tomopy-cli Documentation

Release 0.3

Argonne National Laboratory

Feb 07, 2023

CONTENTS

1 Features **3**

2 Contribute **5**

3 Content **7**

3.1 Install 7
3.2 Tutorials and How-To Guides 8
3.3 Usage 15
3.4 Testing 18
3.5 API reference 18
3.6 Credits 20

Bibliography **23**

Python Module Index **25**

Index **27**

tomopy-cli is commad-line-interface for TomoPy an open-source Python package for tomographic data processing and image reconstruction.

CHAPTER

ONE

FEATURES

- List here
- the module features

CHAPTER

TWO

CONTRIBUTE

- Documentation: https://github.com/tomography/tomopy-cli/tree/master/doc
- Issue Tracker: https://github.com/tomography/tomopy-cli/docs/issues
- Source Code: https://github.com/tomography/tomopy-cli/

CHAPTER

THREE

CONTENT

3.1 Install

3.1.1 Installation from Source

```
$ git clone https://github.com/tomography/tomopy-cli.git
$ cd tomopy-cli
$ python setup.py install
```

in a prepared virtualenv or as root for system-wide installation.

Warning: If your python installation is in a location different from #!/usr/bin/env python please edit the first line of the bin/tomopy file to match yours.

After the installation you will be prompted to:

```
$ source /Users/userid/complete_tomopy.sh
```

This enable the autocompletition of all tomopy recon options. Just press tab after:

```
$ tomopy recon --<TAB> <TAB>
```

to select an optional parameter and show its default value.

Warning: in some systems you are required to set *complete_tomopy.sh* as executable with:

```
$ chmod +x /Users/userid/complete_tomopy.sh
```

Windows

To installat tomopy-cli under Windows watch this video.

3.1.2 Update

tomopy-cli is constantly updated to include new features. To update your locally installed version:

```
$ cd tomopy-cli
$ git pull
$ python setup.py install
```

3.1.3 Dependencies

Install tomopy

```
$ conda install -c conda-forge tomopy
```

install meta

```
$ git clone https://github.com/xray-imaging/meta.git
$ cd meta
$ python setup install meta
```

Optionally, *dxchange* may be needed for file I/O:

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

3.2 Tutorials and How-To Guides

3.2.1 Tutorial 1: Basic Reconstructions

This tutorial will guide you through the first simple tomography reconstruction. By the end, you will be able to:

- Load a set of projection images
- Produce a single reconstructed slice
- Produce the full reconstructed volume
- Supply command line arguments to alter the reconstruction behavior
- Create a configuration file to stream-line reconstructions

Note: This tutorial assumes you have a working python installation (i.e. Anaconda) and have *installed tomopy-cli* and it's *dependencies* (both tomopy and dxchange).

Warning: This tutorial has only been tested on **GNU/Linux** operating systems. Windows and MacOS support is an ongoing project.

Preamble: Prepare Test Images

Note: This tutorial will use some test data known as the *Shepp-Logan phantom*, which will need to be saved in a format that mimics synchrotron tomography data. This preamble section is not needed when performing reconstructions on real experimental data.

First, open a terminal and create an empty directory to hold our data:

```
$ mkdir tomopy_cli_tutorial
$ cd tomopy_cli_tutorial
```

We will also need a way to view the results of our reconstruction, so we'll install matplotlib:

```
$ pip install matplotlib
```

