dataset*) – HDF5 dataset as returned by the h5py API.
- **index** (*int*) – Relative position of the tomographic projection within the dataset.
- **im** (*array_like*) – Image data as numpy array.

pore3d.io.tdf.**write_sino**(*dataset*, *index*, *im*)
Modify the sinogram at the specified relative index from the HDF5 dataset with the image passed as input.

Parameters

- **dataset** (*HDF5 dataset*) – HDF5 dataset as returned by the h5py API.
- **index** (*int*) – Relative position of the sinogram within the dataset.
- **im** (*array_like*) – Image data as numpy array.

pore3d.io.tdf.**get_nr_projs**(*dataset*)
Get the number of projections of the input dataset.

Parameters **dataset** (*HDF5 dataset*) – HDF5 dataset as returned by the h5py API.

pore3d.io.tdf.**get_nr_sinos**(*dataset*)
Get the number of sinograms (or slices) of the input dataset.

Parameters **dataset** (*HDF5 dataset*) – HDF5 dataset as returned by the h5py API.

pore3d.io.tdf.**get_det_size**(*dataset*)
Get the width of the detector (nr of pixels) of the input dataset.

Parameters **dataset** (*HDF5 dataset*) – HDF5 dataset as returned by the h5py API.

pore3d.io.tdf.**get_dset_shape**(*det_size*, *fov_height*, *nr_proj*)
Get the shape of the dataset by arranging the input parameters.

Parameters

- **det_size** (*int*) – Width of the detector.
- **fov_height** (*int*) – Height of the FOV, i.e. the number of sinograms (or slices) of the dataset.
- **nr_proj** (*int*) – Number of collected projections.

pore3d.io.tdf.**get_dset_chunks**(*det_size*)
Get a good chunk combination. This function needs improvement...

Parameters **det_size** (*int*) – Width of the detector.

3.3.2 pore3d.phaseretrieval.phase_retrieval

Module .....

Functions:

prepare_plan(im, beta, delta, energy, ...[, ...])	Pre-compute data to save time in further execution of phase_retrieval
phase_retrieval(im, plan[, method, nr_threads])	Process a tomographic projection image with the selected phase retrieval alg

pore3d.phaseretrieval.phase_retrieval.**prepare_plan**(*im*, *beta*, *delta*, *energy*, *distance*, *pixsize*, *method=1*, *padding=False*)
Pre-compute data to save time in further execution of phase_retrieval

Parameters

- **im** (*array_like*) – Image data as numpy array. Only image size (shape) is actually used.
- **beta** (*double*) – Immaginary part of the complex X-ray refraction index.

- **delta** (*double*) – Decrement from unity of the complex X-ray refraction index.
- **energy [KeV]** (*double*) – Energy in KeV of the incident X-ray beam.
- **distance [mm]** (*double*) – Sample-to-detector distance in mm.
- **pixsize [mm]** (*double*) – Size in mm of the detector element.
- **method** (*int*) – Phase retrieval algorithm {1 = TIE (default), 2 = Born, 3 = Rytov, 4 = Wu}
- **padding** (*bool*) – Apply image padding to better process the boundary of the image

pore3d.phaseretrieval.phase_retrieval.**phase_retrieval**(*im*, *plan*, *method=1*, *nr_threads=2*)

Process a tomographic projection image with the selected phase retrieval algorithm.

Parameters

- **im** (*array_like*) – Flat corrected image data as numpy array.
- **plan** (*structure*) – Structure with pre-computed data (see prepare_plan function)
- **method** (*int*) – Phase retrieval algorithm {1 = TIE (default), 2 = Born, 3 = Rytov, 4 = Wu}
- **nr_threads** (*int*) – Number of threads to be used in the computation of FFT by PyFFTW

3.3.3 pore3d.postprocess.postprocess

Module .....

