# microCT Data Processing Guidelines

Andy Kiss

August 24, 2017

## ABSTRACT

Here is a brief guideline for processing microtomgraphy data collected using the microCT at SLAC/SSRL. This document can be used to reconstruct projection data into meaningful 3D datasets but it should be noted that some tweaks may need to be performed from this outline.

## CONTENTS

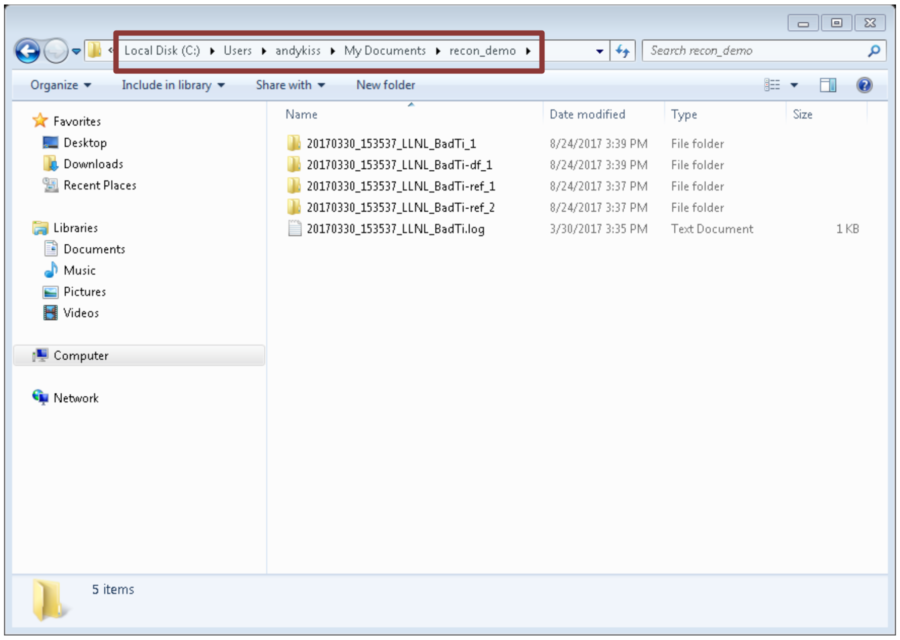
## PREREQUISITES

Below is a list of software used in the following outline.

- Python 3+
  - Anaconda distribution (recommended), using Spyder
  - Go to website https://www.continuum.io/downloads and download Anaconda installation package.
  - Packages:
    - Scipy
    - Numpy
    - Pywavelets
    - ASTRA (http://www.astra-toolbox.com)
- Py_image_processing
  - A series of scripts on GitHub for this reconstruction
  - These can be downloaded from https://github.com/slaclab/py_image_processing
- FIJI (ImageJ 2.0)
  - Not necessary but helpful for viewing and manipulating data
  - Go to website https://fiji.sc/#download and download FIJI zipped folder.
  - Unzip the FIJI folder
  - In Fiji.app folder double click ImageJ-win32 to launch the software.
- TXM Wizard
  - Necessary for mosaic imaging
  -

## Convert TIFF Stacks to Reference Corrected Projection Images

- Locate the folder where you saved your data. There should be a scan log file (*.log) as well as folders with your transmission images, dark field, and flat field image stacks.

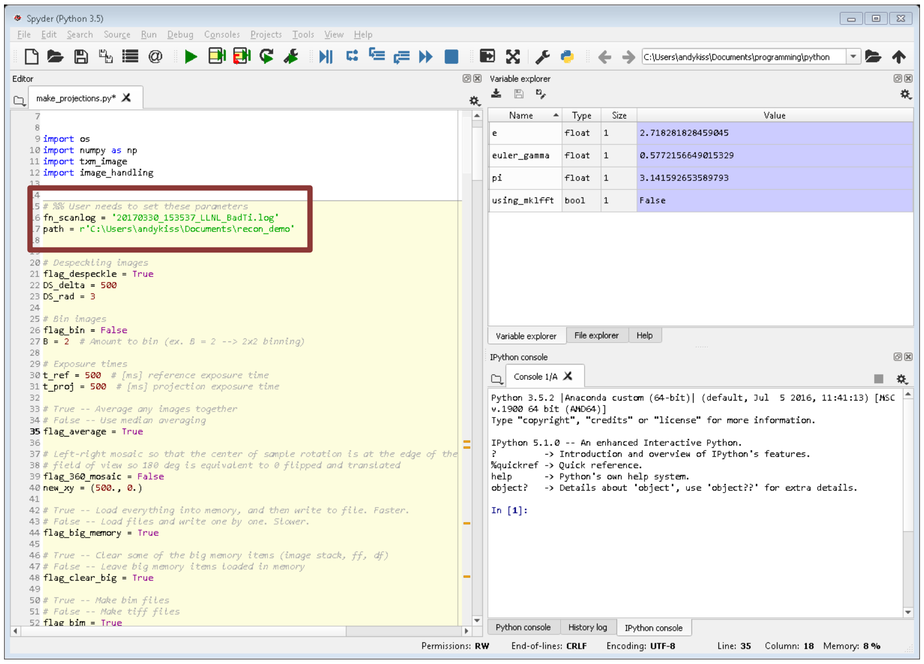


Local Disk (C:) ▸ Users ▸ andykiss ▸ My Documents ▸ recon_demo ▸

| Name | Date modified | Type | Size |
|---|---|---|---|
| 20170330_153537_LLNL_BadTi_1 | 8/24/2017 3:39 PM | File folder | |
| 20170330_153537_LLNL_BadTi-df_1 | 8/24/2017 3:39 PM | File folder | |
| 20170330_153537_LLNL_BadTi-ref_1 | 8/24/2017 3:37 PM | File folder | |
| 20170330_153537_LLNL_BadTi-ref_2 | 8/24/2017 3:37 PM | File folder | |
| 20170330_153537_LLNL_BadTi.log | 3/30/2017 3:35 PM | Text Document | 1 KB |

