JOURNAL OF SYNCHROTRON RADIATION

ISSN 1600-5775

Received 27 September 2016
Accepted 6 February 2017

Edited by P. A. Pianetta, SLAC National Accelerator Laboratory, USA

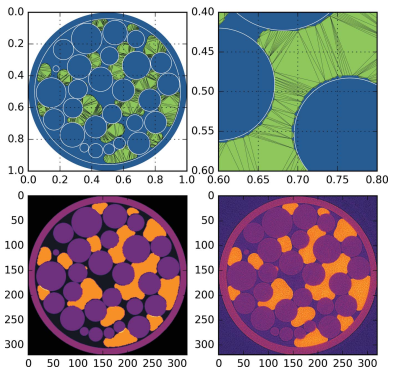
Keywords: tomography; data acquisition; simulation; phantom; scanning X-ray probe; reconstruction; quality; open-source; Python; experiment design.

XDesign: an open-source software package for designing X-ray imaging phantoms and experiments

Daniel J. Ching[a] and Doğa Gürsoy[b]*

[a]Oregon State University, Corvallis, OR 97330, USA, and [b]Advanced Photon Source, Argonne National Laboratory, Lemont, IL 60439, USA. *Correspondence e-mail: dgursoy@aps.anl.gov

The development of new methods or utilization of current X-ray computed tomography methods is impeded by the substantial amount of expertise required to design an X-ray computed tomography experiment from beginning to end. In an attempt to make material models, data acquisition schemes and reconstruction algorithms more accessible to researchers lacking expertise in some of these areas, a software package is described here which can generate complex simulated phantoms and quantitatively evaluate new or existing data acquisition schemes and image reconstruction algorithms for targeted applications.



1. Introduction

Historically, X-ray imaging techniques have been developed by and for the medical imaging community and then adapted for other uses. However, this may be the cause of three common problems in the synchrotron X-ray computed tomography (XCT) community: (i) the Shepp–Logan phantom (Shepp & Logan, 1974) is still used as a standard phantom, but it does not represent the materials of a diverse synchrotron research community, (ii) trying alternative acquisition schemes and experimental setups is difficult, especially for scanning probes, and (iii) researchers are not quantitatively evaluating alternate reconstruction methods. These problems may still exist, in part, because developing the solution for each requires developing solutions for the other two. However, since no one person is an expert materials scientist, physicist and mathematician, these solutions have not been developed.

In order to bridge the gap between materials scientists, physicists and mathematicians, we have created a modular software toolbox/framework (Fig. 1) written in Python to help users and developers of synchrotron-based tomography to easily develop, validate and share XCT experimental methods. With *XDesign*, materials scientists can choose experimental methods based on phantoms they have created to resemble their actual materials of interest, physicists can optimize data acquisition methods using quantitative quality measures, and mathematicians can test their numerical algorithms on more diverse geometries and flexible input data.

This publication is organized as follows. §2 describes custom phantom generation capability; §3 describes data acquisition simulation; §4 describes reconstruction methods; §5 describes the quality metrics implemented in our toolbox; §6 describes features that are not implemented in the initial release, but are planned pending community interest. Source code, documentation and information on how to contribute are freely

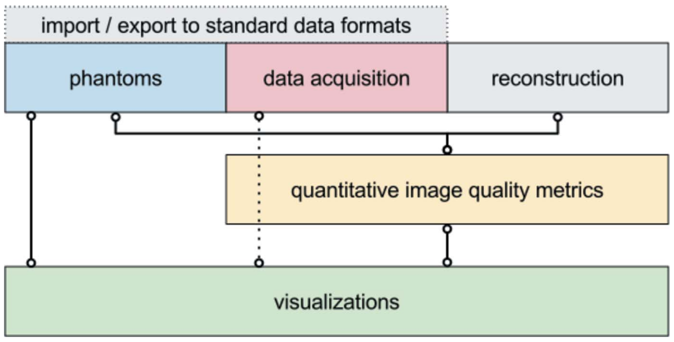

import / export to standard data formats
phantoms
data acquisition
reconstruction
quantitative image quality metrics
visualizations

Figure 1
Modular schematic of *XDesign*. Implemented modules are drawn in solid boxes and proposed modules are drawn in dotted lines.

available through GitHub at tomography/xdesign. All graphics are rendered using *Matplotlib* (Hunter, 2007).