Functions:

postprocess(im, convert_opt, crop_opt)	Post-process a reconstructed image.

pore3d.postprocess.postprocess.**postprocess**(*im*, *convert_opt*, *crop_opt*)

Post-process a reconstructed image.

Parameters

- **im** (*array_like*) – Image data as numpy array.
- **convert_opt** (*string*) – String containing degradation method (8-bit or 16-bit) and min/max rescaling value (e.g. "linear8:-0.01;0.01"). In current version only "linear" for 16-bit and "linear8" are implemented.
- **crop_opt** (*double*) – String containing the parameters to crop an image separated by : with order top, bottom, left, right. (e.g. "100:100:100:100")

3.3.4 pore3d.preprocess.extfov_correction

Module .....

Functions:

extfov_correction(im, ext_fov, ...)	Apply sinogram correction for extended FOV acquisition mode

pore3d.preprocess.extfov_correction.**extfov_correction**(*im*, *ext_fov*, *ext_fov_rot_right*, *ext_fov_overlap*)

Apply sinogram correction for extended FOV acquisition mode

Parameters

- **im** (*array_like*) – Image data (sinogram) as numpy array.
- **ext** (*bool*) – True if the extended FOV mode has been performed.
- **ext_fov_rot_right** (*bool*) – True if the extended FOV mode has been performed with rotation center

  shifted to the right, left otherwise.

  ext_fov_overlap [int] Number of overlapping pixels.

3.3.5 pore3d.pore3d.preprocess.extract_flatdark

Module .....

Functions:

extract_flatdark(f_in, flat_end, logfilename)	Extract the flat and dark reference images to be used during the pre-processing

pore3d.preprocess.extract_flatdark.**extract_flatdark**(*f_in*, *flat_end*, *logfilename*)

Extract the flat and dark reference images to be used during the pre-processing step.

Parameters

- **f_in** (*HDF5 data structure*) – The data structure containing the flat and dark acquired images.
- **flat_end** (*bool*) – Consider the flat/dark images acquired after the projections (if any).

  logilename [string] Absolute file of a log text file where infos are appended.

3.3.6 pore3d.preprocess.flat_fielding

Module .....

Functions:

flat_fielding(im, i, plan, flat_end, ...)	Process a sinogram with conventional flat fielding plus reference normalization.

pore3d.preprocess.flat_fielding.**flat_fielding**(*im*, *i*, *plan*, *flat_end*, *half_half*, *half_half_line*, *norm_sx*, *norm_dx*)

Process a sinogram with conventional flat fielding plus reference normalization.

Parameters

- **im** (*array_like*) – Image data as numpy array
- **i** (*int*) – Index of the sinogram with reference to the height of a projection

- **plan** (*structure*) – Structure created by the extract_flatdark function (see extract_flatdark.py). This structure contains the flat/dark images acquired before the acquisition of the projections and the flat/dark images acquired after the acquisition of the projections as well as a few flags.
- **flat_end** (*bool*) – True if the process considers the flat/dark images (if any) acquired after the acquisition of the projections.
- **half_half** (*bool*) – True if the process has to be separated by processing the first part of the sinogram with the flat/dark images acquired before the acquisition of the projections and the second part with the flat/dark images acquired after the acquisition of the projections.
- **half_half_line** (*int*) – Usually this value is equal to the height of the projection FOV / 2 but the two parts of the sinogram to process can have a different size.
- **norm_sx** (*int*) – Width in pixels of the left window to be consider for the normalization of the sinogram. This value has to be zero in the case of ROI-CT.
- **norm_dx** (*int*) – Width in pixels of the right window to be consider for the normalization of the sinogram. This value has to be zero in the case of ROI-CT.
- **Example (using h5py, tdf.py, tifffile.py)**
- **————————————————**
- **>>> sino_idx = 512**
- **>>> f = getHDF5('dataset.h5', 'r')**
- **>>> im = tdf.read_sino(f['exchange/data'], sino_idx)**
- **>>> plan = extract_flatdark(f_in, True, False, False, 'tomo', 'dark', 'flat', 'logfile.txt')**
- **>>> im = flat_fielding(im, sino_idx, plan, True, True, 900, 0, 0)**
- **>>> imsave('sino_corr.tif', im)**

3.3.7 pore3d.preprocess.ring_correction

Module .....