5 items

- Open Spyder and open the *make_projections.py* script.
- Locate the lines that set the scan log file name, *fn_scanlog*, and path, *path*, and set the variables to point to your data.



Spyder (Python 3.5) — make_projections.py*

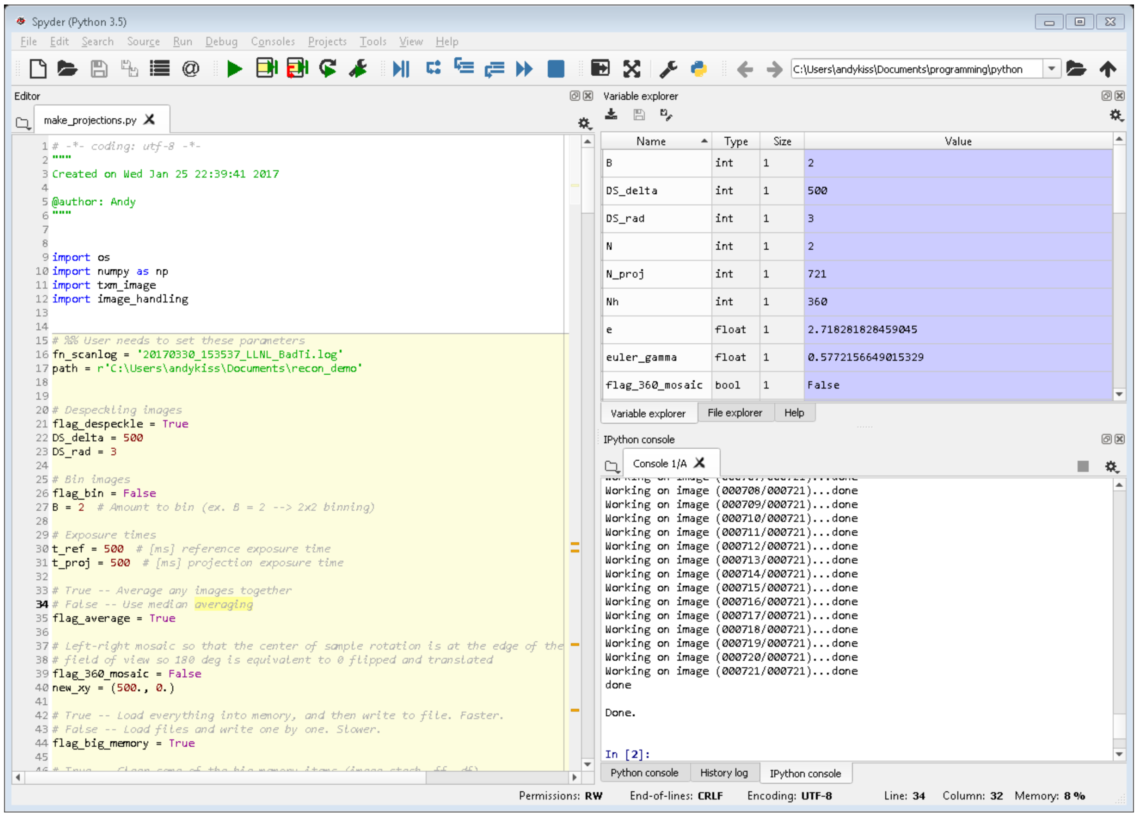
```python


import os
import numpy as np
import txm_image
import image_handling


# %% User needs to set these parameters
fn_scanlog = '20170330_153537_LLNL_BadTi.log'
path = r'C:\Users\andykiss\Documents\recon_demo'


# Despeckling images
flag_despeckle = True
DS_delta = 500
DS_rad = 3

# Bin images
flag_bin = False
B = 2  # Amount to bin (ex. B = 2 --> 2x2 binning)

# Exposure times
t_ref = 500  # [ms] reference exposure time
t_proj = 500  # [ms] projection exposure time

# True -- Average any images together
# False -- Use median averaging
flag_average = True

# Left-right mosaic so that the center of sample rotation is at the edge of the
# field of view so 180 deg is equivalent to 0 flipped and translated
flag_360_mosaic = False
new_xy = (500., 0.)

# True -- Load everything into memory, and then write to file. Faster.
# False -- Load files and write one by one. Slower.
flag_big_memory = True

# True -- Clear some of the big memory items (image stack, ff, df)
# False -- Leave big memory items loaded in memory
flag_clear_big = True

# True -- Make bim files
# False -- Make tiff files
flag_bim = True
```

| Name | Type | Size | Value |
|---|---|---|---|
| e | float | 1 | 2.718281828459045 |
| euler_gamma | float | 1 | 0.5772156649015329 |
| pi | float | 1 | 3.141592653589793 |
| using_mklfft | bool | 1 | False |

```
Python 3.5.2 |Anaconda custom (64-bit)| (default, Jul  5 2016, 11:41:13) [MSC
v.1900 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 5.1.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]:
```