1.1. Why custom phantoms?

For many reconstruction studies, the simulated phantom of choice is the Shepp–Logan phantom, which is a piecewise constant model of a cross section of a human head, but for a majority of the materials community this phantom does not represent the materials studied. For many of the same reasons that one acquisition setup does not fit all experiments, Shepp–Logan does not fit all simulations.

Using physical phantoms for quantitative procedural evaluation is not good practice because the exact dimensions or composition of the object are not known; this precision is important because researchers are already trying to resolve features and estimate quantities of interest at the limits of the tomographic instrument resolution and sensitivity.

Creating custom simulated phantoms is beneficial because it allows coupling of theoretical models with actual tomography. In fact, there are some reconstruction methods which utilize an internal model material model as a key part of the reconstruction algorithm (Zanaga *et al.*, 2016).

1.2. Related works

There are open-source software tools for simulating data acquisition of non-X-ray systems: *GATE* (Jan *et al.*, 2004), *STIR* (Thielemans *et al.*, 2006) and *k-Wave* (Treeby & Cox, 2010); and there are open-source tools that focus on different reconstruction methods: *TXM* wizard (Liu *et al.*, 2012), *MMX-I* (Bergamaschi *et al.*, 2016), *ASTRA* (Palenstijn *et al.*, 2013) and *TomoPy* (Gürsoy *et al.*, 2014). However, most of these tools are not set up for custom data acquisition schemes for simulating streaming reconstructions. Some support different detection geometries such as cone- and parallel-beam geometries, but none support generic geometries for scanning probes. Materials properties which change over space-time can affect optimal data acquisition methods (Hsieh *et al.*, 2006; Holman *et al.*, 2016), and uniform spatio-temporal sampling is becoming undesirable as data sets become larger. Researchers are already developing streaming reconstruction systems with feedback to acquisition systems (Marchesini *et al.*, 2016; Vogelgesang *et al.*, 2016) because one acquisition setup may not fit the needs of all experiments or even a single experiment.

2. Phantom generation

In *XDesign*, each phantom is a collection of multiple features. Features are represented by a geometry and some property functions which are valid within that geometry. A geometry is any sub-region of the phantom's ambient space. A property is what a probe measures in that geometry. It is best to think of a phantom as a piecewise property function in an N-dimensional (ND) space, because, in that way, features' properties such as density, attenuation, position and shape can be represented continuously in space-time.

2.1. Building a phantom

In the first release, *XDesign* supports two-dimensional geometries including circles, triangles and triangular meshes. It is possible to easily construct a phantom from the ground up by assembling geometry objects into features and then assigning them properties. The code below generates the phantom in Fig. 2 and the structural hierarchy in Fig. 3.

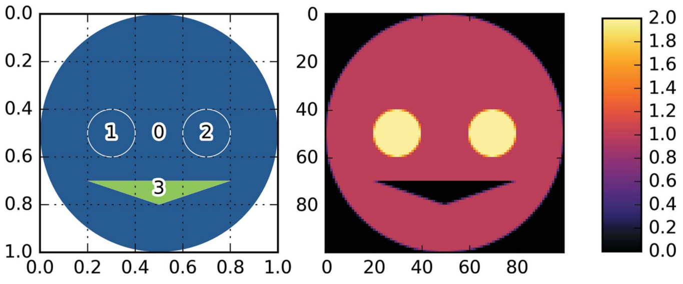


Figure 2
Geometry (left) and attenuation property (right) of a simple phantom described in Fig. 3 and §2.1.