Functions:

ring_correction(im, ringrem, flat_end, ...)	Apply ring artifacts compensation by de-striping the input sinogram.

pore3d.preprocess.ring_correction.**ring_correction**(*im*, *ringrem*, *flat_end*, *skip_flat_after*, *half_half*, *half_half_line*, *ext_fov*)

Apply ring artifacts compensation by de-striping the input sinogram.

Parameters

im (*array_like*) – Image data (sinogram) as numpy array.

ringrem [string] String containing ring removal method and parameters

half_half [bool] True to separately process the sinogram in two parts

half_half_line [int] Line number considered to identify the two parts to be processed separately. (This parameter is ignored if half_half is False)

skip_flat_after e ext_fov SERVE???

3.3.8 pore3d.reconstruct.rec_astra

Module .....

Functions:

recon_astra_fbp(im, angles, method, filter_type)	Reconstruct the input sinogram by using the FBP implemented in ASTRA t
recon_astra_iterative(im, angles, method, ...)	Reconstruct the input sinogram by using one of the iterative algorithms imp

pore3d.reconstruct.rec_astra.**recon_astra_fbp**(*im*, *angles*, *method*, *filter_type*)

Reconstruct the input sinogram by using the FBP implemented in ASTRA toolbox.

Parameters

im (*array_like*) – Image data (sinogram) as numpy array.

angles [double] Value in radians representing the number of angles of the sinogram.

method [string] A string with either "FBP" or "FBP_CUDA".

filter_type [string] The available options are "ram-lak", "shepp-logan", "cosine", "hamming", "hann", "tukey", "lanczos", "triangular", "gaussian", "barlett-hann", "blackman", "nuttall", "blackman-harris", "blackman-nuttall", "flat-top", "kaiser", "parzen".

pore3d.reconstruct.rec_astra.**recon_astra_iterative**(*im*, *angles*, *method*, *iterations*, *zerone_mode*)

Reconstruct the input sinogram by using one of the iterative algorithms implemented in ASTRA toolbox.

Parameters

im (*array_like*) – Image data (sinogram) as numpy array.

angles [double] Value in radians representing the number of angles of the sinogram.

method [string] A string with e.g "SIRT" or "SIRT_CUDA" (see ASTRA documentation)

iterations [int] Number of iterations for the algebraic technique

zerone_mode [bool] True if the input sinogram has been rescaled to the [0,1] range (therefore positivity constraints are applied)

3.3.9 pore3d.reconstruct.rec_fista_tv

Module .....

Functions:

recon_fista_tv(im, angles, lam, fista_iter, iter)	Reconstruct the input sinogram by using the FISTA-TV algorithm

pore3d.reconstruct.rec_fista_tv.**recon_fista_tv**(*im*, *angles*, *lam*, *fista_iter*, *iter*)

Reconstruct the input sinogram by using the FISTA-TV algorithm

Parameters

im (*array_like*) – Image data (sinogram) as numpy array.

angles [double] Value in radians representing the number of angles of the input sinogram.

lam [double] Regularization parameter of the FISTA algorithm.

fista_iter [int] Number of iterations of the FISTA algorihtm.

iter [int] Number of iterations of the TV minimization.

3.3.10 pore3d.reconstruct.rec_gridrec

Module .....

Functions:

recon_gridrec(im1, im2, angles, oversampling)	Reconstruct two sinograms (of the same CT scan) with direct Fourier algorithr

pore3d.reconstruct.rec_gridrec.**recon_gridrec**(*im1*, *im2*, *angles*, *oversampling*)

Reconstruct two sinograms (of the same CT scan) with direct Fourier algorithm.