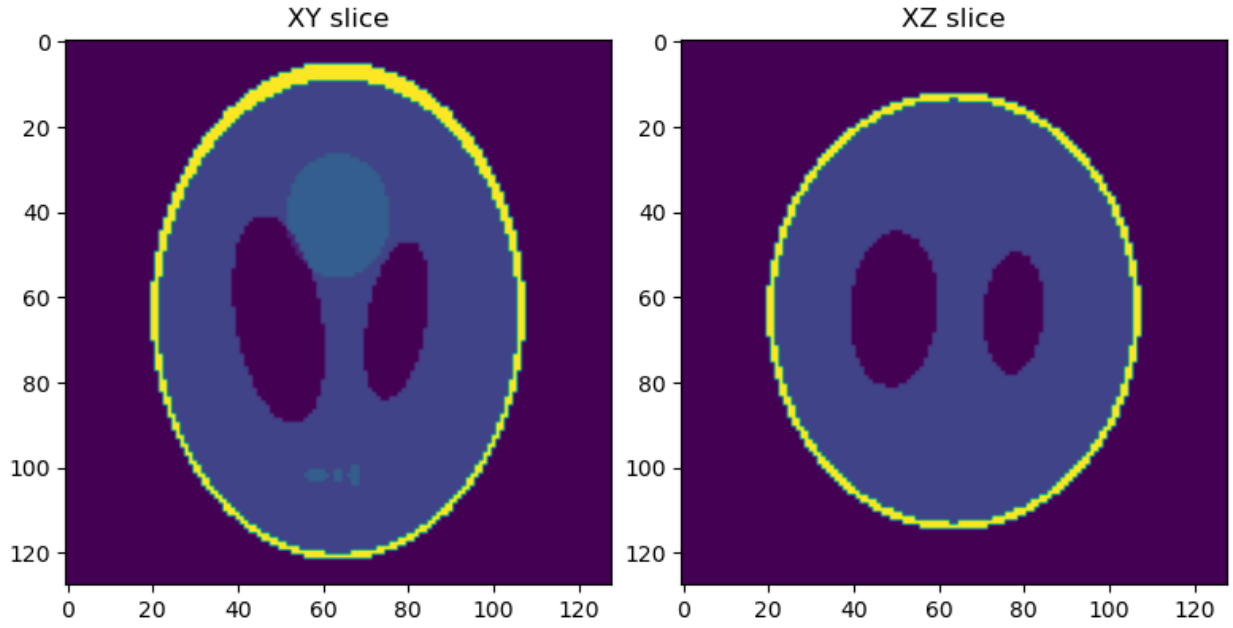
Now **open a python console** (`$ python`) and use the tomopy package to prepare phantom test data:

```
>>> import tomopy, dxchange, numpy, h5py, matplotlib.pyplot as plt
>>> phantom = tomopy.misc.phantom.shepp3d(128)
```

And visualize the results:

```
>>> fig, (axL, axR) = plt.subplots(1, 2)
>>> axL.imshow(phantom[64])
>>> axL.set_title("XY slice")
>>> axR.imshow(phantom[:,64])
>>> axR.set_title("XZ slice")
>>> plt.show(block=False)
```

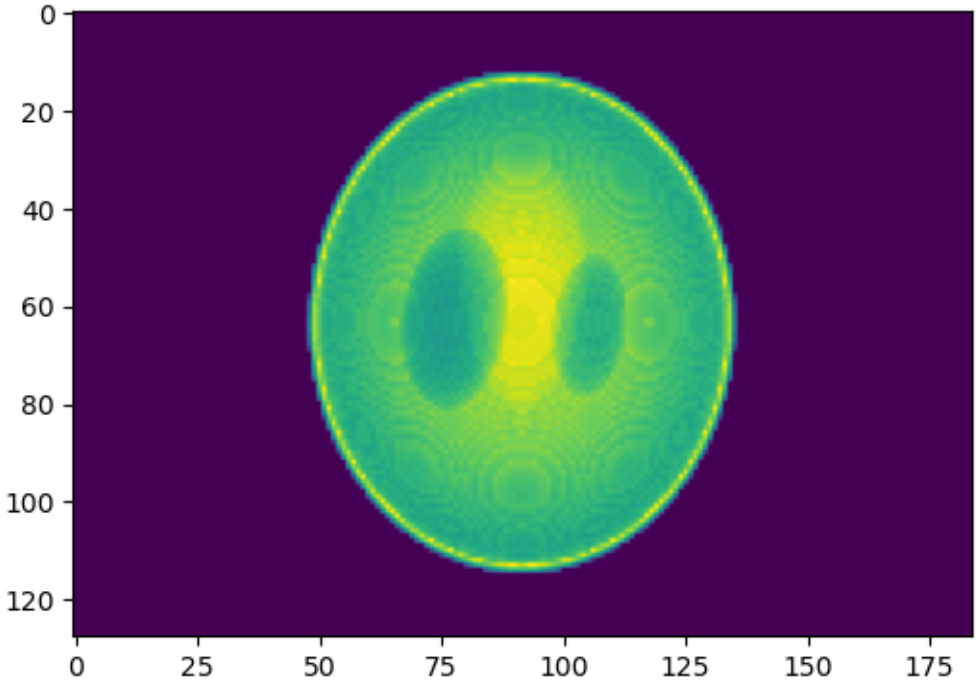
You should now see a plot with a horizontal (XY) slice and a vertical (XZ) slice of the test data:



Next, we will simulate a set of projection images from this phantom volume.

```
>>> angles = tomopy.sim.project.angles(181)
>>> proj = tomopy.sim.project.project(phantom, theta=angles)
>>> plt.figure()
>>> plt.imshow(proj[0])
>>> plt.show(block=False)
```

You should now see the first image in a series of simulated projections of the phantom volume.