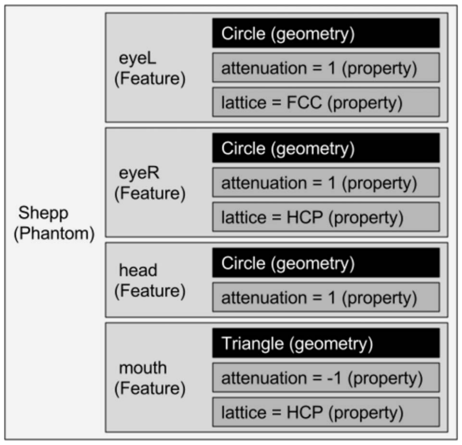


Figure 3
Example phantom data structure for the phantom described in §2.1 and Fig. 2.

```
# Create head
head = Feature(Circle(Point([0.5, 0.5]),
radius = 0.5))
head.mass_atten = 1

# Create left eye
eyeL = Feature(Circle(Point([0.3, 0.5]),
radius = 0.1))
eyeL.mass_atten = 1

# Create right eye
eyeR = Feature(Circle(Point([0.7, 0.5]),
radius = 0.1))
eyeR.mass_atten = 1

# Create mouth
mouth = Feature(Triangle(Point([0.2, 0.7]),
Point([0.5, 0.8]), Point([0.8, 0.7])))
mouth.mass_atten = -1

# Assembly features to have a face phantom
face = Phantom()
face.append([head, eyeL, eyeR, mouth])
```

2.2. Phantom parameterization

You can parameterize phantom construction easily by defining a superclass of the Phantom class. For example, Fig. 4 shows various outputs from the parameterized function below. This class randomly generates a foam-like phantom using void size range, gap and target porosity as parameters.

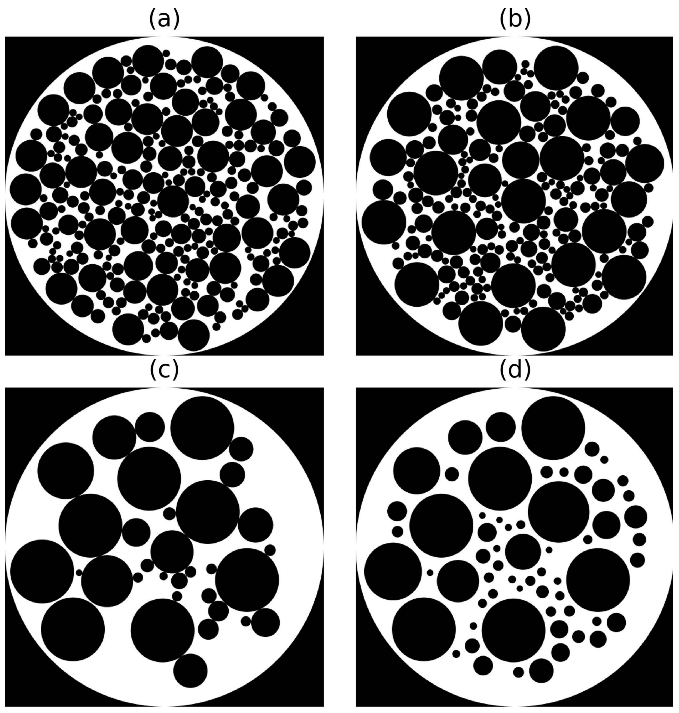


Figure 4
Four different foam-like phantoms generated from the parameterized function in §2.2. (*a*) Size_range = [0.05, 0.01], gap = 0, porosity = 1; (*b*) size_range = [0.07, 0.01], gap = 0, porosity = 0.75; (*c*) size_range = [0.1, 0.01], gap = 0, porosity = 0.5; (*d*) size_range = [0.1, 0.01], gap = 0.015, porosity = 1. Foams based on the appearance of tomography were collected by Patterson *et al.* (2016).

```
# Create a foam-like phantom from application
  specific parameters
foam_like_phantom = Foam(size_range, gap, porosity)
```

More parameterized phantoms are shown in Fig. 5.

