DXchange Documentation

Release 0.1.7

Argonne National Laboratory

Aug 04, 2023

Contents

DXchange provides an interface with TomoPy [Gursoy:14b] and raw tomographic data collected at different synchrotron facilities including the Data Exchange file format (DXfile) [A1], currently in use at the Advanced Photon Source beamline 2-BM and 32-ID, at the Swiss Light Source Tomcat beamline and at the Elettra SYRMEP beamline [Elettra:01].

Warning: DXchange will drop support for Python 2 before 1 January 2020. For more information, visit https://python3statement.org/.

CHAPTER 1

Features

- Scientific Data Exchange file format.
- Readers for tomographic data files collected at different facilities.
- Writers for different file formats.

CHAPTER 2

Highlights

- Based on Hierarchical Data Format 5 (HDF5).
- Focuses on technique rather than instrument descriptions.
- Provenance tracking for understanding analysis steps and results.
- Ease of readability.

CHAPTER 3

Contribute

- Documentation: https://github.com/data-exchange/dxchange/tree/master/doc
- Issue Tracker: https://github.com/data-exchange/dxchange/issues
- Source Code: https://github.com/data-exchange/dxchange

3.1 Install

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

Contents:

3.1.1 Installing from source

Clone the DXchange from GitHub repository:

```
git clone https://github.com/data-exchange/dxchange.git dxchange
```

then:

```
cd dxchange
python setup.py install
```

DXchange is dependent on other libraries, listed in the requirements.txt and meta.yaml files.

3.1.2 Installing from Conda/Binstar

First you must have Conda installed, then open a terminal or a command prompt window and run:

```
conda install -c conda-forge dxchange
```

3.1.3 Updating the installation

Data Management is an active project, so we suggest you update your installation frequently. To update the installation run in your terminal:

```
conda update -c conda-forge dxchange
```

For some more information about using Conda, please refer to the docs.

3.2 API reference

dxchange Modules:

3.2.1 dxchange.exchange

3.2.2 dxchange.reader

3.2.3 dxchange.writer

3.3 Examples

Below are examples for reading tomographic data sets from different facilities and process them with TomoPy [Gursoy:14b].

3.3.1 Anka TopoTomo

This section contains a script to read the Anka TopoTomo tomography dataset and reconstruct it with tomoPy.

Download file: `rec_anka.py`

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

"""
TomoPy example script to reconstruct the Anka topo-tomo tomography data as
original tiff.
"""

from __future__ import print_function
import tomopy
import dxchange

if __name__ == '__main__':
    # Set path to the micro-CT data to reconstruct.
```

(continues on next page)

(continued from previous page)

```
    fname = 'data_dir/'

    proj_start = 0
    proj_end = 1800
    flat_start = 0
    flat_end = 100
    dark_start = 0
    dark_end = 100

    ind_tomo = range(proj_start, proj_end)
    ind_flat = range(flat_start, flat_end)
    ind_dark = range(dark_start, dark_end)

    # Select the sinogram range to reconstruct.
    start = 0
    end = 16

    # Read the Anka tiff raw data.
    proj, flat, dark = dxchange.read_anka_topotomo(fname, ind_tomo, ind_flat,
                                                   ind_dark, sino=(start, end))

    # Set data collection angles as equally spaced between 0-180 degrees.
    theta = tomopy.angles(proj.shape[0])

    # Flat-field correction of raw data.
    proj = tomopy.normalize(proj, flat, dark)

    # Find rotation center.
    rot_center = tomopy.find_center(proj, theta, init=1024,
                                    ind=0, tol=0.5)
    print("Center of rotation: ", rot_center)

    proj = tomopy.minus_log(proj)

    # Reconstruct object using Gridrec algorithm.
    rec = tomopy.recon(proj, theta, center=rot_center, algorithm='gridrec')

    # Mask each reconstructed slice with a circle.
    rec = tomopy.circ_mask(rec, axis=0, ratio=0.95)

    # Write data as stack of TIFs.
    dxchange.write_tiff_stack(rec, fname='recon_dir/recon')
```

3.3.2 Australian Synchrotron

This section contains a script to read the Australian Synchrotron Facility tomography dataset and reconstruct it with tomoPy.

Download file: `rec_australian.py`

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

"""
TomoPy example script to reconstruct the Australian Synchrotron Facility
```

(continues on next page)