Parameters

- **im1** (*array_like*) – Sinogram image data as numpy array.
- **im2** (*array_like*) – Sinogram image data as numpy array.
- **angles** (*double*) – Value in radians representing the number of angles of the input sinogram.
- **oversampling** (*double*) – Input sinogram is rescaled to increase the sampling of the Fourier space and avoid artifacts. Suggested value in the range [1.2,1.6].

3.3.11 pore3d.reconstruct.rec_mr_fbp

Module .....

Functions:

recon_mr_fbp(im, angles)	Reconstruct a sinogram with the Minimum Residual FBP algorithm (Pelt, 2013).

pore3d.reconstruct.rec_mr_fbp.**recon_mr_fbp**(*im*, *angles*)

Reconstruct a sinogram with the Minimum Residual FBP algorithm (Pelt, 2013).

Parameters

- **im** (*array_like*) – Sinogram image data as numpy array.
- **angles** (*double*) – Value in radians representing the number of angles of the input sinogram.

3.3.12 pore3d.reconstruct.rec_scikit

Module .....

Functions:

recon_scikit_sart(im, angles, iter)	Reconstruct a sinogram with the SART algorithm of the Scikit-Image Python packa
recon_scikit_fbp(im, angles, filter_type)	Reconstruct a sinogram with the FBP algorithm of the Scikit-Image Python package

pore3d.reconstruct.rec_scikit.**recon_scikit_sart**(*im*, *angles*, *iter*)
Reconstruct a sinogram with the SART algorithm of the Scikit-Image Python package

Parameters

- **im** (*array_like*) – Sinogram image data as numpy array.
- **angles** (*double*) – Value in radians representing the number of angles of the input sinogram.
- **iterations** (*int*) – Number of iterations for the SART technique.

pore3d.reconstruct.rec_scikit.**recon_scikit_fbp**(*im*, *angles*, *filter_type*)
Reconstruct a sinogram with the FBP algorithm of the Scikit-Image Python package

Parameters

- **im** (*array_like*) – Sinogram image data as numpy array.
- **angles** (*double*) – Value in radians representing the number of angles of the input sinogram.
- **filter_type** (*string*) – A string with e.g. "ramp", "shepp-logan", "cosine", "hamming", "hann".

3.3.13 pore3d.reconstruct.rec_tomopy

Module .....

Functions:

recon_tomopy_iterative(im, angles, method, ...)	

pore3d.reconstruct.rec_tomopy.**recon_tomopy_iterative**(*im*, *angles*, *method*, *iterations*)

3.3.14 pore3d.utils.caching

Module .....

Functions:

cache2plan(infile, cachepath)	Read from cache the flat/dark images of the input TDF file.
plan2cache(corr_plan, infile, cachepath)	Write to cache the flat/dark images of the input TDF file.

pore3d.utils.caching.**cache2plan**(*infile*, *cachepath*)
Read from cache the flat/dark images of the input TDF file.

Parameters

- **infile** (*string*) – Absolute path of the input TDF dataset.
- **Return value**
- **————**
- **A structure with flat/dark images and related flags.**

pore3d.utils.caching.**plan2cache**(*corr_plan*, *infile*, *cachepath*)

Write to cache the flat/dark images of the input TDF file.

Parameters

- **infile** (*string*) – Absolute path of the input TDF dataset.
- **corr_plan** (*structure*) – The plan with flat/dark images and flags.
- **Return value**
- **————**
- **No return value.**

3.3.15 pore3d.utils.findcenter

Module .....

Functions:

usecorrelation(im1, im2)	Assess the offset (to be used for e.g.

pore3d.utils.findcenter.**usecorrelation**(*im1*, *im2*)

Assess the offset (to be used for e.g. the assessment of the center of rotation or the ovarlap) by computation the peak of the correlation between the two input images.