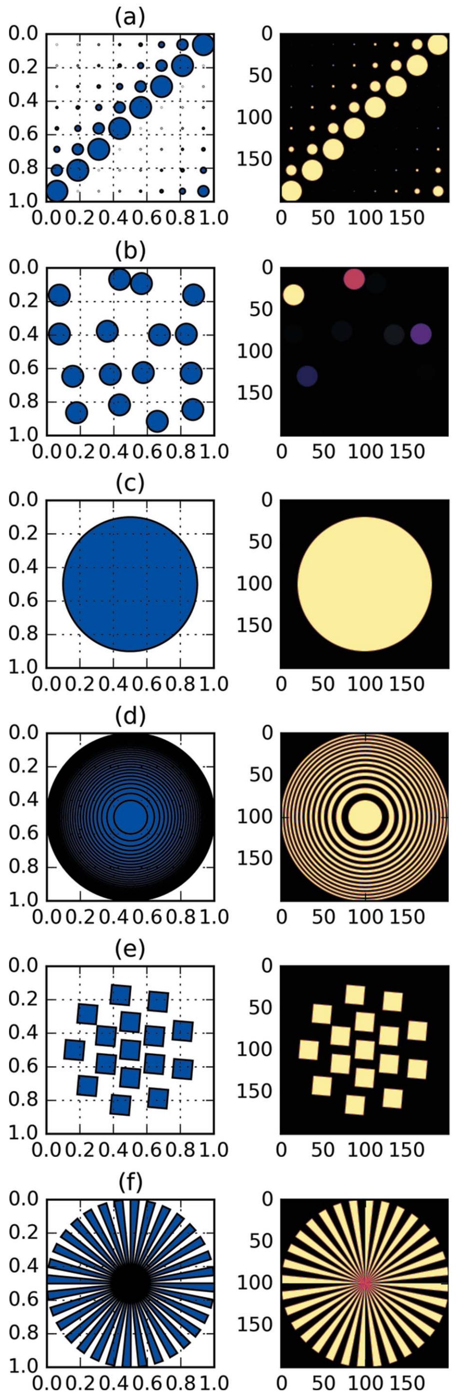


Figure 5
Other parameterized phantoms: (*a*) latin square of different sizes, (*b*) random circles of varying levels of attenuation, (*c*) a unit circle, (*d*) lines of increasingly smaller width, (*e*) slanted squares, (*f*) Siemens star. Phantoms (*a*), (*b*), (*e*) could be used for studying the effects of reconstruction on objects of different sizes and attenuation. Phantom (*c*) could be used for noise reduction studies. Phantoms (*d*) and (*f*) could all be used to calculate the modulation transfer function (MTF).

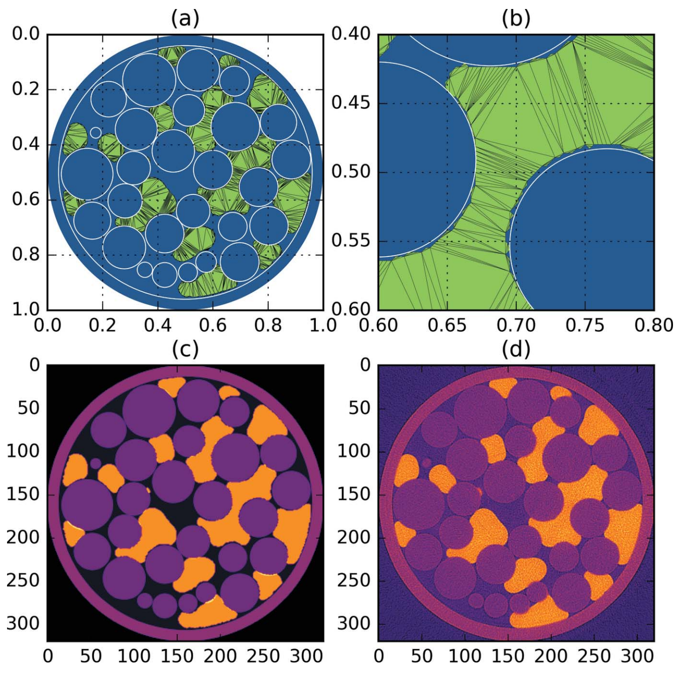


Figure 6
Soil-like phantom (*a*, *c*) with wetting phase constructed from triangular mesh (*b*) and two other phases constructed from circles to resemble a source image (*d*) as seen in Narter & Brusseau (2010). The circles were extracted from the source image using canny edges and a Hough tranform. The wetting phase was extracted from the source using simple thresholding, converted to a contour using the marching squares algorithm, and then converted to a triangular mesh using Python Triangle (https://github.com/drufat/triangle).

