

Tutorial: Installing and Using XCIST with Anaconda

Botao Zhao

October 6, 2025

Contents

| | | |
|----------|---|----------|
| 1 | Introduction | 2 |
| 2 | Installing Anaconda | 2 |
| 2.1 | Download Anaconda | 2 |
| 2.2 | Installation Steps | 2 |
| 3 | Setting up XCIST Environment | 2 |
| 3.1 | Install CatSim Python package | 2 |
| 4 | Running XCIST | 3 |
| 4.1 | Create .cfg Files | 3 |
| 4.1.1 | Create Digital Phantom (saved as Phantom_example.ppm): | 3 |
| 4.1.2 | Create Phantom Config. File (saved as example_phantom.cfg): | 4 |
| 4.1.3 | Create Protocol Config. File (saved as example_protocol.cfg): | 4 |
| 4.1.4 | Create Scanner Config. File (saved as example_scanner.cfg): | 4 |
| 4.1.5 | Create Physics Config. File (Optional, saved as example_physics.cfg): . . . | 5 |
| 4.1.6 | Create Recon Config. File (saved as example_recon.cfg): | 6 |
| 4.2 | Import XCIST-CatSim in Python | 6 |
| 4.3 | Initialization | 6 |
| 4.4 | Run a simple reconstruction | 6 |
| 4.5 | Result Visualization | 7 |
| 4.6 | Output Result | 8 |

1 Introduction

XCIST (X-ray Computed Imaging Simulation Toolkit) is an open-source software package for simulating CT imaging systems. This tutorial provides step-by-step instructions to install XCIST using Anaconda, configure your environment, and run a basic reconstruction example.

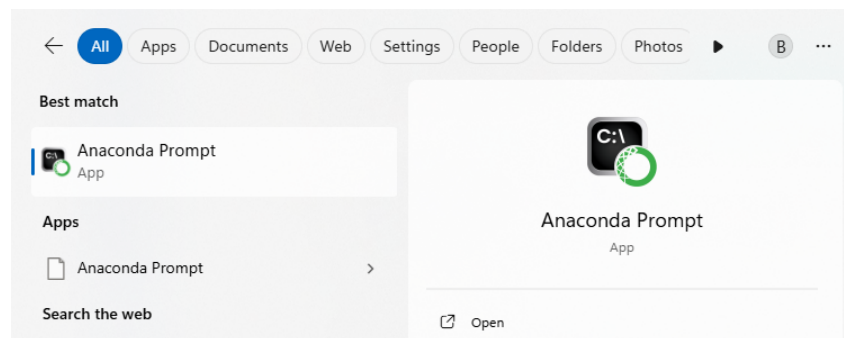
2 Installing Anaconda

2.1 Download Anaconda

1. Go to the official Anaconda website: <https://www.anaconda.com/download>
2. Download the installer for your operating system (Windows, macOS, Linux).

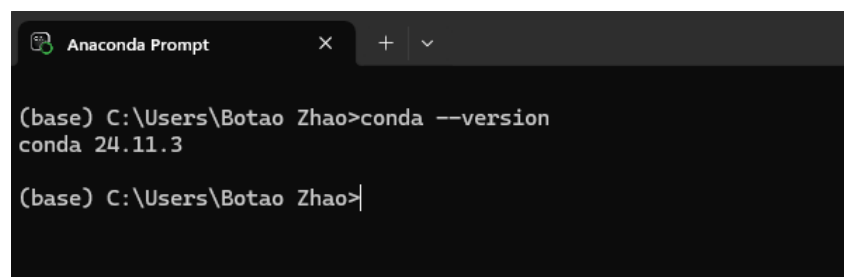
2.2 Installation Steps

- Run the installer and follow the on-screen instructions. Then you should be able to launch Anaconda Prompt from the Windows Start menu:



- Verify installation:

```
conda --version
```



3 Setting up XCIST Environment

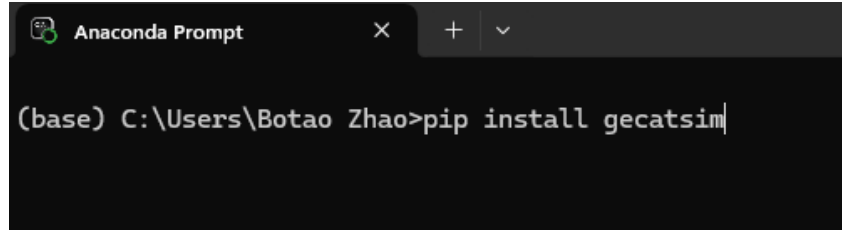
3.1 Install CatSim Python package

It is recommended to follow XCIST installation guide: <https://github.com/xcist/documentation/wiki>

In your Anaconda Prompt, run:

```
pip install gecatsim
```

Now you should be able to run XCIST on your computer.



```
Anaconda Prompt
(base) C:\Users\Botao Zhao>pip install gecatsim
```

4 Running XCIIST

4.1 Create .cfg Files

To run the simulation, the main script must load all required config files. Here are some examples of .cfg files:

4.1.1 Create Digital Phantom (saved as Phantom_example.ppm):

```
materialList = {'pmma' 'water' 'air' 'polyethylene' 'ICRU_skeleton_cortical_bone_adult' };

object.center(1,:) = [0.000000 0.000000 0.000000];
object.half_axes(1,:) = [130.000000 90.000000 75.000000];
object.euler_angs(1,:) = [0.000000 0.000000 0.000000];
object.density(1) = 1.000000;
object.type(1) = 2;
object.material(1) = 1;
object.axial_lims(1,:) = [0 0];
object.shape(1) = 0;
object.clip{1} = [];

object.center(2,:) = [0.000000 0.000000 0.000000];
object.half_axes(2,:) = [6.500000 6.500000 75.000000];
object.euler_angs(2,:) = [0.000000 0.000000 0.000000];
object.density(2) = 1.000000;
object.type(2) = 2;
object.material(2) = 2;
object.axial_lims(2,:) = [0 0];
object.shape(2) = 0;
object.clip{2} = [];

object.center(3,:) = [0.000000 70.000000 0.000000];
object.half_axes(3,:) = [20 6.500000 75.000000];
object.euler_angs(3,:) = [0.000000 0.000000 0.000000];
object.density(3) = 1.000000;
object.type(3) = 2;
object.material(3) = 5;
object.axial_lims(3,:) = [0 0];
object.shape(3) = 0;
object.clip{3} = [];

object.center(4,:) = [70.000000 0.000000 4.000000];
object.half_axes(4,:) = [6.500000 6.500000 75.000000];
object.euler_angs(4,:) = [0.000000 0.000000 0.000000];
object.density(4) = 1.000000;
object.type(4) = 2;
object.material(4) = 3;
object.axial_lims(4,:) = [0 0];
object.shape(4) = 0;
object.clip{4} = [];

object.center(5,:) = [-70.000000 0.000000 0.000000];
object.half_axes(5,:) = [6.500000 6.500000 75.000000];
object.euler_angs(5,:) = [0.000000 0.000000 0.000000];
object.density(5) = 1.000000;
object.type(5) = 2;
object.material(5) = 4;
object.axial_lims(5,:) = [0 0];
object.shape(5) = 0;
object.clip{5} = [];
```

```

object.center(6,:) = [0.000000 -60.000000 -4.000000];
object.half_axes(6,:) = [10 10 75.000000];
object.euler_angs(6,:) = [0.000000 0.000000 0.000000];
object.density(6) = 1.000000;
object.type(6) = 2;
object.material(6) = 5;
object.axial_lims(6,:) = [0 0];
object.shape(6) = 0;
object.clip{6} = [];

object.center(7,:) = [-100.000000 30.000000 0.000000];
object.half_axes(7,:) = [20 6.500000 75.000000];
object.euler_angs(7,:) = [0.000000 0.000000 45.000000];
object.density(7) = 1.000000;
object.type(7) = 2;
object.material(7) = 5;
object.axial_lims(7,:) = [0 0];
object.shape(7) = 0;
object.clip{7} = [];