Permissions: RW   End-of-lines: CRLF   Encoding: UTF-8   Line: 35   Column: 18   Memory: 8 %

- Scan through the remaining settings in that block. Many of them can stay the same from reconstruction to reconstruction. If you are interested in modifying the binning of the images or removing zingers, here is where the settings are.
- Note: if you have a data processing computer with sufficient memory to perform all the analysis in memory (>32 GB), check that *flag_big_memory* is set to *True*. This will run the calculations much faster.
- Note: By default, the program will export *.bim files. These are binary image files with the metadata stored in the header. It is recommended that you perform the analysis using this file format. Check that *flag_bim* is set to *True*, otherwise it will export *.tif files.
- When ready, save and run make_projections.py. This will take a while but there will be output to the console so you can track the process.
- The script will make a post-processing folder, *sample-name-processing*, where it will store the averaged dark and flat field images, and a folder where it will export the reference corrected projections.



```python
# -*- coding: utf-8 -*-
"""
Created on Wed Jan 25 22:39:41 2017

@author: Andy
"""


import os
import numpy as np
import txm_image
import image_handling


# %% User needs to set these parameters
fn_scanlog = '20170330_153537_LLNL_BadTi.log'
path = r'C:\Users\andykiss\Documents\recon_demo'


# Despeckling images
flag_despeckle = True
DS_delta = 500
DS_rad = 3

# Bin images
flag_bin = False
B = 2  # Amount to bin (ex. B = 2 --> 2x2 binning)

# Exposure times
t_ref = 500  # [ms] reference exposure time
t_proj = 500  # [ms] projection exposure time

# True -- Average any images together
# False -- Use median averaging
flag_average = True

# Left-right mosaic so that the center of sample rotation is at the edge of the
# field of view so 180 deg is equivalent to 0 flipped and translated
flag_360_mosaic = False
new_xy = (500., 0.)

# True -- Load everything into memory, and then write to file. Faster.
# False -- Load files and write one by one. Slower.
flag_big_memory = True

```

| Name | Type | Size | Value |
|---|---|---|---|
| B | int | 1 | 2 |
| DS_delta | int | 1 | 500 |
| DS_rad | int | 1 | 3 |
| N | int | 1 | 2 |
| N_proj | int | 1 | 721 |
| Nh | int | 1 | 360 |
| e | float | 1 | 2.718281828459045 |
| euler_gamma | float | 1 | 0.5772156649015329 |
| flag_360_mosaic | bool | 1 | False |

```
Working on image (000708/000721)...done
Working on image (000709/000721)...done
Working on image (000710/000721)...done
Working on image (000711/000721)...done
Working on image (000712/000721)...done
Working on image (000713/000721)...done
Working on image (000714/000721)...done
Working on image (000715/000721)...done
Working on image (000716/000721)...done
Working on image (000717/000721)...done
Working on image (000718/000721)...done
Working on image (000719/000721)...done
Working on image (000720/000721)...done
Working on image (000721/000721)...done
done

Done.


In [2]:
```

Permissions: **RW** End-of-lines: **CRLF** Encoding: **UTF-8** Line: **34** Column: **32** Memory: **8 %**