2.3. Structurally complex phantoms

The geometry module of *XDesign* has three main kinds of entities: simple entities (points and lines), curves (defined by a single equation) and polytopes (defined by multiple equations). We use the polytope library (Filippidis *et al.*, 2016) for computational geometry because it made adding polygonal intersections and expansion into ND space easy. Curves and polytope meshes may be combined in the same phantom (Fig. 6) in order to simulate complex structures like liquids wetting soils (Schlüter *et al.*, 2014). Any number of properties can be added to the features in a phantom because Python allows for dynamic assignment of attributes to objects. Properties could be anything: attenuation, density, grain orientation, crystal structure, *etc.*

3. Data acquisition simulation

The *XDesign* data acquisition currently has one type of probe object. It takes measurements by integrating a property (*e.g.* linear attenuation) of the phantom over the space contained by the probe. A single beam probe can simulate both scanning probes and area detectors because it can be moved and rotated to any position.

3.1. Generating a sinogram

To simulate data acquisition, create a probe object and then code it through a procedure. The probe records data when the measure method is called, and it moves when translated or rotated. The example below simulates raster-scanning of the phantom from §2.1 with standard parallel beam over 180° rotation:

```
# Define raster-scan parameters
sx, sy = 100, 100

# Step size of the probe for raster scanning
step = 1. / sy

# Initial probe creation
probe = Probe(Point([step / 2., -10]),
Point([step / 2., 10]), 0.01)

# Step size of uniformly spaced projection angles
theta = np.pi / sx

# Initialize sinogram array
sinogram = np.zeros(sx * sy)

# Collect data
a = 0
for m in range(sx):
     for n in range(sy):
          # Calculate simulated data for measurement
          sinogram[a] = probe.measure(face)
          a + = 1

          # Translate probe by step size
          probe.translate(step)

     # Translate probe back to original position
     probe.translate(-1)

     # Rotate probe by the angle step size around
     origin
     probe.rotate(theta, Point([0.5, 0.5]))
```

3.2. Implementation details

Because scattering and other effects of the beam are not modeled at this time, there is no detector object. Equivalent sinograms can be generated by moving the phantom or the probe because all positions and movements are described from a global reference frame. Streaming reconstruction can easily be simulated because simulated data are available as soon as they are calculated.

4. Reconstruction algorithms

XDesign includes a number of unoptimized reconstruction algorithms for tinkering: *ART*, *SIRT* and *MLEM*, but we expect for non-simulated experiments researchers will use more efficient implementations.

4.1. Reconstructing data

In *XDesign*, the probe captures a snapshot of its geometry when it measures data and appends it in a list to be used for reconstruction. The example script below shows the use of built-in reconstruction methods in the package. Fig. 7 demonstrates image reconstructions of the Soil phantom (Fig. 8) on a uniformly spaced grid using *GridRec*, *PML* and *SIRT* algorithms.

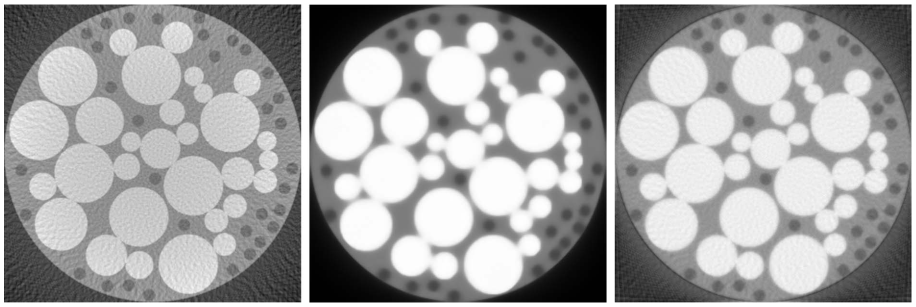


Figure 7
Image reconstructions of the soil phantom on a uniformly spaced grid using, from left to right, *GridRec*, *PML* and *SIRT* algorithms. We employed TomoPy for obtaining the *GridRec* and *PML* reconstructions, and *XDesign* for obtaining the *SIRT* reconstruction.