```

4.1.2 Create Phantom Config. File (saved as example_phantom.cfg):

```

# Phantom
# name of function that reads and models phantom
phantom.callback = "Phantom_Analytic"
# name of function that performs projection through phantom
phantom.projectorCallback = "C_Projector_Analytic"
# phantom filename
phantom.filename = 'Phantom_example.ppm'
# offset of phantom center relative to origin (in mm)
phantom.centerOffset = [0.0, 0.0, 0.0]
# re-scale the size of phantom
phantom.scale = 1.0

```

4.1.3 Create Protocol Config. File (saved as example_protocol.cfg):

```

protocol.scanTypes = [1, 1, 1, 1] # flags for airscan, offset scan, phantom scan, prep

# Table and gantry
protocol.scanTrajectory = "Gantry_Helical" # name of the function that defines the scanning trajectory and model
protocol.viewsPerRotation = 200 # total numbers of view per rotation
protocol.viewCount = 200 # total number of views in scan
protocol.startViewId = 0 # index of the first view in the scan
protocol.stopViewId = protocol.startViewId + protocol.viewCount - 1 # index of the last view in the scan
protocol.airViewCount = 1 # number of views averaged for air scan
protocol.offsetViewCount = 1 # number of views averaged for offset scan
protocol.rotationTime = 1.0 # gantry rotation period (in seconds)
protocol.rotationDirection = 1 # gantry rotation direction (1=CCW, -1 CCW, seen from table foot-end)
protocol.startAngle = 0 # relative to vertical y-axis (n degrees)
protocol.tableSpeed = 2 # speed of table translation along positive z-axis (in mm/sec)
protocol.startZ = -1 # start z-position of table
protocol.tiltAngle = 0 # gantry tilt angle towards negative z-axis (in degrees)
protocol.wobbleDistance = 0.0 # focalspot wobble distance
protocol.focalspotOffset = [0, 0, 0] # focalspot position offset

# X-ray tube technique and filtration
protocol.mA = 500 # tube current (in mA)
protocol.spectrumCallback = "Spectrum" # name of function that reads and models the X-ray spectrum
protocol.spectrumFilename = "xcist_kVp120_tar7_bin1.dat" # name of the spectrum file
protocol.spectrumUnit_mm = 1; # Is the spectrum file in units of photons/sec/mm^2/<current>?
protocol.spectrumUnit_mA = 1; # Is the spectrum file in units of photons/sec/<area>/mA?
protocol.spectrumScaling = 1; # scaling factor, works for both mono- and poly-chromatic spectra
protocol.bowtie = "large.txt" # name of the bowtie file (or [] for no bowtie)
protocol.filterCallback = "Xray_Filter" # name of function to compute additional filtration
protocol.flatFilter = ['air', 0.001] # additional filtration - materials and thicknesses (in mm)
protocol.dutyRatio = 1.0 # tube ON time fraction (for pulsed tubes)
protocol.maxPrep = -1 # set the upper limit of prep, non-positive will disable this feature

```

4.1.4 Create Scanner Config. File (saved as example_scanner.cfg):

```

recon.fov = 400.0 # diameter of the reconstruction field-of-view (in mm)
recon.imageSize = 512 # number of columns and rows to be reconstructed (square)

```

```

recon.sliceCount = 1                                # number of slices to reconstruct

recon.sliceThickness = 0.579                        # reconstruction slice thickness AND inter-slice interval (in mm)
recon.centerOffset = [0.0, 0.0, 0.0]               # reconstruction offset relative to center of rotation (in mm)
recon.reconType = 'helical_equiAngle'               # Name of the recon function to call
recon.kernelType = 'standard'                      # 'R-L' for the Ramachandran-Lakshminarayanan (R-L) filter, rectangular window function
                                                    # 'S-L' for the Shepp-Logan (S-L) filter, sinc window function
                                                    # 'soft', 'standard', 'bone' for kernels similar to those on clinical scanners
                                                    # in degrees; 0 is with the X-ray source at the top
                                                    # '/mm', '/cm', or 'HU'
                                                    # in /mm; typically around 0.02/mm
                                                    # unit is HU, -1000 HU by definition but sometimes something else is preferable
recon.startAngle = 0                                # Flag to print the recon parameters
recon.unit = 'HU'                                    # Flag to save recon results as one big file
recon.mu = 0.02                                     # Flag to save recon results as individual images
recon.huOffset = -1000                              # Flag to display the recon results as .png images
recon.printReconParameters = False                 # Flag to save the recon results as .png images
recon.saveImageVolume = True                       # Flag to display the axes on the .png images
recon.saveSingleImages = False                     # Flag to display the titles on the .png images
recon.displayImagePictures = False
recon.saveImagePictureFiles = False
recon.displayImagePictureAxes = False
recon.displayImagePictureTitles = False