(continued from previous page)

```
data as original tiff.
"""

from __future__ import print_function
import tomopy
import dxchange

if __name__ == '__main__':

    # Set path to the micro-CT data to reconstruct.
    fname = 'data_dir/'

    proj_start = 0
    proj_end = 1801
    flat_start = 0
    flat_end = 10
    dark_start = 0
    dark_end = 10

    ind_tomo = range(proj_start, proj_end)
    ind_flat = range(flat_start, flat_end)
    ind_dark = range(dark_start, dark_end)

    # Select the sinogram range to reconstruct.
    start = 290
    end = 294

    # Read the Australian Synchrotron Facility data
    proj, flat, dark = dxchange.read_aus_microct(fname, ind_tomo, ind_flat, ind_dark,␣
↪sino=(start, end))

    # Set data collection angles as equally spaced between 0-180 degrees.
    theta = tomopy.angles(proj.shape[0])

    # Flat-field correction of raw data.
    proj = tomopy.normalize(proj, flat, dark)

    # Find rotation center.
    rot_center = tomopy.find_center(proj, theta, init=1024, ind=0, tol=0.5)
    print("Center of rotation: ", rot_center)

    proj = tomopy.minus_log(proj)

    # Reconstruct object using Gridrec algorithm.
    rec = tomopy.recon(proj, theta, center=rot_center, algorithm='gridrec')

    # Mask each reconstructed slice with a circle.
    rec = tomopy.circ_mask(rec, axis=0, ratio=0.95)

    # Write data as stack of TIFs.
    dxchange.write_tiff_stack(rec, fname='recon_dir/aus_')
```

3.3.3 ALS 8.3.2

This section contains a script to read the als 8.3.2 tomography dataset and reconstruct it with tomoPy.

Download file: `rec_als.py` and `rec_als_hdf5.py`

3.3.4 Elettra Syrmep

This section contains a script to read the Elettra syrmep tomography dataset and reconstruct it with tomoPy.

Download file: `rec_elettra.py`

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

"""
TomoPy example script to reconstruct the Elettra syrmep data as original tiff.
"""

from __future__ import print_function
import tomopy
import dxchange

if __name__ == '__main__':

    # Set path to the CT data to reconstruct.
    fname = 'data_dir/'

    proj_start = 1
    proj_end = 1801
    flat_start = 1
    flat_end = 11
    dark_start = 1
    dark_end = 11

    ind_tomo = range(proj_start, proj_end)
    ind_flat = range(flat_start, flat_end)
    ind_dark = range(dark_start, dark_end)

    # Select the sinogram range to reconstruct.
    start = 0
    end = 16

    # Read the Elettra syrmep
    proj, flat, dark = dxchange.read_elettra_syrmep(fname, ind_tomo, ind_flat, ind_
↪dark, sino=(start, end))

    # Set data collection angles as equally spaced between 0-180 degrees.
    theta = tomopy.angles(proj.shape[0], 0, 180)

    # Flat-field correction of raw data.
    proj = tomopy.normalize(proj, flat, dark)

    # Find rotation center.
    rot_center = tomopy.find_center(proj, theta, init=1024, ind=0, tol=0.5)
    print("Center of rotation: ", rot_center)

    proj = tomopy.minus_log(proj)

    # Reconstruct object using Gridrec algorithm.
    rec = tomopy.recon(proj, theta, center=rot_center, algorithm='gridrec')
```

(continues on next page)

(continued from previous page)

```
    # Mask each reconstructed slice with a circle.
    rec = tomopy.circ_mask(rec, axis=0, ratio=0.95)

    # Write data as stack of TIFs.
    dxchange.write_tiff_stack(rec, fname='recon_dir/recon')
```

3.3.5 ESRF ID-19

This section contains a script to read the ESRF ID-19 tomography dataset and reconstruct it with tomoPy.

Download file: `rec_esrf.py`

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

"""
TomoPy example script to reconstruct the ESRF tomography data as original edf
files.
"""

from __future__ import print_function
import tomopy
import dxchange

if __name__ == '__main__':
    # Set path to the micro-CT data to reconstruct.
    fname = 'data_dir/'

    # Select the sinogram range to reconstruct.
    start = 0
    end = 16

    # Read the ESRF ID-19 raw data.
    proj, flat, dark = dxchange.read_esrf_id19(fname, sino=(start, end))

    # Set data collection angles as equally spaced between 0-180 degrees.
    theta = tomopy.angles(proj.shape[0])

    # Flat-field correction of raw data.
    proj = tomopy.normalize(proj, flat, dark)

    # Find rotation center.
    rot_center = tomopy.find_center(proj, theta, init=1024,
                                    ind=0, tol=0.5)
    print("Center of rotation: ", rot_center)

    proj = tomopy.minus_log(proj)

    # Reconstruct object using Gridrec algorithm.
    rec = tomopy.recon(proj, theta, center=rot_center, algorithm='gridrec')

    # Mask each reconstructed slice with a circle.
    rec = tomopy.circ_mask(rec, axis=0, ratio=0.95)

