AXITOM Documentation

Release 0.1

PolymerGuy

Oct 10, 2019

Getting started:

This python package provides tools for axis-symmetric cone-beam computed tomography. A Feldkamp David Kress algorithm performs the reconstruction which have been adapted such that is exploits the axis-symmetric nature of the tomogram.

This toolkit has a highly specialised usage, and there are plenty of more general and excellent frameworks for tomographic reconstruction, such as:

- TomoPy (https://github.com/tomopy/tomopy) General computed tomography, cone and parallel beam geometry
- PyAbel (https://github.com/PyAbel/PyAbel) Computed tomography based on the inverse Abel transform, parallel beam geometry

This project aims at providing a simple, accessible toolkit for forward-projection and reconstruction of axis-symmetric tomograms based on a conical beam geometry.

CHAPTER 1

Installation

In order to get started with AXITOM, you need to install it on your computer. There are two main ways to to this:

- You can install it via a package manager like PIP
- You can clone the repo

1.1 Install via package manager:

The toolkit is available via PIP, and the instructions below shows how a virtual environment can be created and the toolkit installed.

Prerequisites:

```
This toolkit is tested on Python 3.6
We recommend the use of virtualenv
```

Installing:

```
virtualenv -p python3.6 env
source ./env/bin/activate #On Linux and Mac OS
env\Scripts\activate.bat #On Windows
pip install axitom
```

Now the toolkit is installed and ready for use.

Run the tests:

```
nosetests axitom
```

If you want to check out the examples, then download the files in the examples folder and run the examples.

1.2 By cloning the repo:

These instructions will get you a copy of the project up and running on your local machine for development and testing purposes.

Prerequisites:

```
This toolkit is tested on Python 3.6
We recommend the use of virtualenv
```

Start to clone this repo to your preferred location:

```
git init
git clone https://github.com/PolymerGuy/axitom.git
```

We recommend that you always use virtual environments, either by virtualenv or by Conda env

Virtual env:

```
virtualenv -p python3.6 env
source ./env/bin/activate #On Linux and Mac OS
env\Scripts\activate.bat #On Windows
pip install -r requirements.txt
```

You can now run an example:

```
$ python path_to_axitom/examples/comparison_to_Nikon.py
```

1.3 Running the tests

The tests should always be launched to check your installation. These tests are integration and unit tests

If you cloned the repo, you have to call pytest from within the folder:

```
pytest
```

CHAPTER 2

Quick start

Let's now go through the necessary steps for doing reconstruction of a tomogram based on a single image. First, we need to import the tools:

```
import axitom as tom
from scipy.ndimage.filters import median_filter
```

The example data can be downloaded from the AXITOM/tests/example_data/ folder. The dataset was collected during tensile testing of a polymer specimen. Assuming that the example data from the repo is located in root folder, we can make a config object from the .xtekct file:

```
config = tom.config_from_xtekct("radiogram.xtekct")
```

We now import the projection:

```
projection = tom.read_image(r"radiogram.tif", flat_corrected=True)
```

As we will use a single projection only in this reconstruction, we will reduce the noise content of the projection by employing a median filter. This works fine since the density gradients within the specimen are relatively small. You may here choose any filter of your liking:

```
projection = median_filter(projection, size=21)
```

Now, the axis of rotation has to be determined. This is done be binarization of the image into object and background and determining the center of gravity of the object:

```
_, center_offset = tom.object_center_of_rotation(projection, background_internsity=0.
↪9)
```

The config object has to be updated with the correct values:

```
config = config.with_param(center_of_rot=center_offset)
```