This set of projections will be used as input for reconstructions in the rest of this tutorial, so we will save them and then leave the python console for the next segment of this tutorial:

Warning: This following commands will overwrite any existing file named *phantom_projections.h5*.

```
>>> file = h5py.File("phantom_projections.h5", mode="w")
>>> file.create_dataset("exchange/data", data=numpy.exp(-proj))
>>> file.create_dataset("exchange/data_white", data=numpy.ones(shape=(1, *proj.shape[-
↪2:])))
>>> file.create_dataset("exchange/data_dark", data=numpy.zeros(shape=(1, *proj.shape[-
↪2:])))
>>> file.close()
>>> exit()
```

Perform a Single Slice Reconstruction

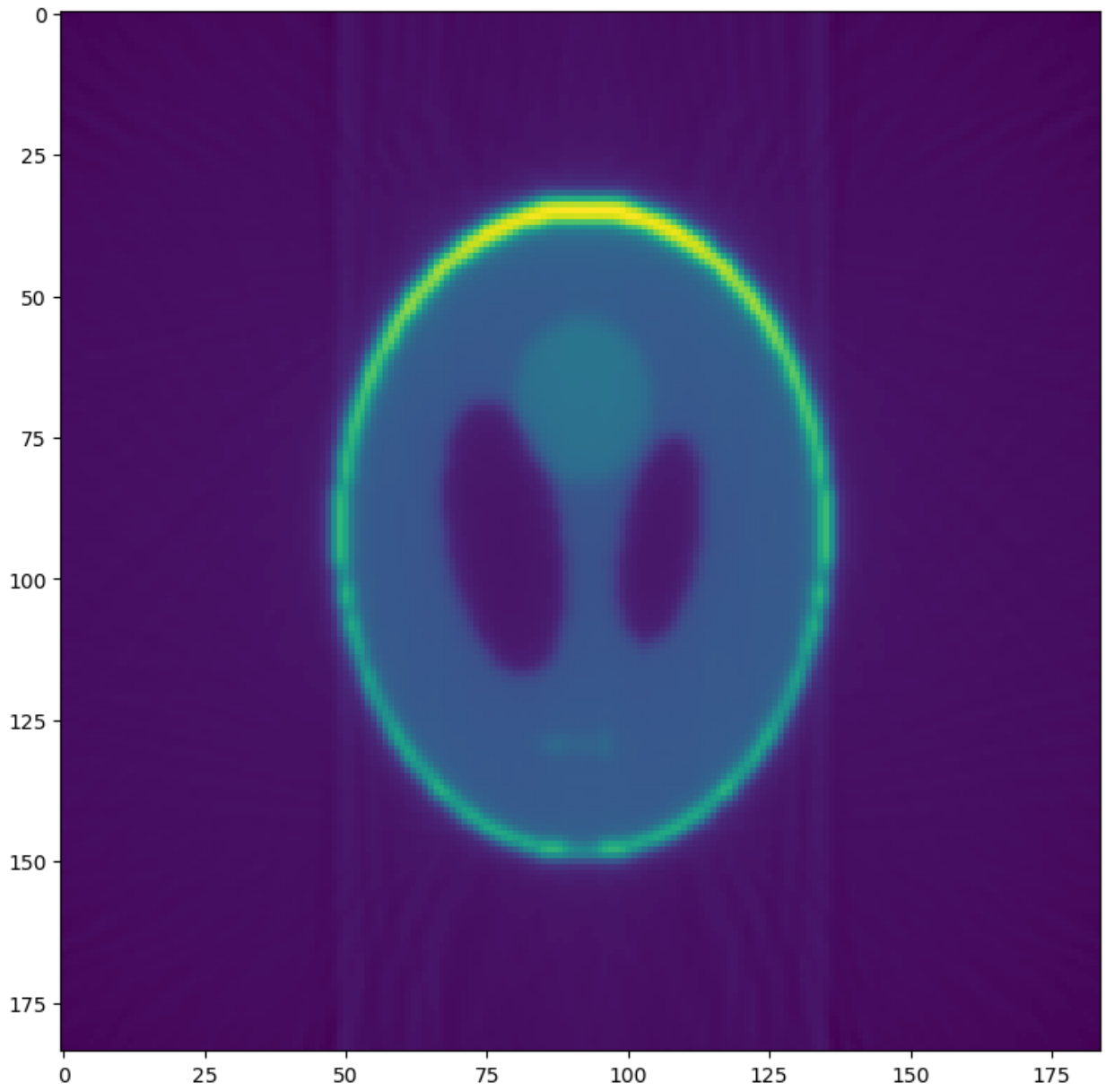
Now that we have some projection data work with, we will perform a simple single-slice reconstruction:

```
$ tomopy recon --file-name phantom_projections.h5 --reconstruction-type=slice --output-
↪folder=_rec
```