```

4.1.5 Create Physics Config. File (Optional, saved as example_physics.cfg):

```

# Geometric and energy sampling
physics.energyCount = 12                            # number of energy bins
physics.monochromatic = -1                          # -1 for polychromatic (see protocol.cfg);
                                                    # for monoenergetic specify the energy in keV

physics.colSampleCount = 1                          # number of samples of detector cells in lateral direction
physics.rowSampleCount = 1                          # number of samples of detector cells in longitudinal direction
physics.srcXSampleCount = 2                        # number of samples of focal spot in lateral direction
physics.srcYSampleCount = 2                        # number of samples of focal spot cells in longitudinal direction
physics.viewSampleCount = 2                        # number of samples of each view angle range in rotational direction

# Flags to determine what has to be recalculated each view
physics.recalcDet = 0                               # recalculate detector geometry
physics.recalcSrc = 0                               # recalculate source geometry and relative intensity
physics.recalcRayAngle = 0                         # recalculate source-to-detector-cell ray angles
physics.recalcSpec = 0                             # recalculate spectrum
physics.recalcFilt = 0                             # recalculate filters
physics.recalcFlux = 0                             # recalculate flux
physics.recalcPht = 0                             # recalculate phantom

# Noise on/off settings
physics.enableQuantumNoise = 1                     # enable quantum noise
physics.enableElectronicNoise = 1                  # enable electronic noise

# Internal physics models
physics.rayAngleCallback = "Detector_RayAngles_2D" # name of function to calculate source-to-detector-cell ray
physics.fluxCallback = "Detection_Flux"            # name of function to calculate flux
physics.scatterCallback = "Scatter_ConvolutionModel" # name of function to calculate scatter
physics.scatterKernelCallback = ""                 # name of function to calculate scatter kernel (" " for default kernel)
physics.scatterScaleFactor = 55                    # scale factor, 1 appropriate for 64-mm detector and 20-cm water
physics.callback_pre_log = "Scatter_Correction"
physics.scatterCorrectionScaleFactor = 24
physics.prefilterCallback = "Detection_prefilter" # name of function to calculate detection pre-filter
physics.crosstalkCallback = "CalcCrossTalk"        # name of function to calculate X-ray crosstalk in the
physics.col_crosstalk = 0.025
physics.row_crosstalk = 0.02
physics.opticalCrosstalkCallback = "CalcOptCrossTalk" # name of function to calculate X-ray crosstalk
physics.col_crosstalk_opt = 0.04
physics.row_crosstalk_opt = 0.045
physics.lagCallback = ""                          # name of function to calculate detector lag
physics.opticalCrosstalkCallback = ""              # name of function to calculate optical crosstalk in the detector
physics.DASCallback = "Detection_DAS"             # name of function to calculate the detection process

# I/O preferences
physics.outputCallback = "WriteRawView"           # name of function to produce the simulation output

```

```
# BHC
physics.callback_post_log = 'Prep_BHC_Accurate'
physics.EffectiveMu = 0.2
physics.BHC_poly_order = 5
physics.BHC_max_length_mm = 300
physics.BHC_length_step_mm = 10
```