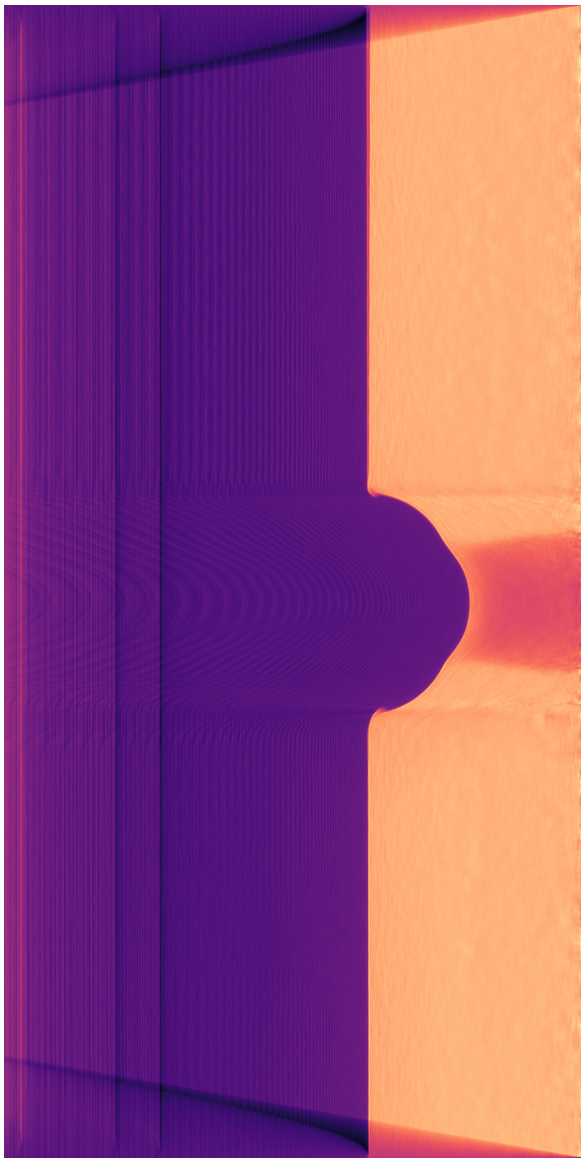
We are now ready to initiate the reconstruction:

```
tomo = tom.fdk(projection, config)
```

The results can then be visualized:

```
import matplotlib.pyplot as plt
plt.title("Radial slice")
plt.imshow(tomo.transpose(), cmap=plt.cm.magma)
```

and looks like this:



CHAPTER 3

API documentation

This document contains the documentation contained within the docstrings of all available functions and classes.

3.1 Back projection

axitom.backprojection.**map_object_to_detector_coords**(*object_xs*, *object_ys*, *object_zs*, *config*)

Map the object coordinates to detector pixel coordinates accounting for cone beam divergence.

Parameters

- **object_xs** (*np.ndarray*) – The x-coordinate array of the object to be reconstructed
- **object_ys** (*np.ndarray*) – The y-coordinate array of the object to be reconstructed
- **object_zs** (*np.ndarray*) – The z-coordinate array of the object to be reconstructed
- **config** (*obj*) – The config object containing all necessary settings for the reconstruction

Returns

- *detector_cords_a* – The detector coordinates along the a-axis corresponding to the given points
- *detector_cords_b* – The detector coordinates along the b-axis corresponding to the given points

axitom.backprojection.**fdk**(*projection*, *config*)

Filtered back projection algorithm as proposed by Feldkamp David Kress, adapted for axisymmetry.

This implementation has been adapted for axis-symmetry by using a single projection only and by only reconstructing a single R-Z slice.

This algorithm is based on: https://doi.org/10.1364/JOSAA.1.000612

but follows the notation used by: Turbell, H. (2001). Cone-Beam Reconstruction Using Filtered Backprojectionn. Science And Technology. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.134.5224&rep=rep1&type=pdf

Parameters

- **projection** (*np.ndarray*) – The projection used in the reconstruction
- **config** (*obj*) – The config object containing all necessary settings for the reconstruction

Returns The reconstructed slice is a R-Z plane of a axis-symmetric tomogram where Z is the symmetry axis.

Return type ndarray

3.2 Forward projection

axitom.projection.**axis_sym_projection**(*volume*, *config*, *angles=None*)

Projection of a volume onto a sensor plane using a conical beam geometry

This implementation has been adapted for axis-symmetry and returns a single projection only. The forward projection is done for several angles and the average projection is returned.