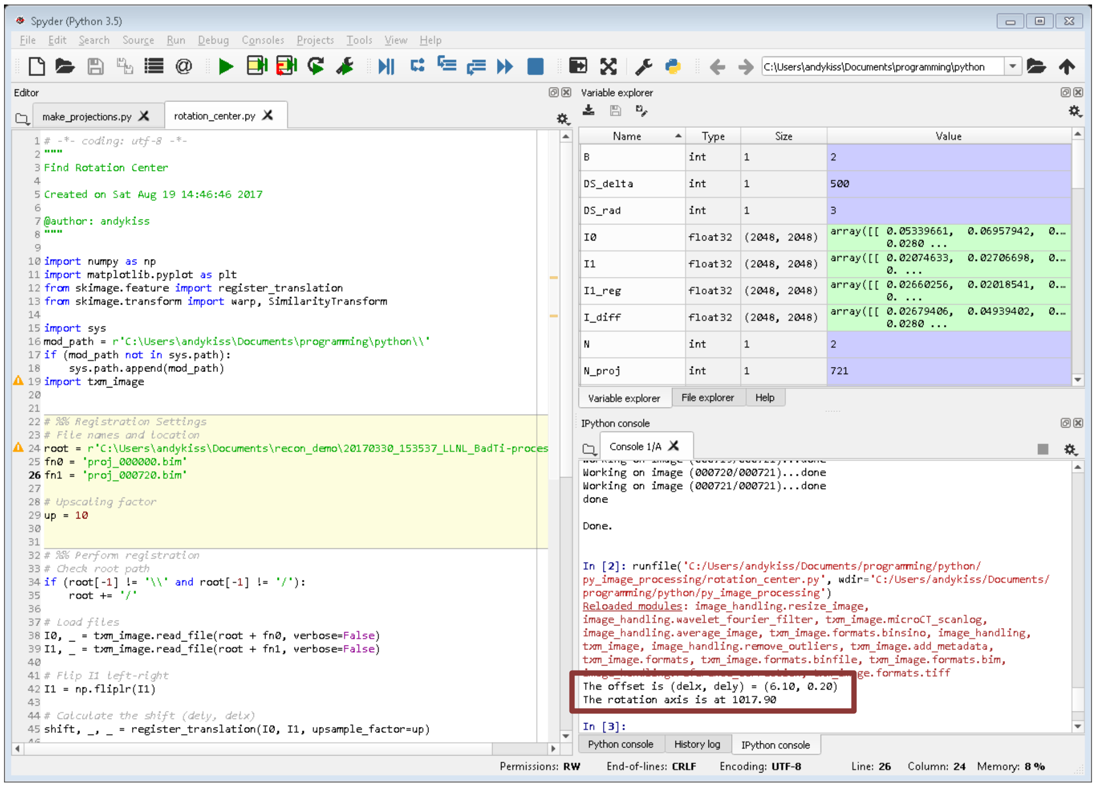
## **MOSAIC STITCHING**

TXM Wizard

- Open *Mosaic Stitcher*
- Click *Browse* and load only the images necessary for the first mosaic image
    - All of the imaging properties should load into the fields on the right
- Start to use the options explained below to stitch together those. Click *Run Stitching* to see the stitched image.
    - *Use PhCo Alignment* will use phase correlation to help align the images for stage errors
    - *Normalization among Tiles* will normalize the edges
    - *Smooth The Edge* will blend the images together
- Once satisfied with the result, enable *MultipleCPU* and make sure the images are saved as **.bim*
- Click *Batch Stitching* and select all the images

## Find the Rotation Center

- In Spyder, open *rotation_center.py*
- Find the Registration Settings cell and change
  - *root* to point to the folder with the reference corrected *.bim files
  - *fn0* to the file name of the first image in your dataset (usually at theta = 0°)
  - *fn1* to the file name of the image in the dataset rotated 180° from the first file
- Save and run the script. The console will output the X and Y shift, as well as the pixel location of the rotation axis.



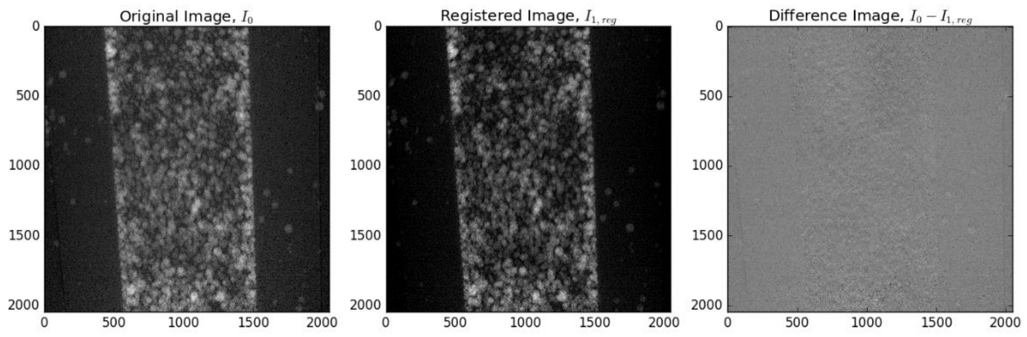
```python
# -*- coding: utf-8 -*-
"""
Find Rotation Center

Created on Sat Aug 19 14:46:46 2017

@author: andykiss
"""

import numpy as np
import matplotlib.pyplot as plt
from skimage.feature import register_translation
from skimage.transform import warp, SimilarityTransform

import sys
mod_path = r'C:\Users\andykiss\Documents\programming\python\\'
if (mod_path not in sys.path):
    sys.path.append(mod_path)
import txm_image


# %% Registration Settings
# File names and location
root = r'C:\Users\andykiss\Documents\recon_demo\20170330_153537_LLNL_BadTi-proces
fn0 = 'proj_000000.bim'
fn1 = 'proj_000720.bim'

# Upscaling factor
up = 10


# %% Perform registration
# Check root path
if (root[-1] != '\\' and root[-1] != '/'):
    root += '/'

# Load files
I0, _ = txm_image.read_file(root + fn0, verbose=False)
I1, _ = txm_image.read_file(root + fn1, verbose=False)

# Flip I1 left-right
I1 = np.fliplr(I1)

# Calculate the shift (dely, delx)
shift, _, _ = register_translation(I0, I1, upsample_factor=up)
```

| Name | Type | Size | Value |
|---|---|---|---|
| B | int | 1 | 2 |
| DS_delta | int | 1 | 500 |
| DS_rad | int | 1 | 3 |
| I0 | float32 | (2048, 2048) | array([[ 0.05339661,  0.06957942,  0.… 0.0280 ... |
| I1 | float32 | (2048, 2048) | array([[ 0.02074633,  0.02706698,  0.… 0.  ... |
| I1_reg | float32 | (2048, 2048) | array([[ 0.02660256,  0.02018541,  0.… 0.  ... |
| I_diff | float32 | (2048, 2048) | array([[ 0.02679406,  0.04939402,  0.… 0.0280 ... |
| N | int | 1 | 2 |
| N_proj | int | 1 | 721 |

```
Working on image (000720/000721)...done
Working on image (000721/000721)...done
done

Done.


In [2]: runfile('C:/Users/andykiss/Documents/programming/python/
py_image_processing/rotation_center.py', wdir='C:/Users/andykiss/Documents/
programming/python/py_image_processing')
Reloaded modules: image_handling.resize_image,
image_handling.wavelet_fourier_filter, txm_image.microCT_scanlog,
image_handling.average_image, txm_image.formats.binsino, image_handling,
txm_image, image_handling.remove_outliers, txm_image.add_metadata,
txm_image.formats, txm_image.formats.binfile, txm_image.formats.bim,
image_handling.reference_correction, txm_image.formats.tiff
The offset is (delx, dely) = (6.10, 0.20)
The rotation axis is at 1017.90

In [3]:
```