```

(continues on next page)

(continued from previous page)

```
    # Write data as stack of TIFs.
    dxchange.write_tiff_stack(rec, fname='recon_dir/recon')
```

3.3.6 APS 1-ID

This section contains a script to read the APS 1-ID tomography dataset and reconstruct it with tomoPy.

Download file: `rec_aps_1id.py`

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

"""
TomoPy example script to reconstruct the APS 1-ID tomography data as original tiff.
"""

from __future__ import print_function
import tomopy
import dxchange

if __name__ == '__main__':

    # Set path to the micro-CT data to reconstruct.
    fname = 'data_dir/sample_name_prefix'

    # Select the sinogram range to reconstruct.
    start = 0
    end = 16

    # Read the APS 1-ID raw data.
    proj, flat, dark = dxchange.read_aps_1id(fname, sino=(start, end))

    # Set data collection angles as equally spaced between 0-180 degrees.
    theta = tomopy.angles(proj.shape[0])

    # Flat-field correction of raw data.
    proj = tomopy.normalize(proj, flat, dark)

    # Find rotation center.
    rot_center = tomopy.find_center(proj, theta, init=1024, ind=0, tol=0.5)
    print("Center of rotation: ", rot_center)

    proj = tomopy.minus_log(proj)

    # Reconstruct object using Gridrec algorithm.
    rec = tomopy.recon(proj, theta, center=rot_center, algorithm='gridrec')

    # Mask each reconstructed slice with a circle.
    rec = tomopy.circ_mask(rec, axis=0, ratio=0.95)

    # Write data as stack of TIFs.
    dxchange.write_tiff_stack(rec, fname='recon_dir/recon')
```

3.3.7 APS 5-BM

This section contains a script to read the APS 5-BM tomography dataset and reconstruct it with tomoPy.

Download file: `rec_aps_5bm.py`

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

"""
TomoPy example script to reconstruct the APS 5-BM data as original xmt.
xmt are 16 bit unsigned integer tiff file that requires a byte swap before
being processed.
"""

from __future__ import print_function
import tomopy
import dxchange

if __name__ == '__main__':

    # Set path to the micro-CT data to reconstruct.
    fname = 'data_dir/'

    # Select the sinogram range to reconstruct.
    start = 290
    end = 294

    # Read the APS 5-BM raw data
    proj, flat, dark = dxchange.read_aps_5bm(fname, sino=(start, end))

    # Set data collection angles as equally spaced between 0-180 degrees.
    theta = tomopy.angles(proj.shape[0])

    # Flat-field correction of raw data.
    proj = tomopy.normalize(proj, flat, dark)

    # remove stripes
    proj = tomopy.remove_stripe_fw(proj,level=7,wname='sym16',sigma=1,pad=True)

    # Set rotation center.
    rot_center = proj.shape[2] / 2.0
    print("Center of rotation: ", rot_center)

    proj = tomopy.minus_log(proj)

    # Reconstruct object using Gridrec algorithm.
    rec = tomopy.recon(proj, theta, center=rot_center, algorithm='gridrec')

    # Mask each reconstructed slice with a circle.
    rec = tomopy.circ_mask(rec, axis=0, ratio=0.95)

    # Write data as stack of TIFs.
    dxchange.write_tiff_stack(rec, fname='recon_dir/recon')
```

3.3.8 APS 8-BM

This section contains a script to read the X-radia XRM tomography dataset and reconstruct it with tomoPy.