This will save reconstructions as TIFF files in the *_rec* folder. Single slice reconstructions are stored in *_rec/slice_rec*.

To visualize the results, **open a python console** again and use matplotlib to view the reconstructed data.

```
>>> import matplotlib.pyplot as plt, imageio
>>> slc = imageio.read("_rec/slice_rec/recon_phantom_projections.tiff").get_data(0)
>>> plt.figure()
>>> plt.imshow(slc)
>>> plt.show(block=False)
```



Compare this reconstructed slice to the original phantom produced in the first section.

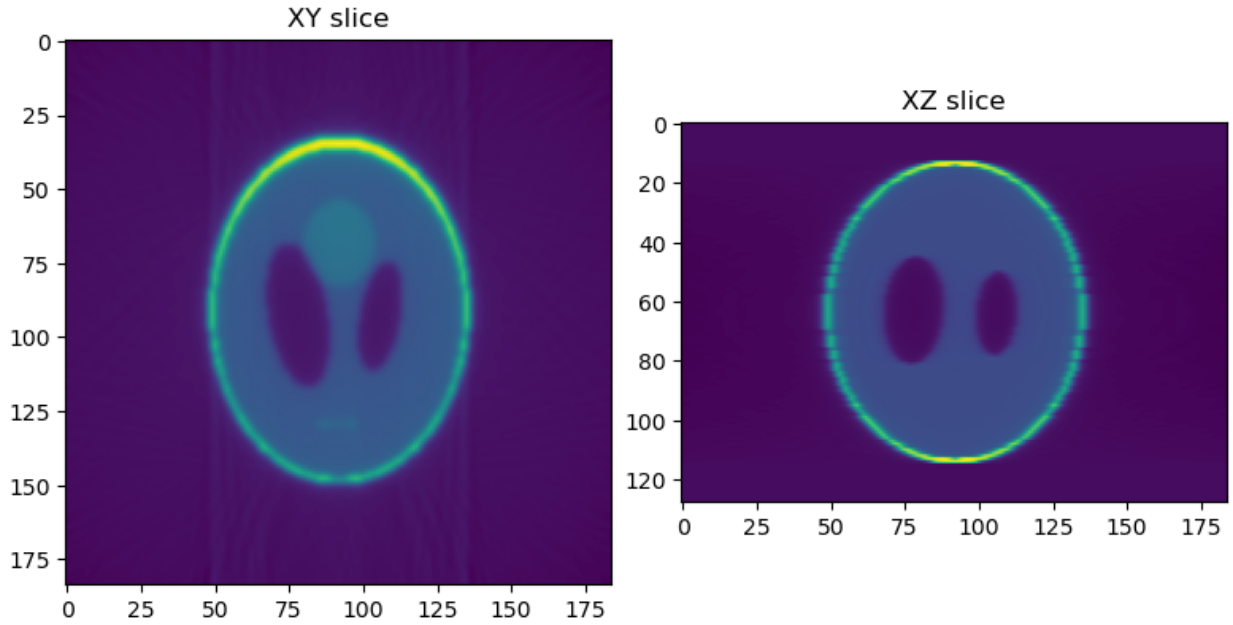
Perform a Full Volume Reconstruction

We will now use *tomopy-cli* to reconstruct the full volume. This is done by changing the `--reconstruction-type` option:

```
$ tomopy recon --file-name phantom_projections.h5 --reconstruction-type=full --output-
↪folder=_rec --output-format=tiff_stack
```