Parameters

- **volume** (*np.ndarray*) – The volume which will be projected onto the sensor
- **config** (*obj*) – The settings object
- **angles** (*list*) – The angles in degrees which will be used for the forward projection

Returns The projection

Return type ndarray

3.3 Config

axitom.config.**config_from_xtekct**(*file_path*)

Make config object from a Nikon X-tek CT input file

The .xtekct file is parsed and a config file containing all relevant settings is returned.

Parameters **file_path** (*str*) – The path to the .xtekct file

Returns Config object

Return type obj

class axitom.config.**Config**(*n_pixels_u*, *n_pixels_v*, *detector_size_u*, *detector_size_v*, *source_to_detector_dist*, *source_to_object_dist*, *angular_inc=1*, *center_of_rot=0*, ***kwargs*)

Methods

with_param(self, **kwargs)	Get a clone of the object with the parameter changed and all calculations repeated

__init__(*self*, *n_pixels_u*, *n_pixels_v*, *detector_size_u*, *detector_size_v*, *source_to_detector_dist*, *source_to_object_dist*, *angular_inc=1*, *center_of_rot=0*, ***kwargs*)

Configuration object which contains all settings necessary for the forward projection and tomo-

graphic reconstruction.

Parameters

- **n_pixels_u** (*int*) – Number of pixels in the u direction of the sensor
- **n_pixels_v** (*int*) – Number of pixels in the u direction of the sensor
- **detector_size_u** (*float*) – Detector size in the u direction [mm]
- **detector_size_v** (*float*) – Detector size in the u direction [mm]
- **source_to_detector_dist** (*float*) – Distance between source and detector [mm]
- **source_to_object_dist** (*float*) – Distance between source and object [mm]
- **angular_inc** (*float*) – Angular increment used in the reconstruction [deg]
- **center_of_rot** (*float*) – Position of the rotation axis in pixels. 0 corresponds to the center of the image
- **NOTE** (*Any non valid arguments are neglected without warning*) –

with_param(*self*, ***kwargs*)

Get a clone of the object with the parameter changed and all calculations repeated

Parameters **kwargs** – The arguments of the config object that should be changed

Returns Config object with new arguments set

Return type obj

3.4 Filtering

axitom.filtering.**ramp_filter_and_weight**(*projection*, *config*)

Add weights to the projection and apply a ramp-high-pass filter set to 0.5*Nyquist_frequency

Parameters

- **projection** (*np.ndarray*) – The projection used in the reconstruction
- **config** (*obj*) – The config object containing all necessary settings for the reconstruction

Returns The projections weighted by the ray length and filtered by ramp filter

Return type ndarray

3.5 Parse files

3.6 Phantoms

axitom.phantoms.**barrel**(*domain_size=128*, *outer_rad_fraction=0.7*, *center_val=None*)

Barrel shaped phantom with a linear density gradient The domain size is cubic with dimension "domain_size" along each axis

Parameters

- **domain_size** (*int*) – The length of the sides of the domain

- **outer_rad_fraction** (*float*) – The diameter of the barrel given as a the fraction of the side length
- **center_val** (*float*) – The density value in the center of the barrel

Returns The phantom

Return type ndarray

3.7 Utilities

axitom.utilities.**rotate_coordinates**(*xs_array*, *ys_array*, *angle_rad*)

Rotate coordinate arrays by a given angle

Parameters

- **xs_array** (*ndarray*) – Two dimensional coordinate array with x-coordinates
- **ys_array** (*ndarray*) – Two dimensional coordinate array with y-coordinates
- **angle_rad** (*float*) – Rotation angle in radians

Returns

- *ndarray* – The rotated x-coordinates
- *ndarray* – The rotated x-coordinates

axitom.utilities.**list_files_in_folder**(*path*, *file_type='.tif'*)

List all files with a given extension for a given path. The output is sorted

Parameters

- **path** (*str*) – Path to the folder containing the files
- **file_type** (*str*) – The file extension ex. ".tif"

Returns A list of sorted file names

Return type list

axitom.utilities.**shading_correction**(*projections*, *flats*, *darks*)

Perform shading correction on a a list of projections based on a list of flat images and list of dark images.