Download file: `rec_aps_8bm.py`

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

"""
TomoPy example script to reconstruct the xrm tomography data from
the original stack of xrm. To use rename the xrm data as
radios/image_00000.xrm and flats/ref_00000.xrm
"""

from __future__ import print_function
import tomopy
import dxchange

if __name__ == '__main__':
    # Set path to the micro-CT data to reconstruct.
    fname = 'data_dir/'

    proj_start = 0
    proj_end = 1800
    flat_start = 0
    flat_end = 100

    ind_tomo = range(proj_start, proj_end)
    ind_flat = range(flat_start, flat_end)

    # Select the sinogram range to reconstruct.
    start = 0
    end = 16

    # Read the APS 8-BM raw data.
    proj, flat, metadata = dxchange.read_aps_8bm(fname, ind_tomo, ind_flat,
                                                 sino=(start, end))

    # make the darks
    dark = np.zeros((1, proj.shape[1], proj.shape[2]))

    # Set data collection angles as equally spaced between 0-180 degrees.
    theta = tomopy.angles(proj.shape[0])

    # Flat-field correction of raw data.
    proj = tomopy.normalize(proj, flat, dark)

    # Find rotation center.
    rot_center = tomopy.find_center(proj, theta, init=1024,
                                    ind=0, tol=0.5)
    print("Center of rotation: ", rot_center)

    proj = tomopy.minus_log(proj)

    # Reconstruct object using Gridrec algorithm.
    rec = tomopy.recon(proj, theta, center=rot_center, algorithm='gridrec')

```

(continues on next page)

(continued from previous page)

```
    # Mask each reconstructed slice with a circle.
    rec = tomopy.circ_mask(rec, axis=0, ratio=0.95)

    # Write data as stack of TIFs.
    dxchange.write_tiff_stack(rec, fname='recon_dir/recon')
```

3.3.9 APS 13-BM

This section contains a script to read the APS 13-BM tomography dataset and reconstruct it with tomoPy.

Download file: `rec_aps_13bm.py`

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

"""
TomoPy example script to reconstruct the APS 13-BM tomography
data as original netcdf files. To use, change fname to just
the file name (e.g. 'sample[2].nc' would be 'sample'.
Reconstructed dataset will be saved as float32 netcdf3.
"""
import glob
import numpy as np
import tomopy as tp
import dxchange as dx

from netCDF4 import Dataset

if __name__ == '__main__':
    ## Set path (without file suffix) to the micro-CT data to reconstruct.
    fname = 'data_dir/sample'

    ## Import Data.
    proj, flat, dark, theta = dx.exchange.read_aps_13bm(fname, format = 'netcdf4')

    ## Flat-field correction of raw data.
    proj = tp.normalize(proj, flat = flat, dark = dark)

    ## Additional flat-field correction of raw data to negate need to mask.
    proj = tp.normalize_bg(proj, air = 10)

    ## Set rotation center.
    rot_center = tp.find_center_vo(proj)
    print('Center of rotation: ', rot_center)

    tp.minus_log(proj, out = proj)

    # Reconstruct object using Gridrec algorith.
    rec = tp.recon(proj, theta, center = rot_center, sinogram_order = False,␣
↪algorithm = 'gridrec', filter_name = 'hann')
    rec = tp.remove_nan(rec)

    ## Writing data in netCDF3 .volume.
    ncfile = Dataset('filename.volume', 'w', format = 'NETCDF3_64BIT', clobber = True)
    NX = ncfile.createDimension('NX', rec.shape[2])
```

(continues on next page)

(continued from previous page)

```
    NY = ncfile.createDimension('NY', rec.shape[1])
    NZ = ncfile.createDimension('NZ', rec.shape[0])
    volume = ncfile.createVariable('VOLUME', 'f4', ('NZ','NY','NX'))
    volume[:] = rec
    ncfile.close()
```

3.3.10 APS 26-ID

This section contains a script to read the X-radia XRM tomography dataset and reconstruct it with tomoPy.

Download file: `rec_aps_26id.py`

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

"""
TomoPy example script to reconstruct the xrm tomography data from
the original stack of xrm. To use rename the xrm data as
radios/image_00000.xrm and flats/ref_00000.xrm
"""

from __future__ import print_function
import tomopy
import dxchange

if __name__ == '__main__':
    # Set path to the micro-CT data to reconstruct.
    fname = 'data_dir/'

    proj_start = 0
    proj_end = 1800
    flat_start = 0
    flat_end = 100

    ind_tomo = range(proj_start, proj_end)
    ind_flat = range(flat_start, flat_end)

    # Select the sinogram range to reconstruct.
    start = 0
    end = 16

    # Read the APS 26-ID raw data.
    proj, flat, metadata = dxchange.read_aps_26id(fname, ind_tomo, ind_flat,
                                                  sino=(start, end))