Parameters

- **im1** (*array_like*) – Image data as numpy array.

  im2 [array_like] Image data as numpy array.
- **Return value**
- **———-**
- **An integer value of the location of the maximum peak correlation.**

3.3.16 pore3d.utils.padding

Module .....

Functions:

upperPowerOfTwo(v)	Return the upper power of two of input value

Table 3.16 – continued from previous page

replicatePadImage(im, marg0, marg1)	Pad the input image by replicating first and last column as well as first and last row the
padImage(im, n_pad0, n_pad1)	Replicate pad the input image to the specified new dimensions.
padSino(im, n_pad)	Pad the input sinogram to the specified width by replicate padding with Hanning smoo

pore3d.utils.padding.**upperPowerOfTwo**(*v*)

Return the upper power of two of input value

Parameters

- **v** (*int*) – A positive integer value
- **Return value**
- **———-**
- **An integer value**

pore3d.utils.padding.**replicatePadImage**(*im*, *marg0*, *marg1*)

Pad the input image by replicating first and last column as well as first and last row the specified number of times.

Parameters

- **im** (*array_like*) – Image data as numpy array.
- **marg0** (*int*) – The number of times first and last row have to be replicated.
- **marg1** (*int*) – The number of times first and last column have to be replicated.
- **Return value**
- **————**
- **A replicated-padded image.**

pore3d.utils.padding.**padImage**(*im*, *n_pad0*, *n_pad1*)

Replicate pad the input image to the specified new dimensions.

Parameters

- **im** (*array_like*) – Image data as numpy array
- **n_pad0** (*int*) – The new height of the image
- **n_pad1** (*int*) – The new width of the image
- **Return value**
- **———-**
- **A padded image**

pore3d.utils.padding.**padSino**(*im*, *n_pad*)

Pad the input sinogram to the specified width by replicate padding with Hanning smoothing to zero.

Parameters

- **im** (*array_like*) – Image data as numpy array.
- **n_pad** (*int*) – The new width of the sinogram.
- **Return value**
- **———-**
- **A padded image**

3.4 Examples

Here we describe what the examples are doing. You can cite with *[B1]*.

3.4.1 Example 02

This section contains the example_01 script.

Download file: `example_01.py`

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

"""
Example script
"""

from __future__ import print_function
import project

if __name__ == '__main__':

    parameter_01 = 1
    parameter_02 = 2
    parameter_03 = 3

    # call function_01
    result = project.function_01(parameter_01, parameter_02, parameter_03)
    print (result)
```

3.4.2 Example 02

This section contains the example_02 script.

Download file: `example_02.py`

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

"""
Example script
"""

from __future__ import print_function
import project

if __name__ == '__main__':

    parameter_01 = 1
    parameter_02 = 2
    parameter_03 = 3

    # call function_01
    result = project.function_02(parameter_01, parameter_02, parameter_03)
    print (result)
```

3.5 Credits

3.5.1 Citations

We kindly request that you cite the following article *[A1]* if you use project.

3.5.2 References

Bibliography

[A1] De Carlo F, Gursoy D, Marone F, Rivers M, Parkinson YD, Khan F, Schwarz N, Vine DJ, Vogt S, Gleber SC, Narayanan S, Newville M, Lanzirotti T, Sun Y, Hong YP, and Jacobsen C. Scientific data exchange: a schema for hdf5-based storage of raw and analyzed data. *Journal of Synchrotron Radiation*, 21(6):1224–1230, 2014.

[B1] De Carlo F, Gursoy D, Marone F, Rivers M, Parkinson YD, Khan F, Schwarz N, Vine DJ, Vogt S, Gleber SC, Narayanan S, Newville M, Lanzirotti T, Sun Y, Hong YP, and Jacobsen C. Scientific data exchange: a schema for hdf5-based storage of raw and analyzed data. *Journal of Synchrotron Radiation*, 21(6):1224–1230, 2014.

p

C

E

F

G

P

R

U

W