The correction is: corrected = (projections-dark)/(flat-dark)

Parameters

- **projections** (*list*) – A list of projections that will be corrected
- **flats** (*list*) – A list of flat images
- **darks** (*list*) – A list of dark images

Returns A list of corrected projections

Return type list

axitom.utilities.**read_image**(*file_path*, *flat_corrected=False*)

Read an image specified by the file_path This function always returns a transposed float64 single channel image

Parameters **file_path** (*str*) – Path to the file

Returns The single channel image with float64 values

Return type ndarray

CHAPTER 4

FDK

The Feldkamp David Kress (FDK) algorithm is often used in the reconstruction of tomographic data where the radiograms are acquired using a conical X-ray beam. The original article is found here:

> Feldkamp, L. A. (1984). Practical cone-beam algorithm Sfrdr I _ f. America, 1(6), 612–619. https://doi.org/10.1364/JOSAA.1.000612

The notation we will use in the following document is taken from this thesis:

> Turbell, H. (2001). Cone-Beam Reconstruction Using Filtered Backprojectionn. Science And Technology. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.134.5224&rep=rep1&type=pdf

Full 3D reconstruction is usually performed using this algorithm, but we will here modify is slightly to more efficiently reconstruct an axisymmetric tomogram.

4.1 Full 3D reconstruct

In its original form, the reconstructed tomogram $f_{FDK}(x, y, z)$ is determined by the following equation:

$$f_{FDK}(x, y, z) = \int_0^{2\pi} \frac{R^2}{U(x, y, \beta)^2} \tilde{p}^F(\beta, a(x, y, \beta), b(x, y, z, \beta)) d\beta \tag{4.1}$$

where

$$\tilde{p}^F(\beta, a, b) = (\frac{R}{\sqrt{R^2 + a^2 + b^2}} p(\beta, a, b)) * g^P(a) \tag{4.2}$$

is the filtered and weighted radiograms. $p(\beta, a, b)$ is the radiogram acquired for angle β wheras a and b denotes the sensor coordinates and R is the sensor to specimen distance. A ramp filter $g^P(a)$ is applied in the horizontal direction of the sensor by means of convolution.

The term $U(x, y, \beta)$ is determined by:

$$U(x, y, \beta) = R + x \cos\beta + y \sin\beta \tag{4.3}$$

where x and y are coordinates to material points in the specimen.

The sensor coordinates a and b corresponding to the material point defined by the x, y and z coordinates, for a given angle β can be determined by:

$$a(x, y, \beta) = R\frac{-x\sin\beta + y\cos\beta}{R + x\cos\beta + y\sin\beta} \tag{4.4}$$

$$b(x, y, z\beta) = z\frac{R}{R + x\cos\beta + y\sin\beta} \tag{4.5}$$

4.2 Axis-symmetry

In the case where the tomogram $f_{FDK}(x, y, z)$ is axisymmetric around a rotational axis tomogram z, all radial slices of the tomogram should be equal.

We here reduce the tomographic problem by assuming that all projections p are independent of β, and we reconstruct only the plaing laying in $x = 0$ giving:

$$f_{FDK}(y, z) = \int_0^{2\pi} \frac{R^2}{U(y, \beta)^2}\tilde{p}^F(a(y, \beta), b(y, z, \beta))d\beta \tag{4.6}$$

where

$$a(y, \beta) = R\frac{y\cos\beta}{R + y\sin\beta} \tag{4.7}$$

$$b(y, z\beta) = z\frac{R}{R + y\sin\beta} \tag{4.8}$$

The values of $\tilde{p}^F(a(x, y, \beta), b(x, y, z, \beta))$ are obtained by means of interpolation employing bi-cubic splines.