    # make the darks
    dark = np.zeros((1, proj.shape[1], proj.shape[2]))

    # Set data collection angles as equally spaced between 0-180 degrees.
    theta = tomopy.angles(proj.shape[0])

    # Flat-field correction of raw data.
    proj = tomopy.normalize(proj, flat, dark)

    # Find rotation center.
```

(continues on next page)

(continued from previous page)

```
    rot_center = tomopy.find_center(proj, theta, init=1024,
                                    ind=0, tol=0.5)
    print("Center of rotation: ", rot_center)

    proj = tomopy.minus_log(proj)

    # Reconstruct object using Gridrec algorithm.
    rec = tomopy.recon(proj, theta, center=rot_center, algorithm='gridrec')

    # Mask each reconstructed slice with a circle.
    rec = tomopy.circ_mask(rec, axis=0, ratio=0.95)

    # Write data as stack of TIFs.
    dxchange.write_tiff_stack(rec, fname='recon_dir/recon')
```

3.3.11 APS 2-BM & 32-ID

This section contains a script to read the APS 2-BM and 32-ID tomography dataset and reconstruct it with tomoPy.

Download file: `rec_aps_32id_full.py`

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

"""
TomoPy example script to reconstruct TXM data set.
"""

from __future__ import print_function
import tomopy
import dxchange

if __name__ == '__main__':

    # Set path to the micro-CT data to reconstruct.
    fname = 'data_dir/sample.h5'

    # Select sinogram range to reconstruct.
    start = 0
    end = 16

    # Read APS 32-ID raw data.
    proj, flat, dark, theta = dxchange.read_aps_32id(fname, sino=(start, end))

    # If data collection angles is not defined in the hdf file then set it as equally␣
↪spaced between 0-180 degrees.
    if (theta is None):
        theta = tomopy.angles(proj.shape[0])
    else:
        pass

    # Flat-field correction of raw data.
    proj = tomopy.normalize(proj, flat, dark)

    # Find rotation center.
```

(continues on next page)

(continued from previous page)

```
    rot_center = tomopy.find_center(proj, theta, ind=0, init=1024, tol=0.5)
    print("Center of rotation: ", rot_center)

    proj = tomopy.minus_log(proj)

    # Reconstruct object using Gridrec algorithm.
    rec = tomopy.recon(proj, theta, center=rot_center, algorithm='gridrec')

    # Mask each reconstructed slice with a circle.
    rec = tomopy.circ_mask(rec, axis=0, ratio=0.95)

    # Write data as stack of TIFs.
    dxchange.write_tiff_stack(rec, fname='recon_dir/recon')
```

3.3.12 Petra III P05

This section contains a script to read the Petra III P05 tomography dataset and reconstruct it with tomoPy.

Download file: `rec_petraIII.py`

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

"""
TomoPy example script to reconstruct the PetraIII P05 tomography data as original␣
↪tiff.
"""

from __future__ import print_function
import tomopy
import dxchange

if __name__ == '__main__':

    # Set path to the micro-CT data to reconstruct.
    fname = '/data_dir/sample_name00_0000/'

    proj_start = 0
    proj_end = 1441
    flat_start = 0
    flat_end = 20
    dark_start = 0
    dark_end = 20

    ind_tomo = range(proj_start, proj_end)
    ind_flat = range(flat_start, flat_end)
    ind_dark = range(dark_start, dark_end)

    # Select the sinogram range to reconstruct.
    start = 0
    end = 16

    # Read the Petra III P05
    proj, flat, dark = dxchange.read_petraIII_p05(fname, ind_tomo, ind_flat, ind_dark,
↪ sino=(start, end))
```

(continues on next page)

(continued from previous page)

```

    # Set data collection angles as equally spaced between 0-180 degrees.
    theta = tomopy.angles(proj.shape[0])

    # Flat-field correction of raw data.
    proj = tomopy.normalize(proj, flat, dark)

    # Find rotation center.
    rot_center = tomopy.find_center(proj, theta, init=1024, ind=0, tol=0.5)
    print("Center of rotation: ", rot_center)

    proj = tomopy.minus_log(proj)

    # Reconstruct object using Gridrec algorithm.
    rec = tomopy.recon(proj, theta, center=rot_center, algorithm='gridrec')