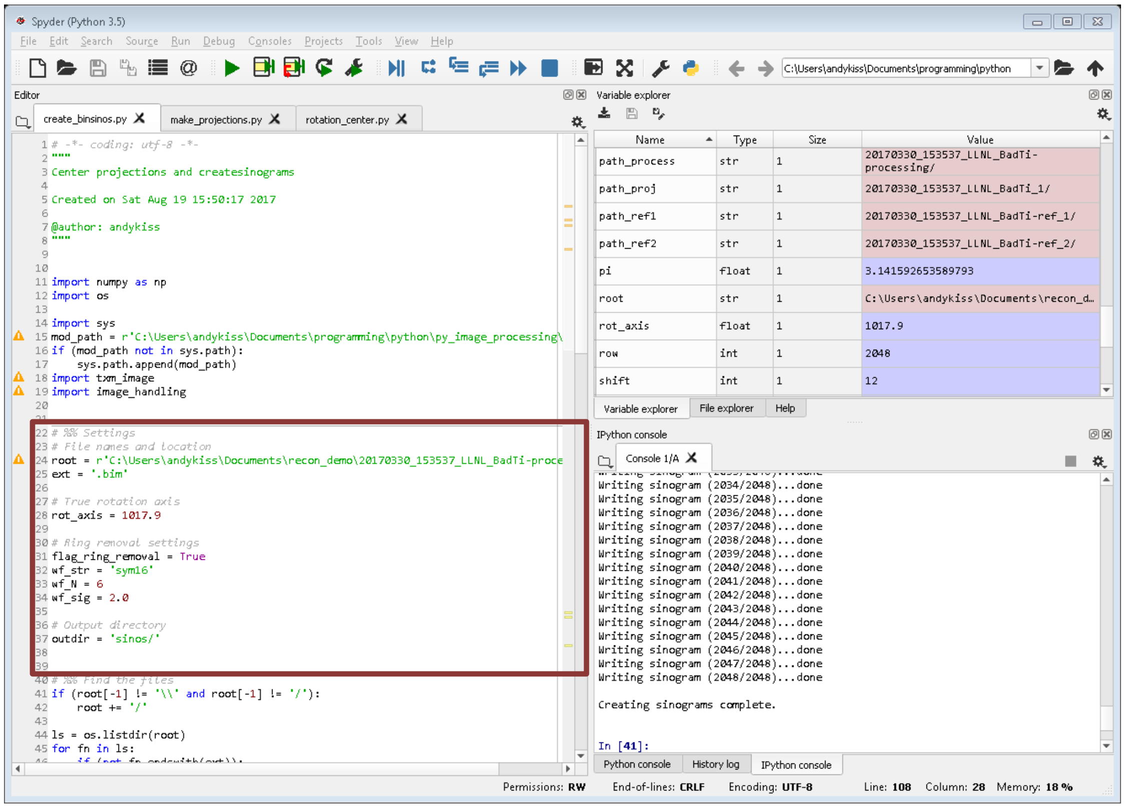
Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 26 Column: 24 Memory: 8 %

- A figure will show the loaded images and the difference map after they have been aligned. The quality of the alignment can be judged based on the difference map. If it is easy to see the features in the original images, then the alignment is poor and will not lead to a high quality reconstruction.


Original Image, $I_0$
Registered Image, $I_{1,reg}$
Difference Image, $I_0 - I_{1,reg}$
0
500
1000
1500
2000

## Create Sinograms

- In Spyder, *open create_binsinos.py*
- Find the Settings cell and change the path variable, *root*, to the folder with the reference corrected projections
- Set the rotation axis, *rot_axis*, to the rotation axis value found in the previous step
- Ring artifacts in the reconstruction can be removed by using a filter. Set the ring removal filter flag, *flag_ring_removal*, to *True* to remove these.
- Run the script and wait for the sinograms to be output to the *sinos* folder



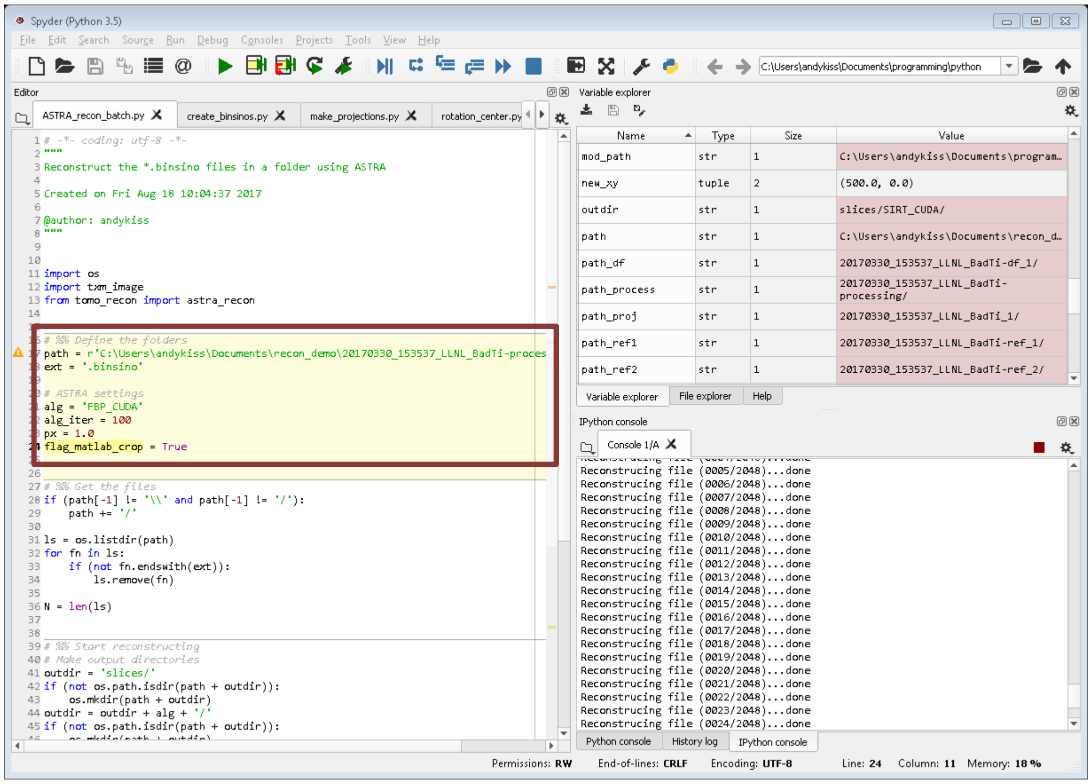
```python
# -*- coding: utf-8 -*-
"""
Center projections and createsinograms

Created on Sat Aug 19 15:50:17 2017

@author: andykiss
"""


import numpy as np
import os

import sys
mod_path = r'C:\Users\andykiss\Documents\programming\python\py_image_processing\
if (mod_path not in sys.path):
    sys.path.append(mod_path)
import txm_image
import image_handling


# %% Settings
# File names and location
root = r'C:\Users\andykiss\Documents\recon_demo\20170330_153537_LLNL_BadTi-proce
ext = '.bim'

# True rotation axis
rot_axis = 1017.9

# Ring removal settings
flag_ring_removal = True
wf_str = 'sym16'
wf_N = 6
wf_sig = 2.0

# Output directory
outdir = 'sinos/'


# %% Find the files
if (root[-1] != '\\' and root[-1] != '/'):
    root += '/'

ls = os.listdir(root)
for fn in ls:
```

| Name | Type | Size | Value |
|---|---|---|---|
| path_process | str | 1 | 20170330_153537_LLNL_BadTi-processing/ |
| path_proj | str | 1 | 20170330_153537_LLNL_BadTi_1/ |
| path_ref1 | str | 1 | 20170330_153537_LLNL_BadTi-ref_1/ |
| path_ref2 | str | 1 | 20170330_153537_LLNL_BadTi-ref_2/ |
| pi | float | 1 | 3.141592653589793 |
| root | str | 1 | C:\Users\andykiss\Documents\recon_d… |
| rot_axis | float | 1 | 1017.9 |
| row | int | 1 | 2048 |
| shift | int | 1 | 12 |

```
Writing sinogram (2034/2048)...done
Writing sinogram (2035/2048)...done
Writing sinogram (2036/2048)...done
Writing sinogram (2037/2048)...done
Writing sinogram (2038/2048)...done
Writing sinogram (2039/2048)...done
Writing sinogram (2040/2048)...done
Writing sinogram (2041/2048)...done
Writing sinogram (2042/2048)...done
Writing sinogram (2043/2048)...done
Writing sinogram (2044/2048)...done
Writing sinogram (2045/2048)...done
Writing sinogram (2046/2048)...done
Writing sinogram (2047/2048)...done
Writing sinogram (2048/2048)...done

Creating sinograms complete.


In [41]:
```

## Sinogram Reconstruction

- In Spyder, open *ASTRA_recon_batch.py*
- Set the path to the sinograms, *path*
- Define your reconstruction settings
    - If possible, use a CUDA enabled algorithm as the GPU is much faster than the CPU
- Run the script



```python
# -*- coding: utf-8 -*-
"""
Reconstruct the *.binsino files in a folder using ASTRA

Created on Fri Aug 18 10:04:37 2017

@author: andykiss
"""


import os
import txm_image
from tomo_recon import astra_recon


# %% Define the folders
path = r'C:\Users\andykiss\Documents\recon_demo\20170330_153537_LLNL_BadTi-proces
ext = '.binsino'

# ASTRA settings
alg = 'FBP_CUDA'
alg_iter = 100
px = 1.0
flag_matlab_crop = True


# %% Get the files
if (path[-1] != '\\' and path[-1] != '/'):
    path += '/'

ls = os.listdir(path)
for fn in ls:
    if (not fn.endswith(ext)):
        ls.remove(fn)

N = len(ls)


# %% Start reconstructing
# Make output directories
outdir = 'slices/'
if (not os.path.isdir(path + outdir)):
    os.mkdir(path + outdir)
outdir = outdir + alg + '/'
if (not os.path.isdir(path + outdir)):
```