Reconstructions are again stored in *_rec*, but full volume reconstructions can be found in *_rec/phantom_projections_rec*. Again, **open a python console** to visualize the results:

```
>>> import matplotlib.pyplot as plt, dxchange
>>> recon = dxchange.read_tiff_stack("_rec/phantom_projections_rec/recon_00000.tiff",␣
↪ind=range(0, 128))
>>> fig, (axL, axR) = plt.subplots(1, 2)
>>> axL.imshow(recon[64])
>>> axL.set_title("XY slice")
>>> axR.imshow(recon[:,92])
>>> axR.set_title("XZ slice")
>>> plt.show(block=False)
```



Modify Reconstruction Behavior

Now we will use some of tomopy-cli's command line options to modify the reconstruction behavior. We will add four features to our reconstruction:

1. Downsample (bin) the data by a factor of 2
2. Manually specify the rotation center
3. Remove any NaN or infinity values

```
$ tomopy recon --file-name phantom_projections.h5 --reconstruction-type=slice --output-
↪folder=_rec --binning=1 --rotation-axis-auto=manual --rotation-axis=91.5 --fix-nan-and-
↪inf --fix-nan-and-inf-value=34.05
```

Now we will visualize the new reconstruction. **Open a python console** again and use matplotlib to view the reconstructed data.

Note: If you ran the previous slice reconstruction command more than once, you may have to modify the file name in the example below, since each time tomopy is run, a new tiff file is saved.

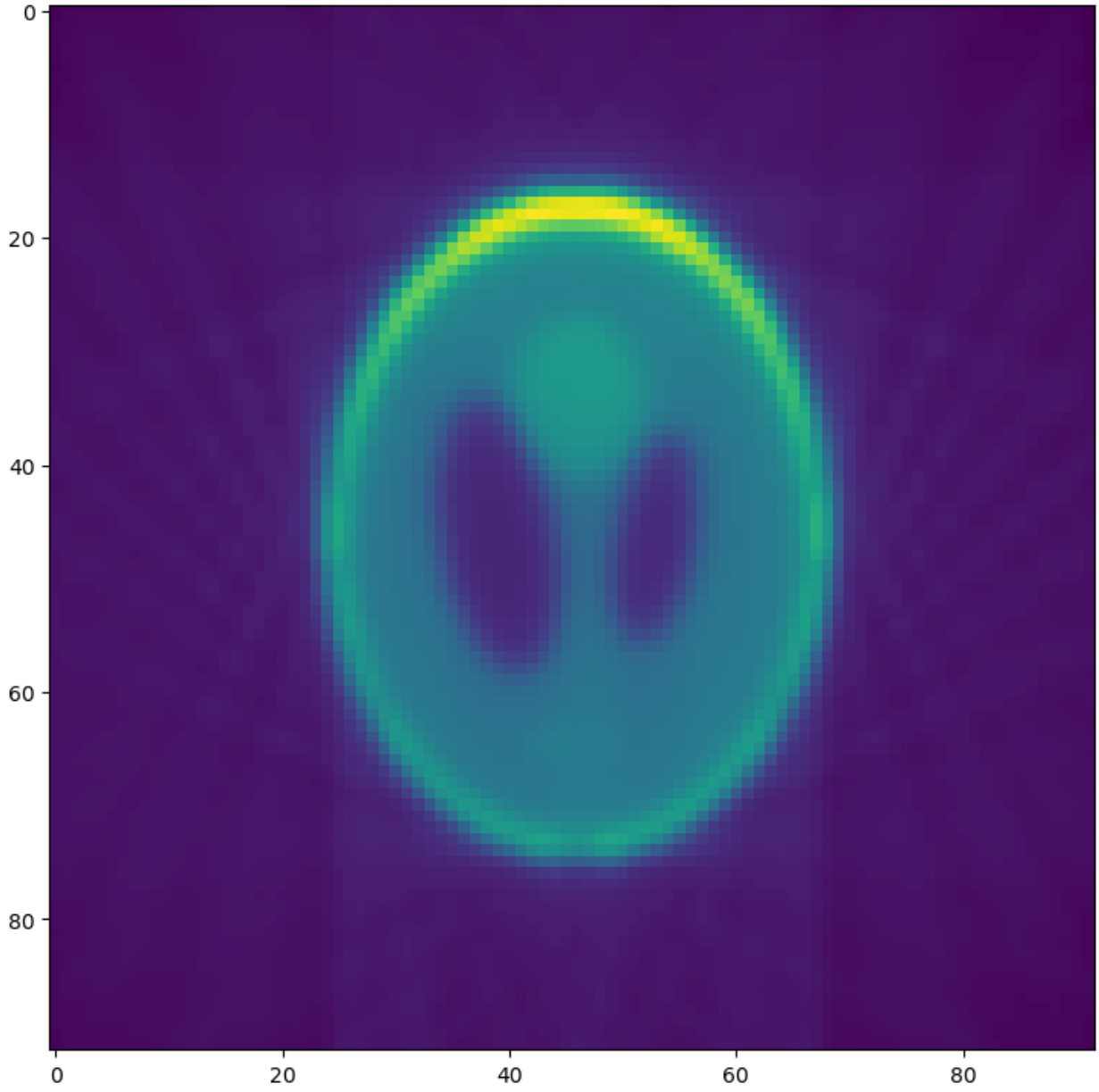
In this case, if the previous command was run twice, this would create files *recon_phantom_projections.tiff* and