4.1.6 Create Recon Config. File (saved as example_recon.cfg):

```
recon.fov = 400.0 # diameter of the reconstruction field-of-view (in mm)
recon.imageSize = 512 # number of columns and rows to be reconstructed (square)
recon.sliceCount = 1 # number of slices to reconstruct

recon.sliceThickness = 0.579 # reconstruction slice thickness AND inter-slice interval (in mm)
recon.centerOffset = [0.0, 0.0, 0.0] # reconstruction offset relative to center of rotation (in mm)
recon.reconType = 'helical_equiAngle' # Name of the recon function to call
recon.kernelType = 'standard' # 'R-L' for the Ramachandran-Lakshminarayanan (R-L) filter, rectangular
# 'S-L' for the Shepp-Logan (S-L) filter, sinc window function
# 'soft', 'standard', 'bone' for kernels similar to those on clinical s
# in degrees; 0 is with the X-ray source at the top
# '/mm', '/cm', or 'HU'
# in /mm; typically around 0.02/mm
# unit is HU, -1000 HU by definition but sometimes something else is pr
# Flag to print the recon parameters
recon.startAngle = 0
recon.unit = 'HU'
recon.mu = 0.02
recon.huOffset = -1000
recon.printReconParameters = False # Flag to print the recon parameters
recon.saveImageVolume = True # Flag to save recon results as one big file
recon.saveSingleImages = False # Flag to save recon results as individual imagesrecon.printReconParame
recon.displayImagePictures = False # Flag to display the recon results as .png images
recon.saveImagePictureFiles = False # Flag to save the recon results as .png images
recon.displayImagePictureAxes = False # Flag to display the axes on the .png images
recon.displayImagePictureTitles = False # Flag to display the titles on the .png images
```

4.2 Import XCIST-CatSim in Python

Open Python or Jupyter Notebook and run:

```
import numpy as np
import gecatsim as xc
import gecatsim.reconstruction.pyfiles.recon as recon
```

4.3 Initialization

Load config files ("example_physics" is optional):

```
ct = xc.CatSim(
    "example_phantom",
    "example_protocol",
    "example_scanner",
    "example_physics",
    "example_recon")
```

4.4 Run a simple reconstruction

Here is an example of running simulation and reconstructing:

```
##----- Run simulation
ct.run_all() # run the scans defined by protocol.scanTypes

##----- Reconstruction
cfg = ct.get_current_cfg();
cfg.do_Recon = 1
cfg.waitForKeypress = 0
recon.recon(cfg)
```

4.5 Result Visualization

Here is an example of plotting sinogram and reconstruction images:

```
##----- Show results
import matplotlib.pyplot as plt

# Plot Sinogram
sino = xc.rawread(
    ct.resultsName+'.prep',
    [ct.protocol.viewCount, ct.scanner.detectorRowCount, ct.scanner.detectorColCount],
    'float')

sino = np.squeeze(sino[:, 0, :])

xc.rawwrite(
    ct.resultsName+"_sino_%dx%d.raw"%(
    ct.scanner.detectorColCount,
    ct.protocol.viewCount),
    sino.copy(order="C"))

sinoname = ct.resultsName+"_sino_%dx%d.raw"%(
    ct.scanner.detectorColCount,
    ct.protocol.viewCount)

sinogram = xc.rawread(
    sinoname,
    [ct.protocol.viewCount, ct.scanner.detectorColCount],
    'float')

plt.figure(1)
plt.imshow(sinogram, cmap='gray')
plt.show()

# Plot Recon Result
imgFname = "%s_%dx%d.raw" %(
    ct.resultsName,
    ct.recon.imageSize,
    ct.recon.imageSize,
    ct.recon.sliceCount)

img = xc.rawread(
    imgFname,
    [ct.recon.sliceCount, ct.recon.imageSize, ct.recon.imageSize],
    'float')

plt.figure(2)
plt.imshow(img[ct.recon.sliceCount//2, :, :], cmap='gray')
plt.show()
```

4.6 Output Result

Running the above scripts should give you something like:

```
In [1]: runfile('F:/My Drive/XCIST/example_NXR/example_main.py', wdir='F:/My Drive/XCIST/
example_NXR')
Airscan
Offset scan
Phantom scan
 0%|          | 0/200 [00:00<?, ?it/s]Starting to read ANALYTIC phantom...
Reading phantom file Phantom_example.ppm...
Reading phantom file Phantom_example.ppm.
Converting boxes to cylinders with clipping.
Scaling and compacting format.
Setting 7 objects in the ANALYTIC phantom.
Converting tori in the ANALYTIC phantom.
Bounding objects in the ANALYTIC phantom.
... done with phantom.
... done reading phantom.
60%|██████    | 120/200 [00:35<00:24, 3.27it/s]
```

The sinogram and recon plots:

