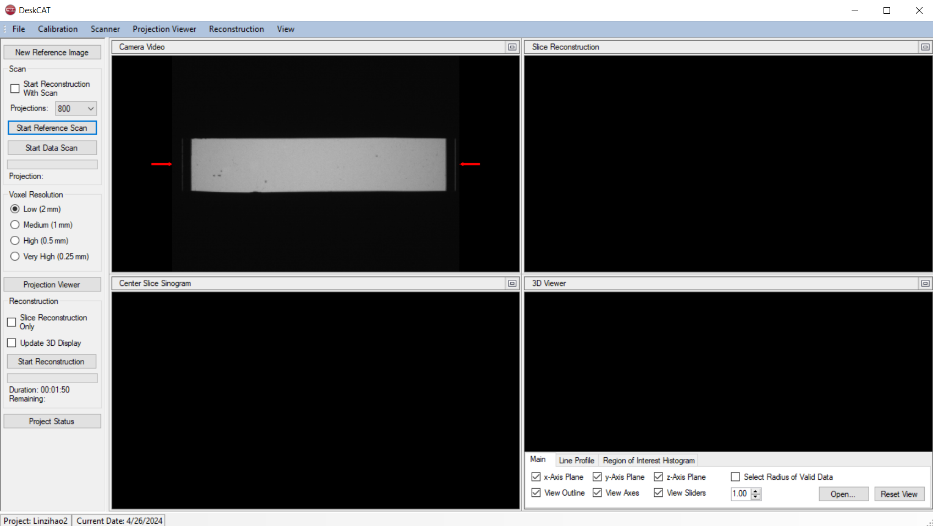
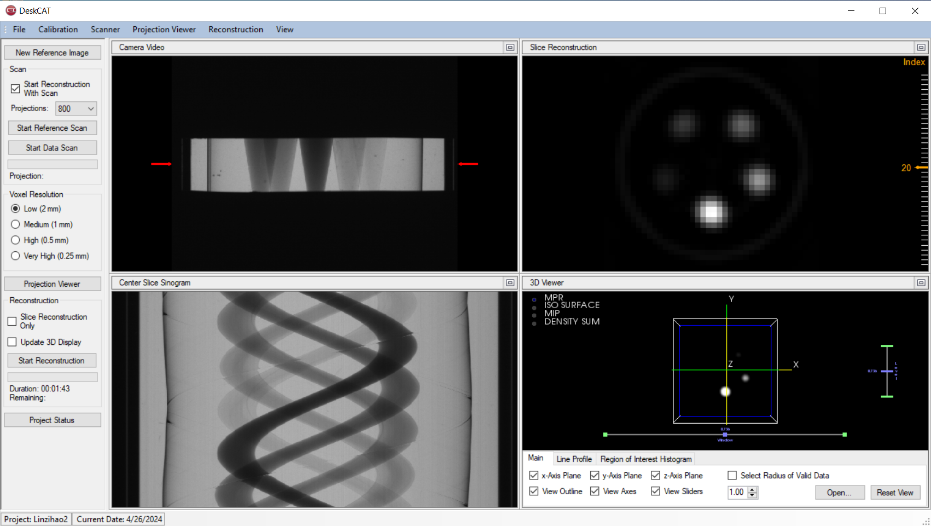
# Reconstruction

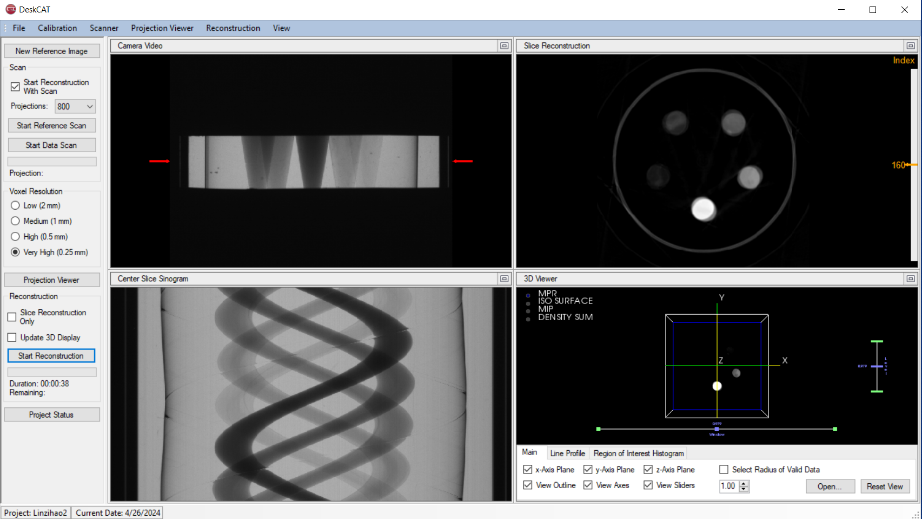
## 3.1 Dataset

3.1.1 Data acquired

During the experimental procedure, first, install a Collimator in front of the camera to capture data from the central slice and ensure the slit in the collimator remains horizontal. The camera settings were adjusted to 50% of maximum brightness to ensure evenly distributed noise. Geometry calibration was then performed using the auto-calibration function, without any phantom loaded. Subsequently, the number of projections was set on the side panel for data acquisition, and reference data was scanned without placing a phantom. And the Finger Phantom was loaded onto the scanner and a data scan was acquired using the Start Data Scan button. Reconstruction options were configured to use a Hamming filter, and the reconstruction process was initiated by selecting the voxel resolution option and pressing Start Reconstruction. The reconstructed results were then observed using the software interface.







This project focuses on studying how Contrast-to-Noise Ratio (CNR) and Signal-to-Noise Ratio (SNR) change with varying radiation doses. To achieve this, we collected three sets of projection data under three different lighting conditions. Each set consists of 800 projections, with both projection data and background reference data for each angle. The geometric parameters crucial for data calibration and processing are as follows:

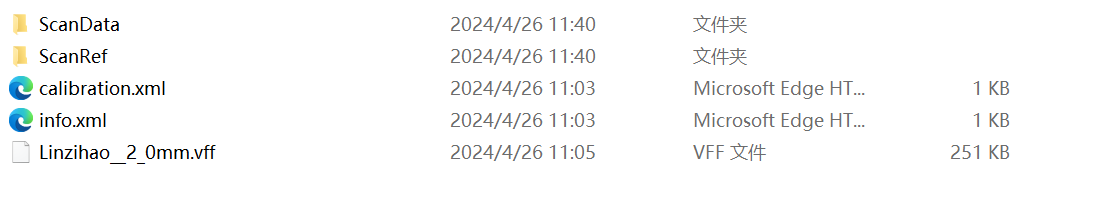
|  |  |  |
| --- | --- | --- |
| Parameter | Description | Value (cm) |
| SourceToAxis | Distance from the X-ray source to the rotation axis | 47.69 |
| AxisToDetector | Distance from the rotation axis to the detector | 4.5 |
| HorizLightSize | Horizontal size of the light beam | 9.95 |
| VertLightSize | Vertical size of the light beam | 7.46 |
| AxisOfRotationOffset | Offset of the rotation axis | 6.25 |
| EquatorialOffset | Equatorial offset | 0 |
| HorizPixelSize | Horizontal pixel size | 0.0006 |
| VertPixelSize | Vertical pixel size | 0.0006 |

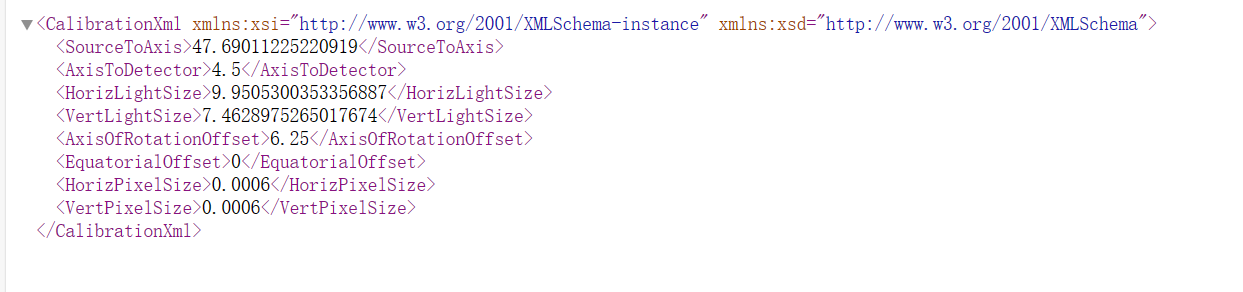
1. **class** Paramaters:
2. **def** \_\_init\_\_(self):
3. self.param = {}
4. self.param['nx'] = 640  # image width
5. self.param['ny'] = 640  # image height
6. self.param['dect\_count'] = 512  # number of detectors
7. self.param['dsd'] = 1500  # distance from source to detector
8. self.param['dso'] = 1000  # distance from source to object
9. self.param['nProj'] = 720  # number of projection views
10. self.param['startangle'] = 0  # start angle
11. self.param['endangle'] = 2\*np.pi  # end angle
12. self.param['detector\_width'] = 1  # detector spacing
13. self.param['algorithm'] = 'FBP\_CUDA'  # reconstruction algorithm SIRT
14. self.param['interation'] = -1  # interations, only used in interative reconstruction algorithms
15. self.param['short\_scan'] = False  # park compensation, only used in fanbeam and conebeam
16. self.param['noise'] = False
17. self.reuse = False

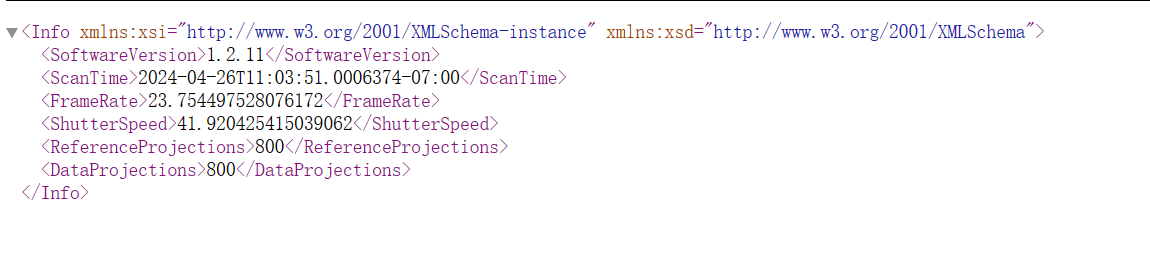
Using the provided geometric parameters, we can derive essential reconstruction parameters for accurate image reconstruction: Distance Source to Object (DSO), Distance Source to Detector (DSD) and Detector Width.

where .

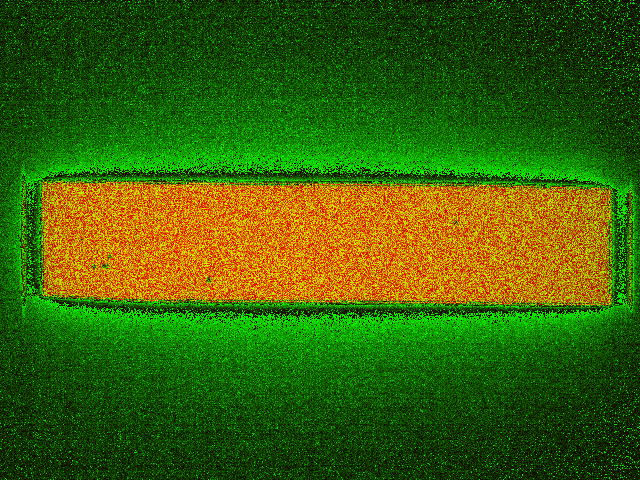
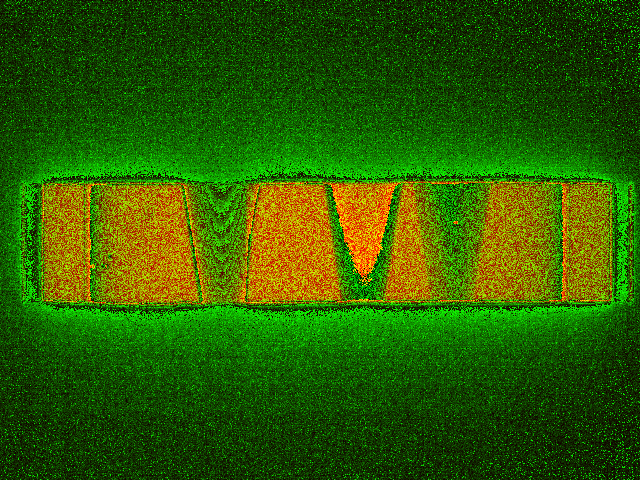
After the geometry calibration, the information of calibration and scan has been loaded in an XML file. Here are the details.







The part of the RefData and ScanData can also be shown here.



**3.1.2 Reconstruction steps**

The preprocessing part includes the following steps:

* Get fan sinogram from the projection

To obtain the sinogram, sum all the blank field projection data, get the middle data of the figure and calculate the average of the projection. Then take the middle row of each projection image sequentially, preprocess, and concatenate into a sinogram. After getting the sinogram, flat field correction should be done for preprocessing.

* Flat field correction

The blank field image is the image without the sample, while the dark field image is the image with the X-ray source turned off. Both images contain signals from the detector itself as well as background signals from other instruments. After the flat field correction, geometric correction could be used for correct detector for the final sinogram.

Corrected\_field =

* Geometric correction

The offsets provided by the system often have some deviation, requiring manual adjustment to achieve the best results.

Offset the reconstructed image 8 pixels upward and 5 pixels left

* Filtered Back Projection

In the FBP algorithm, the first step involves filtering the projection data using a specific filter kernel, commonly known the Hamming filter or ram-lak filter. The purpose of filtering is to remove noise and artefacts from the high frequency while enhancing contrast in the resulting image. Subsequently, the filtered projection data is back-projected into the image space. The back-projection process maps the filtered projection data back into the image space based on their corresponding angular positions, utilizing the reverse geometric relationships. By summing up the results of back-projection, the original structure information of the object can be reconstructed in the image space.

The FBP algorithm in the ASTRA Toolbox can be simply used by

1. **def** recon\_fan(alg, sino, source\_ori, ori\_detector, dect\_w, vol\_geom\_size,
2. angles, interations=-1, short\_scan=False, filter="ram-lak"):
3. astra.algorithm.clear()
4. vol\_geom\_fan = astra.create\_vol\_geom(vol\_geom\_size)
5. proj\_geom\_fan = astra.create\_proj\_geom('fanflat', dect\_w, sino.shape[1],
6. angles,
7. source\_ori, ori\_detector)
8. proj\_fan\_id = astra.create\_projector('cuda', proj\_geom\_fan, vol\_geom\_fan)
9. sinogram\_fan\_id = astra.data2d.create('-sino', proj\_geom\_fan, sino)
10. rec\_fan\_id = astra.data2d.create('-vol', vol\_geom\_fan)
11. cfg\_fan = astra.astra\_dict(alg)
12. cfg\_fan['ReconstructionDataId'] = rec\_fan\_id
13. cfg\_fan['ProjectionDataId'] = sinogram\_fan\_id
14. # cfg\_fan['ProjectorId'] = proj\_fan\_id
15. cfg\_fan['option'] = {'ShortScan': short\_scan}
16. **if** alg.startswith("FBP"):
17. cfg\_fan["FilterType"] = filter
18. alg\_fan\_id = astra.algorithm.create(cfg\_fan)
19. **if** interations != -1:
20. astra.algorithm.run(alg\_fan\_id, interations)
21. **else**:
22. astra.algorithm.run(alg\_fan\_id)
23. rec\_fan = astra.data2d.get(rec\_fan\_id)
24. astra.algorithm.delete(alg\_fan\_id)
25. astra.data2d.delete(rec\_fan\_id)
26. astra.data2d.delete(sinogram\_fan\_id)
27. astra.projector.delete(proj\_fan\_id)
28. # rec\_fan[rec\_fan < 0] = 0
29. # rec\_fan = np.flipud(rec\_fan)
30. astra.algorithm.clear()
31. astra.clear()
32. **return** rec\_fan