recon_phantom_projections-1.tiff, so we would now need to replace the file name with *recon_phantom_projections-2.tiff*

```
>>> import matplotlib.pyplot as plt, imageio
>>> slc = imageio.read("_rec/slice_rec/recon_phantom_projections-1.tiff").get_data(0)
>>> plt.figure()
>>> plt.imshow(slc)
>>> plt.show(block=False)
```

Comparing this reconstruction with the previous versions, the major difference is the lower resolution and noise resulting from the `--binning=1` option. Since we are using well-behaved test data, the other options have no noticeable effect.

Details for these options and many more can be seen by running `tomopy recon --help` from the command line.



3.3 Usage

3.3.1 Reconstruction

To do a tomographic reconstruction:

```
$ tomopy recon --file-name /local/data.h5
```

from the command line. To get correct results, you will likely need to provide *reconstruction parameters* as options, such as `--rotation-axis` to set the rotation axis position:

```
$ tomopy recon --rotation-axis 1024.0 --file-name /local/data.h5
```

To list all available options, use:

```
$ tomopy recon -h
```

3.3.2 Reconstruction Parameters

The behavior of tomopy-cli is controlled by reconstruction parameters, which can be given in one of three ways, in order of precedence:

1. Directly as *Command Line Arguments* (e.g. `--rotation-axis=1024.0`).
2. Per-tomogram in a *YAML Parameter File* given as the argument to the `--parameter-file` option.
3. In the *Global Configuration File*.

Command Line Arguments

The **simplest way** to give a reconstruction parameter is to directly pass it as an option to the `tomopy` command. Some options also accept an argument, while others simple enable certain behavior. Parameters given directly via the command line will override those given via a parameter file or global configuration file.

For example, to specify the rotation center of 1023, apply Vo's stripe removal algorithms, and enable automatic reading of the pixel size for a tomogram stored in *my_tomogram.h5*, use:

```
$ tomopy recon --file-name my_tomogram.h5 --rotation-center 1023.0 --remove-stripe-
↪method vo-all --pixel-size-auto
```

YAML Parameter File

In some cases, many tomograms need to be reconstructed with slightly different parameters. A common example of this is the rotation axis (discussed in detail *below*), which may drift over the course of an operando experiment. This can be done with a YAML file containing the extra parameters for each tomogram:

```
my_tomogram_001.h5:
  rotation-axis: 1023.0
my_tomogram_002.h5:
  rotation-axis: 1025.0
  reconstruction-algorithm: gridrec
```

Any parameters specified in this way will override those in the *Global Configuration File*. The filenames listed in the YAML file can be relative to the current working directory, including subdirectories, but cannot use other file-system shortcuts (e.g. "..", "~").

Warning: *tomopy-cli* does not modify parameters other than those given. In the above example, specifying a rotation axis will have no effect if `--rotation-axis-auto=auto`, since the rotation axis will be calculated, and the value of `--rotation-axis` is then ignored.

