Short Scan Using XCIST

October 7, 2025

1 Introduction

This document explains how to perform short-scan reconstruction using XCIST. XCIST supports two reconstruction modes: `fdk_equiAngle` and `helical_equiAngle`. `fdk_equiAngle` is used for axial scans, where the patient table remains stationary (table speed = 0). `helical_equiAngle` is used for helical scans, where the patient table moves continuously (table speed $\neq$ 0). The table speed can be configured in the `protocol.cfg` file, while the reconstruction type is specified in the `recon.cfg` file.

```
1   protocol.scanTypes = [1, 1, 1, 1]               # flags for airscan, offset scan, phantom scan, prep
2
3   # Table and gantry
4   protocol.scanTrajectory = "Gantry_Helical"  # name of the function that defines the scanning trajectory and model
5   protocol.viewsPerRotation = 500             # total numbers of view per rotation
6   protocol.viewCount = 500                    # total number of views in scan
7   protocol.startViewId = 0                    # index of the first view in the scan
8   protocol.stopViewId = protocol.startViewId + protocol.viewCount - 1 # index of the last view in the scan
9   protocol.airViewCount = 1                   # number of views averaged for air scan
10  protocol.offsetViewCount = 1                # number of views averaged for offset scan
11  protocol.rotationTime = 1.0                 # gantry rotation period (in seconds)
12  protocol.rotationDirection = 1              # gantry rotation direction (1=CW, -1 CCW, seen from table foot-end)
13  protocol.startAngle = 0                     # relative to vertical y-axis (n degrees)
14  protocol.tableSpeed = 0                     # speed of table translation along positive z-axis (in mm/sec)
15  protocol.startZ = 0                         # start z-position of table
16  protocol.tiltAngle = 0                      # gantry tilt angle towards negative z-axis (in degrees)
17  protocol.wobbleDistance = 0.0              # focalspot wobble distance
18  protocol.focalspotOffset = [0, 0, 0]       # focalspot position offset
19
20  # X-ray tube technique and filtration
21  protocol.mA = 500                          # tube current (in mA)
22  protocol.spectrumCallback = "Spectrum"      # name of function that reads and models the X-ray spectrum
23  protocol.spectrumFilename = "xcist_kVp120_tar7_bin1.dat" # name of the spectrum file
24  protocol.spectrumUnit_mm = 1;               # Is the spectrum file in units of photons/sec/mm^2/<current>?
25  protocol.spectrumUnit_mA = 1;               # Is the spectrum file in units of photons/sec/<area>/mA?
26  protocol.spectrumScaling = 1;               # scaling factor, works for both mono- and poly-chromatic spectra
27  protocol.bowtie = "Large.txt"               # name of the bowtie file (or [] for no bowtie)
28  protocol.filterCallback = "Xray_Filter"     # name of function to compute additional filtration
29  protocol.flatFilter = ['air', 0.001]        # additional filtration - materials and thicknesses (in mm)
30  protocol.dutyRatio = 1.0                    # tube ON time fraction (for pulsed tubes)
31  protocol.maxPrep = -1                       # set the upper limit of prep, non-positive will disable this feature
```
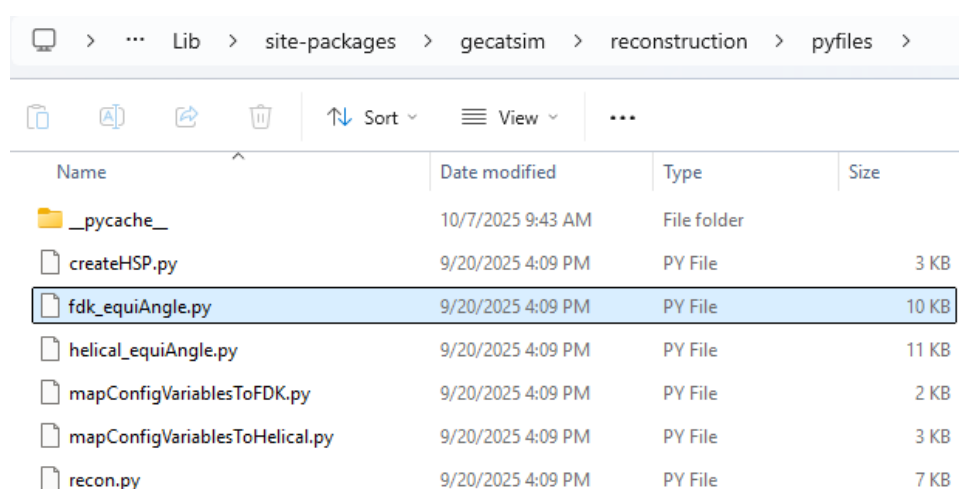
```
1   recon.fov = 500.0                      # diameter of the reconstruction field-of-view (in mm)
2   recon.imageSize = 512                  # number of columns and rows to be reconstructed (square)
3   recon.sliceCount = 1                   # number of slices to reconstruct
4
5   recon.sliceThickness = 0.579           # reconstruction slice thickness AND inter-slice interval (in mm)
6   recon.centerOffset = [0.0, 0.0, 0.0]   # reconstruction offset relative to center of rotation (in mm)
7   recon.reconType = 'fdk_equiAngle' #'helical_equiAngle'    # Name of the recon function to call
8   recon.kernelType = 'standard'          # 'R-L' for the Ramachandran-Lakshminarayanan (R-L) filter, rectangular w
9                                          # 'S-L' for the Shepp-Logan (S-L) filter, sinc window function
10                                         # 'soft', 'standard', 'bone' for kernels similar to those on clinical sca
11  recon.startAngle = 0                   # in degrees; 0 is with the X-ray source at the top
12  recon.unit = 'HU'                      # '/mm', '/cm', or 'HU'
13  recon.mu = 0.02                        # in /mm; typically around 0.02/mm
14  recon.huOffset = -1000                 # unit is HU, -1000 HU by definition but sometimes something else is pre
15  recon.printReconParameters = False     # Flag to print the recon parameters
16  recon.saveImageVolume = True           # Flag to save recon results as one big file
17  recon.saveSingleImages = False         # Flag to save recon results as individual imagesrecon.printReconParamet
18  recon.displayImagePictures = False     # Flag to display the recon results as .png images
19  recon.saveImagePictureFiles = False    # Flag to save the recon results as .png images
20  recon.displayImagePictureAxes = False  # Flag to display the axes on the .png images
21  recon.displayImagePictureTitles = False # Flag to display the titles on the .png images
22
```