CHAPTER 5

Comparison to NIKON CT-PRO

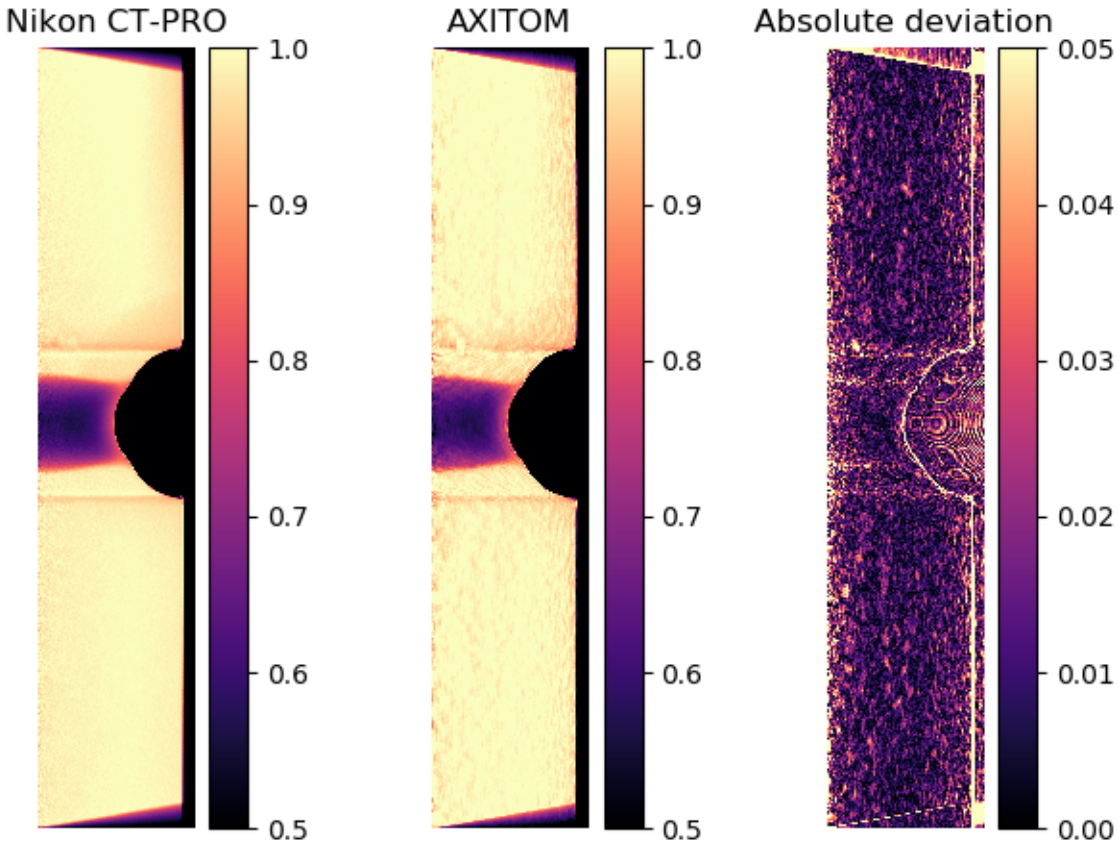
As a verification, the results from the FDK algorithm are compared to results obtained from NIKON CT-PRO. An axis-symmetric tensile specimen is used in this study.

We do this by first acquiring a full dataset (3142 projections) and reconstruct the tomogram using the proprietary software NIKON CT-PRO. As the tomogram is axis-symmetric along the centre axis of the specimen, we reslice the tomogram for all R-Z planes and compute the average. This slice is then considered as the reference slice and is our target.

We then pick a single radiogram from the same dataset, filter the radiogram using a median low pass filter, and reconstruct a tomogram using the FDK algorithm modified for axis-symmetry.

5.1 Comparison of a radial slice

A comparison of the results from NIKON CT-PRO and AXITOM is shown below. The grey scale values have been normalized for easier comparison. There is a close correspondence between the results and the absolute deviation is in the order of 2%, here is considered a satisfactory result.


Nikon CT-PRO
1.0
0.9
0.8
0.7
0.6
0.5
AXITOM
1.0
0.9
0.8
0.7
0.6
0.5
Absolute deviation
0.05
0.04
0.03
0.02
0.01
0.00

Index

Symbols

A

B

C

F

L

M

R

S

W