Global Configuration File

Reconstruction parameters can also be stored in a global configuration file specified with the `--config` option (default: *tomopy.conf*). You can create a template with:

```
$ tomopy init
```

If *tomopy-cli* is invoked with the `--config-update` option, then the configuration file is updated to keep track of the last stored parameters, as initalized by `$ tomopy init` or modified by setting a new option value. For example to re-run the last reconstrusction with identical parameters just use:

```
$ tomopy recon
```

To run a reconstruction with a different and previously stored configuration file *alternate_tomopy.conf* just use:

```
$ tomopy recon --config alternate_tomopy.conf
```

3.3.3 Output Folder

The output folder for reconstructed data can be given with the `--output-folder` option. The other configuration parameters can be inserted with curly braces:

```
$ tomopy recon --output-folder={file_name}_rec
```

An additional parameter (`{file_name_parent}`) is available with the path of the parent directory. If `--file-name` is a directory, then `{file_name_parent}` will contain the directory itself. If `--file-name` is a file, then `{file_name_parent}` will be the parent directory of the file. The following lines will both place reconstructed data in the directory */path/to/my/data_rec/*:

```
$ tomopy recon --file-name=/path/to/my/data/file.hdf --output-folder={file_name_parent}_
↪rec/
$ tomopy recon --file-name=/path/to/my/data/ --output-folder={file_name_parent}_rec/
```

3.3.4 Find Center

To automatically find the rotation axes locations of all tomographic HDF data sets in a folder (e.g. */local/data/*), use:

```
$ tomopy find_center --file-name /local/data/
```

this generates, in the */local/data/* directory, a YAML file (default: *extra_params.yaml*) containing all the automatically calculated centers:

```
proj_0000.hdf:
    rotation-axis: 1287.25
proj_0001.hdf:
    rotation-axis: 1297.75
proj_0002.hdf:
    rotation-axis: 1287.25
proj_0003.hdf:
    rotation-axis: 1297.75
proj_0004.hdf:
    rotation-axis: 1287.25
proj_0005.hdf:
    rotation-axis: 1297.75
```

If the YAML file already exists, it will be updated with the new rotation axes.

To list all available options:

```
$ tomopy find_center -h
```

After using `$ tomopy find_center`, one can do tomographic reconstructions of all tomographic HDF data sets in a folder (e.g. */local/data/*) with:

```
$ tomopy recon --file-name /local/data/
```

3.3.5 Stripe Removal

Several methods of stripe removal are available in *tomopy-cli*, and can be selected with the `--remove-stripe-method` parameter. Each method may also have a set of associated parameters for controlling its behavior (e.g. `--remove-stripe-method=fw` relies on `--fw-sigma`, `--fw-filter`, etc.).

More information about each method and the accompanying parameters can be found in the corresponding *tomopy* documentation:

Method	`--remove-stripe-method=` Value	Tomopy Function
Fourier-Wavelet	fw	remove_stripe_fw()
Titarenko	ti	remove_stripe_ti()
Smoothing Filter	sf	remove_stripe_sf()
Vo's Algorithms	vo-all	remove_all_stripe()

3.3.6 Help

```
$ tomopy -h
usage: tomopy [-h] [--config FILE] [--version]  ...

optional arguments:
  -h, --help     show this help message and exit
  --config FILE  File name of configuration file
  --version      show program's version number and exit

Commands:

    init         Create configuration file
    recon        Run tomographic reconstruction
    status       Show the tomographic reconstruction status
    segment      Run segmentation on reconstured data
    find_center  Find rotation axis location for all hdf files in a directory
    convert      Convert pre-2015 (proj, dark, white) hdf files in a single
                 data exchange h5 file
```

3.4 Testing

Tomopy-cli contains partial test coverage. For development and testing, a developer installation is recommended:

```
$ git clone https://github.com/tomography/tomopy-cli.git
$ cd tomopy-cli
$ python setup.py develop
```

Tests can be run using the pytest runner:

```
$ pip install pytest
$ pytest
```

3.5 API reference

tomopy-cli Modules:

3.5.1 tomopy_cli.file_io

3.5.2 tomopy_cli.find_center

3.5.3 tomopy_cli.flat_drift_correction

Functions:

chunk(iterable, size)	Splitting by chunks
apply_shift(data, p)	Apply (p[0],p[1]) shift of data
find_min_max(flat)	Find min and max values according to histogram
register_shift_sift(data, flat)	Find shifts via SIFT detecting features
flat_drift_correction(params)	Fix drift of flat field during data acquistion by using a small region not containing the sample.

tomopy_cli.flat_drift_correction.**apply_shift**(*data*, *p*)

Apply (p[0],p[1]) shift of data

tomopy_cli.flat_drift_correction.**chunk**(*iterable*, *size*)

Splitting by chunks

tomopy_cli.flat_drift_correction.**find_min_max**(*flat*)

Find min and max values according to histogram

tomopy_cli.flat_drift_correction.**flat_drift_correction**(*params*)

Fix drift of flat field during data acquistion by using a small region not containing the sample. Note: the method may not work if the region contains a part of the sample

tomopy_cli.flat_drift_correction.**register_shift_sift**(*data*, *flat*)

Find shifts via SIFT detecting features

3.5.4 tomopy_cli.post

Functions:

segment(params)	

tomopy_cli.post.**segment**(*params*)

3.5.5 tomopy_cli.prep

3.5.6 tomopy_cli.recon

3.5.7 tomopy_cli.util

Functions:

theta_step(start, end, proj_number)	
positive_int(value)	Convert *value* to an integer and make sure it is positive.
range_list(value)	Split *value* separated by ':' into int triple, filling missing values with 1s.
restricted_float(x)	
guess_center(first_projection, last_projection)	Compute the tomographic rotation center based on cross-correlation technique.
update_dict(original, new)	Recursively update a dictionary in place with new values.

tomopy_cli.util.**guess_center**(*first_projection*, *last_projection*)

Compute the tomographic rotation center based on cross-correlation technique. *first_projection* is the projection at 0 deg, *last_projection* is the projection at 180 deg.

tomopy_cli.util.**positive_int**(*value*)

Convert *value* to an integer and make sure it is positive.

tomopy_cli.util.**range_list**(*value*)

Split *value* separated by ':' into int triple, filling missing values with 1s.

tomopy_cli.util.**restricted_float**(*x*)

tomopy_cli.util.**theta_step**(*start*, *end*, *proj_number*)

tomopy_cli.util.**update_dict**(*original: Mapping*, *new: Mapping*) → Mapping

Recursively update a dictionary in place with new values.

This is distinct from the python `dict.update` method in that it respects existing entries and just updates them with new values.

https://stackoverflow.com/questions/3232943/update-value-of-a-nested-dictionary-of-varying-depth

Parameters

- **original** – The target dictionary that will be updated.
- **new** – The dictionary with new values that will be added to *original*.

Returns

original – Same dictionary that was passed, with values modified in place.

3.6 Credits

3.6.1 Citations

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

3.6.2 References

BIBLIOGRAPHY

[A1] Gürsoy D, De Carlo F, Xiao X, and Jacobsen C. Tomopy: a framework for the analysis of synchrotron tomographic data. *Journal of Synchrotron Radiation*, 21(5):1188–1193, 2014.

[B1] Francesco De Carlo, Doga Gursoy, Daniel Jackson Ching, Kees Joost Batenburg, Wolfgang Ludwig, Lucia Mancini, Federica Marone, Rajmund Mokso, Daniel M. Pelt, Jan Sijbers, and Mark Rivers. Tomobank: a tomographic data repository for computational x-ray science. *Measurement Science and Technology*, 2017. URL: https://doi.org/10.1088/1361-6501/aa9c19.

[B2] 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.

PYTHON MODULE INDEX

t

tomopy_cli, 20
tomopy_cli.flat_drift_correction, 18
tomopy_cli.post, 19
tomopy_cli.util, 19

INDEX

A

apply_shift() (*in module tomopy_cli.flat_drift_correction*), 19

C

chunk() (*in module tomopy_cli.flat_drift_correction*), 19

F

find_min_max() (*in module tomopy_cli.flat_drift_correction*), 19
flat_drift_correction() (*in module tomopy_cli.flat_drift_correction*), 19

G

guess_center() (*in module tomopy_cli.util*), 20

M

module
    tomopy_cli, 20
    tomopy_cli.flat_drift_correction, 18
    tomopy_cli.post, 19
    tomopy_cli.util, 19

P

positive_int() (*in module tomopy_cli.util*), 20

R

range_list() (*in module tomopy_cli.util*), 20
register_shift_sift() (*in module tomopy_cli.flat_drift_correction*), 19
restricted_float() (*in module tomopy_cli.util*), 20

S

segment() (*in module tomopy_cli.post*), 19

T

theta_step() (*in module tomopy_cli.util*), 20
tomopy_cli
    module, 20
tomopy_cli.flat_drift_correction
    module, 18
tomopy_cli.post
    module, 19
tomopy_cli.util
    module, 19

U

update_dict() (*in module tomopy_cli.util*), 20