2   Running Short Scan

The easiest way to do short scan is setting the value of `protocol.viewCount` to be less than `protocol.viewsPerRotation`, in `protocol.cfg` file. For example, if you set `protocol.viewsPerRotation = 100` and `protocol.viewCount = 50`, that's equivalent to "half scan".

```
1   protocol.scanTypes = [1, 1, 1, 1]              # flags for airscan, offset scan, phantom scan, prep
2
3   # Table and gantry
4   protocol.scanTrajectory = "Gantry_Helical"   # name of the function that defines the scanning trajectory and model
5   protocol.viewsPerRotation = 100                # total numbers of view per rotation
6   protocol.viewCount = 50                        # total number of views in scan
7   protocol.startViewId = 0                       # index of the first view in the scan
8   protocol.stopViewId = protocol.startViewId + protocol.viewCount - 1 # index of the last view in the scan
```

2.1   Running Short Scan - Axial Scanning

To run short scan with axial scanning mode, you need to set `recon.reconType = 'fdk_equiAngle'` in `recon.cfg`. However, the default code doesn't allow you to do short scan in axial scan mode (It will gives you index error). So we have to modify the source code `fdk_equiAngle.py` in the XCIST package.

Run `pip show gecatsim` in Anaconda Prompt:

This should give you the location of where the package is installed at. Then go to `gecatsim\reconstruction\pyfiles`. Open `fdk_equiAngle.py`.

At line 138, replace `Dg = Dgy` with `Dg = np.append(Dgy, np.zeros((YL, ZL, ProjScale - cfg.protocol.viewCount)), axis=2)`:

```
129
130        ############### pre-weighting for ramp-filter
131
132        print("* Pre-weighting the filter...")
133        for Yindex in range(YL):
134            for zindex in range(ZL):
135                Dgy[Yindex, zindex, :] = (DistD / np.sqrt(DistD ** 2 + ((zindex - ZCtr) * DeltaZ) ** 2)) * ProjData[Yindex,
136                                zindex,:] * math.cos((Yindex - YCtr) * DeltaUW)
137
138        #Dg=Dgy
139        Dg = np.append(Dgy, np.zeros((YL, ZL, ProjScale - cfg.protocol.viewCount)), axis=2) # try padding (cut unwantted section from the
140    scanned data)
141
```

What this does is it pads the missing section from a short scan (pretending it's a full scan but fill the missing section with 0), and then reconstructs using pre-defined view counts.

Running the main script with `protocol.viewsPerRotation = 100` and `protocol.viewCount = 50`. You should get the result shown below:
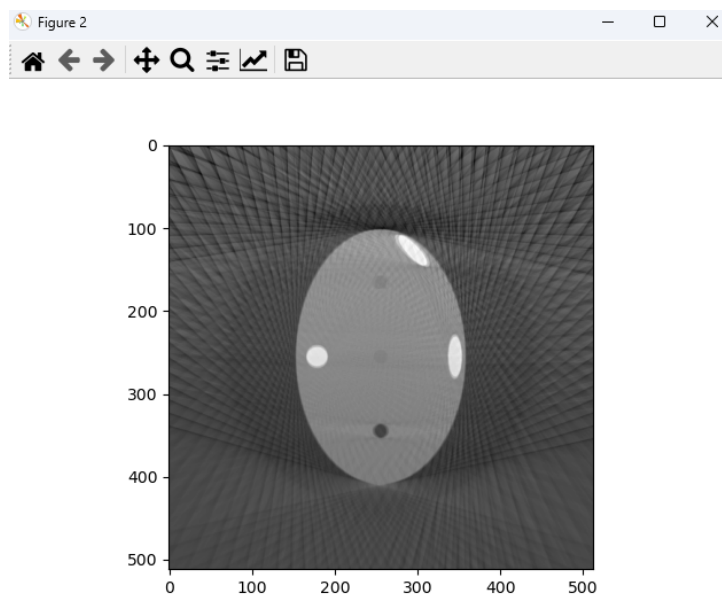
2.2 Running Short Scan - Helical Scanning

Running Short Scan with Helical Scanning mode is much easier. Simply set `recon.reconType = 'helical_equiAngle'` in `recon.cfg`. You don't need to modify anything, but the table speed must be non-zero otherwise the recon result will be pure dark image.