| Name | Type | Size | Value |
|---|---|---|---|
| mod_path | str | 1 | C:\Users\andykiss\Documents\program… |
| new_xy | tuple | 2 | (500.0, 0.0) |
| outdir | str | 1 | slices/SIRT_CUDA/ |
| path | str | 1 | C:\Users\andykiss\Documents\recon_d… |
| path_df | str | 1 | 20170330_153537_LLNL_BadTi-df_1/ |
| path_process | str | 1 | 20170330_153537_LLNL_BadTi-processing/ |
| path_proj | str | 1 | 20170330_153537_LLNL_BadTi_1/ |
| path_ref1 | str | 1 | 20170330_153537_LLNL_BadTi-ref_1/ |
| path_ref2 | str | 1 | 20170330_153537_LLNL_BadTi-ref_2/ |

```
Reconstrucing file (0005/2048)...done
Reconstrucing file (0006/2048)...done
Reconstrucing file (0007/2048)...done
Reconstrucing file (0008/2048)...done
Reconstrucing file (0009/2048)...done
Reconstrucing file (0010/2048)...done
Reconstrucing file (0011/2048)...done
Reconstrucing file (0012/2048)...done
Reconstrucing file (0013/2048)...done
Reconstrucing file (0014/2048)...done
Reconstrucing file (0015/2048)...done
Reconstrucing file (0016/2048)...done
Reconstrucing file (0017/2048)...done
Reconstrucing file (0018/2048)...done
Reconstrucing file (0019/2048)...done
Reconstrucing file (0020/2048)...done
Reconstrucing file (0021/2048)...done
Reconstrucing file (0022/2048)...done
Reconstrucing file (0023/2048)...done
Reconstrucing file (0024/2048)...done
```

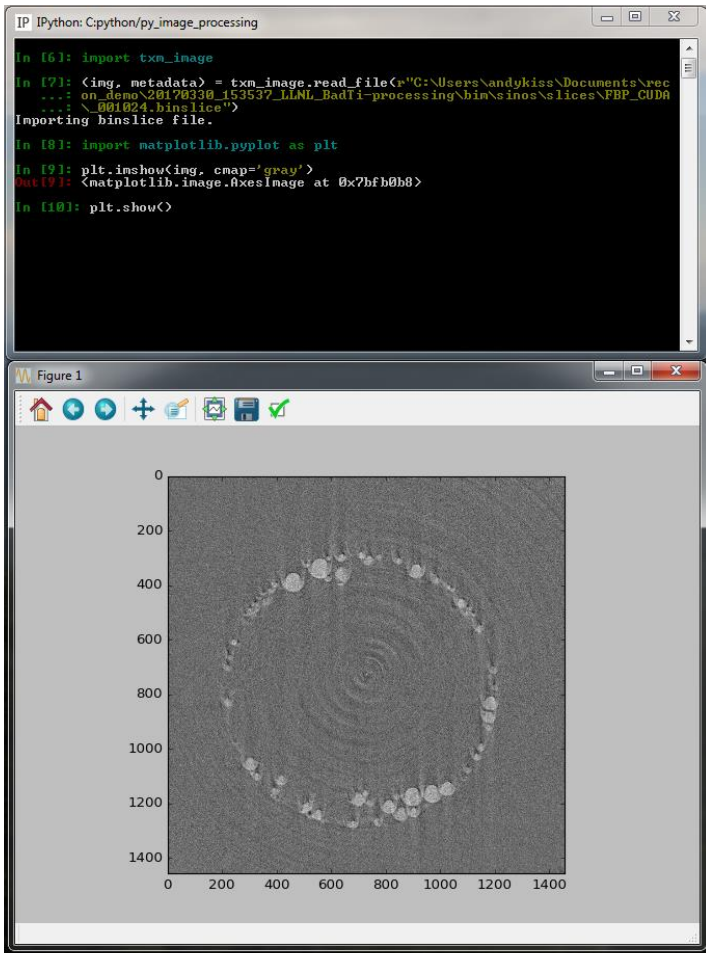
Permissions: RW   End-of-lines: CRLF   Encoding: UTF-8   Line: 24   Column: 11   Memory: 18 %

## OPENING THE RECONSTRUCTED SLICES

- The reconstructed slices can be opened using Python, Fiji, or TXM Wizard

### Python

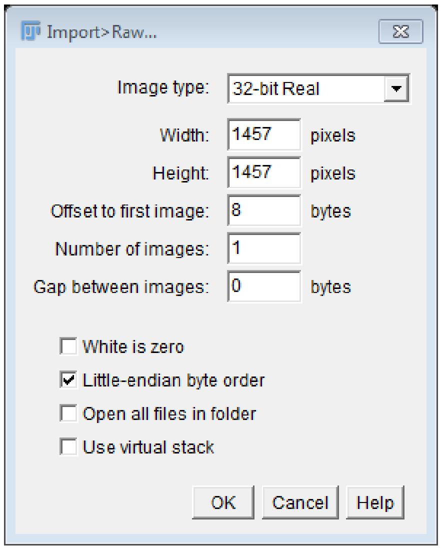
- Import the txm_image package
- Use the read_file function to load the image



```
In [6]: import txm_image

In [7]: (img, metadata) = txm_image.read_file(r"C:\Users\andykiss\Documents\rec
   ...: on_demo\20170330_153537_LLNL_BadTi-processing\bim\sinos\slices\FBP_CUDA
   ...: \_001024.binslice")
Importing binslice file.

In [8]: import matplotlib.pyplot as plt

In [9]: plt.imshow(img, cmap='gray')
Out[9]: <matplotlib.image.AxesImage at 0x7bfb0b8>

In [10]: plt.show()
```