    # Mask each reconstructed slice with a circle.
    rec = tomopy.circ_mask(rec, axis=0, ratio=0.95)

    # Write data as stack of TIFs.
    dxchange.write_tiff_stack(rec, fname='recon_dir/petra_')
```

3.3.13 SLS Tomcat

This section contains a script to read the Swiss Light Source tomcat tomography dataset and reconstruct it with tomoPy.

Download file: `rec_tomcat.py`

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

"""
TomoPy example script to reconstruct the Swiss Light Source TOMCAT tomography
data as original tiff.
"""

from __future__ import print_function
import tomopy
import dxchange

if __name__ == '__main__':
    # Set path to the micro-CT data to reconstruct.
    fname = 'data_dir/sample_name_prefix'

    # Select the sinogram range to reconstruct.
    start = 0
    end = 16

    # Read the APS 1-ID raw data.
    proj, flat, dark = dxchange.read_sls_tomcat(fname, sino=(start, end))

    # Set data collection angles as equally spaced between 0-180 degrees.
    theta = tomopy.angles(proj.shape[0], 0, 180)

    # Flat-field correction of raw data.
```

(continues on next page)

(continued from previous page)

```
    proj = tomopy.normalize(proj, flat, dark)

    # Find rotation center.
    rot_center = tomopy.find_center(proj, theta, init=1024,
                                    ind=0, tol=0.5)
    print("Center of rotation:", rot_center)

    proj = tomopy.minus_log(proj)

    # Reconstruct object using Gridrec algorithm.
    rec = tomopy.recon(proj, theta, center=rot_center, algorithm='gridrec')

    # Mask each reconstructed slice with a circle.
    rec = tomopy.circ_mask(rec, axis=0, ratio=0.95)

    # Write data as stack of TIFs.
    dxchange.write_tiff_stack(rec, fname='recon_dir/recon')
```

3.3.14 X-radia XRM

This section contains a script to read the X-radia XRM tomography dataset and reconstruct it with tomoPy.

Download file: `rec_xradia_xrm.py`

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

"""
TomoPy example script to reconstruct the xrm tomography data from
the original stack of xrm. To use rename the xrm data as
radios/image_00000.xrm and flats/ref_00000.xrm
"""

from __future__ import print_function
import tomopy
import dxchange

if __name__ == '__main__':
    # Set path to the micro-CT data to reconstruct.
    fname = 'data_dir/'

    proj_start = 0
    proj_end = 1800
    flat_start = 0
    flat_end = 100

    ind_tomo = range(proj_start, proj_end)
    ind_flat = range(flat_start, flat_end)

    # Select the sinogram range to reconstruct.
    start = 0
    end = 16

    # APS 26-ID has an x-radia system collecting raw data as xrm.
    proj, flat, metadata = dxchange.read_aps_26id(fname, ind_tomo, ind_flat,
```

(continues on next page)

(continued from previous page)

```
                                                sino=(start, end))

    # make the darks
    dark = np.zeros((1, proj.shape[1], proj.shape[2]))

    # Set data collection angles as equally spaced between 0-180 degrees.
    theta = tomopy.angles(proj.shape[0])

    # Flat-field correction of raw data.
    proj = tomopy.normalize(proj, flat, dark)

    # Find rotation center.
    rot_center = tomopy.find_center(proj, theta, init=1024,
                                    ind=0, tol=0.5)
    print("Center of rotation: ", rot_center)

    proj = tomopy.minus_log(proj)

    # Reconstruct object using Gridrec algorithm.
    rec = tomopy.recon(proj, theta, center=rot_center, algorithm='gridrec')

    # Mask each reconstructed slice with a circle.
    rec = tomopy.circ_mask(rec, axis=0, ratio=0.95)

    # Write data as stack of TIFs.
    dxchange.write_tiff_stack(rec, fname='recon_dir/recon')
```

For a repository of experimental and simulated data sets please check TomoBank [DeCarlo:17].

3.4 Credits

3.4.1 Citations

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

Code from algotom [A2] was used in the reader read_hdf_meta to generate an hdf file tree

3.4.2 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.

[A2] Nghia T. Vo, Robert C. Atwood, Michael Drakopoulos, and Thomas Connolley. Data processing methods and data acquisition for samples larger than the field of view in parallel-beam tomography. *Opt. Express*, 29(12):17849–17874, Jun 2021. URL: http://www.opticsexpress.org/abstract.cfm?URI=oe-29-12-17849, doi:10.1364/OE.418448.