### Fiji

- File -> Import -> Raw…
- Choose the file you would like to open using the file dialog. If you want to open all the files in the folder, still only choose one.
- Use the settings listed below, 32-bit Real, 8-bit offset, Little-endian byte order to load the data
- Type in the width and height of the image
- If you want to open all the files in the folder, select that option



Import>Raw...

Image type: 32-bit Real

Width: 1457 pixels

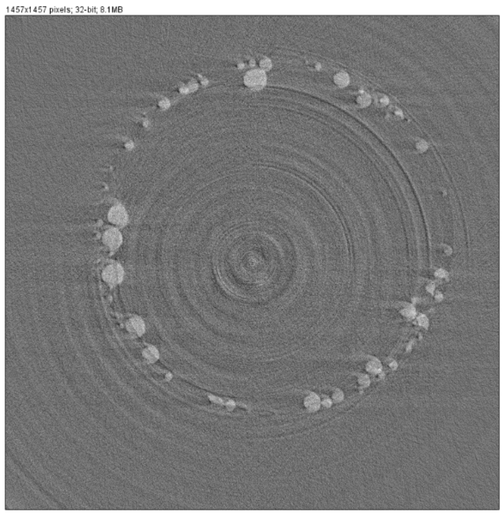
Height: 1457 pixels

Offset to first image: 8 bytes

Number of images: 1

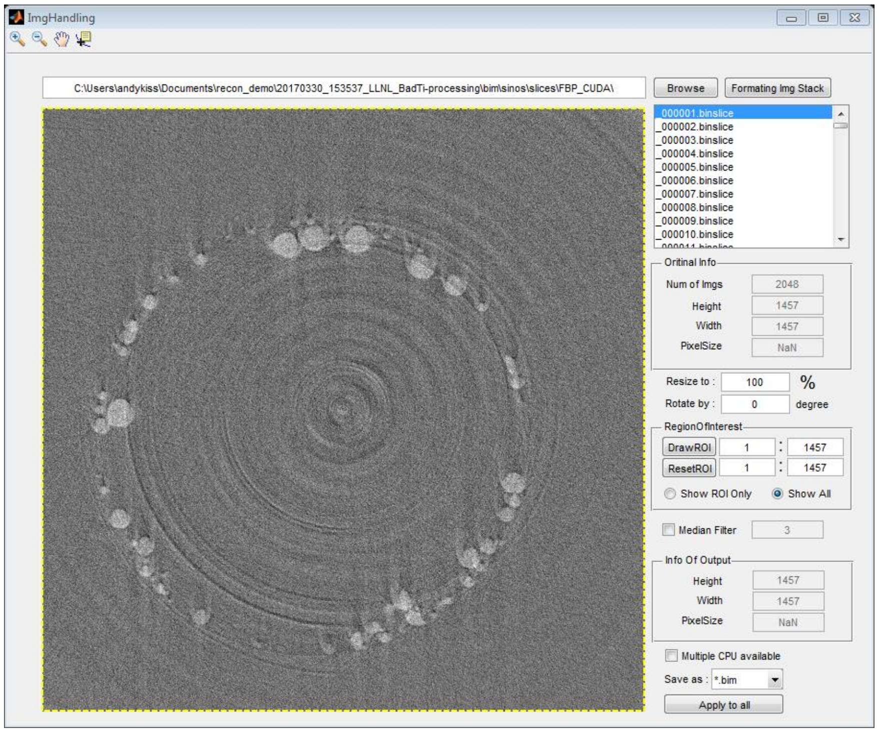
Gap between images: 0 bytes

- [ ] White is zero
- [x] Little-endian byte order
- [ ] Open all files in folder
- [ ] Use virtual stack



## TXM Wizard

- Open TXM Wizard and go to Data Evaluation -> Image Handling
- Click Browse and select the files you would like to load from the file dialog. Make sure the file filter is set to *.binslice
- Click open and view



ImgHandling

C:\Users\andykiss\Documents\recon_demo\20170330_153537_LLNL_BadTi-processing\bim\sinos\slices\FBP_CUDA\

_000001.binslice
_000002.binslice
_000003.binslice
_000004.binslice
_000005.binslice
_000006.binslice
_000007.binslice
_000008.binslice
_000009.binslice
_000010.binslice

Oritinal Info

Num of Imgs: 2048
Height: 1457
Width: 1457
PixelSize: NaN

Resize to : 100 %
Rotate by : 0 degree

RegionOfInterest

DrawROI 1 : 1457
ResetROI 1 : 1457

○ Show ROI Only ● Show All

- [ ] Median Filter 3

Info Of Output

Height: 1457
Width: 1457
PixelSize: NaN

- [ ] Multiple CPU available

Save as : *.bim