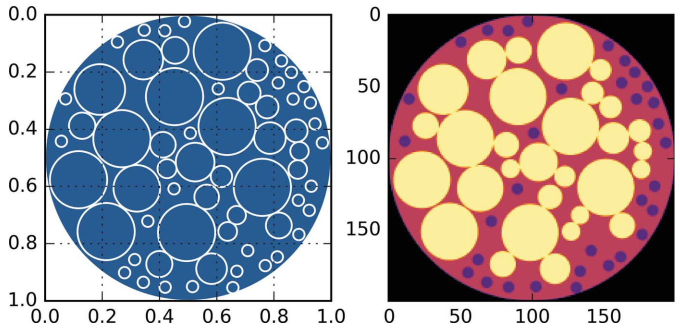


Figure 8
The Soil phantom geometry (left) and its discretization (right) are shown.

```
# Reconstruction grid size
rx, ry = 100, 100

# Number of iterations
niter = 20

# Initial phantom estimate
init = np.zeros((rx, ry))

# Reconstruct using SIRT algorithm
recon = sirt(probe, sinogram, init, niter)
```

5. Image quality metrics

There are three classes of image quality metrics: full reference, partial reference and no reference. Full reference metrics generally measure the amount of shared information between a reference and distorted image. The importance of different types of information, *i.e.* edge intensity, color and contrast, is weighted differently in various methods, and the result is only applicable to a particular image. Partial reference metrics are used when the full reference exists but is not reliably accessible. No reference methods often try to measure the highest resolvable frequency or noise content of an image-capturing system by using a standard test pattern; these quantities are believed to predict the quality of all images captured by a system. Our tool includes full reference and no reference metrics.

5.1. Full reference image quality metrics

This section briefly describes each of the available full reference image quality metrics. However, for full technical descriptions, the reader should refer to each method's original publication.

5.1.1. MS-SSIM. The multiscale structural similarity index (MS-SSIM) measures differences in 'luminance, contrast and structure' at multiple levels of detail (Wang *et al.*, 2003). Each of these three qualities is calculated from a combination of the local mean, standard deviation and covariance of images. Using local means and standard deviations calculated from Gaussian filters it is possible to calculate a contour map of image quality at multiple resolution scales. Similarity overall is calculated by averaging the structural similarity index over the entire image and at each scale.

5.1.2. FSIM. The feature similarity index (FSIM) measures the similarity of images using gradient magnitude and Fourier phase congruency (Zhang *et al.*, 2011). Because high phase congruency had been correlated with image features important to the human visual system, this method weights the importance of each gradient magnitude depending on phase congruency. Since the gradient magnitude is only a measure of edges, this method ignores whether luminance is correctly captured, but that might not be important for some users.

5.1.3. VIFp. The visual information fidelity in the pixel domain (VIFp) measures shared information between a reference and distorted image using a framework based on natural scene statistics (Sheikh & Bovik, 2006). It uses Gaussian scale mixtures and wavelet analysis. It directly compares the intensity information in the images at different scales by separating it into levels using Gaussian filters of different sizes. Because it uses wavelets, the accuracy of this quality metric is dependent on the depth of the wavelet transform.

5.2. No reference image quality metrics

These metrics use predetermined phantom geometries to estimate the noise characteristics and minimum resolvable spatial frequencies of a system (Hsieh, 2003).

5.2.1. NPS. Noise power spectra (NPS) uses a Fourier transform of uniform area in the reconstruction to give information about the frequency composition of the noise. This is better than the signal-to-noise ratio because it shows how coarse or fine the noise is. Two-dimensional images produce a two-dimensional noise power spectrum, but the two-dimensional spectrum can be reduced to a histogram by binning radially.

5.2.2. SFR and MTF. Spatial frequency response (SFR) and modulation transfer function (MTF) commonly use a slanted edge or standard pattern of lines at increasingly smaller intervals to measure how the fidelity of an image decreases as the frequency of a signal increases. The ability of an imaging system to accurately capture high-frequency signals is related to the sharpness of the images it creates. We have implemented the MTF calculation described by Friedman *et al.* (2013) which does not normalize the zero frequency to unity in order to prevent artificially inflating responses at other frequencies.

5.3. Using image quality metrics

In order to use the full reference quality metrics in *XDesign* you need to generate a reconstructed phantom, discretize the source phantom on a uniform grid to the same size, and choose a method for comparison. The compute_quality function will generate average and local quality for a series of images:

```
# Create a discrete phantom of size 100
ref = discrete_phantom(a_phantom, 100)

# Compare true phantom with reconstructed values
  using MS-SSIM metric
metrics = compute_quality(ref, [rec_art, rec_sirt,
  rec_mlem], method="MSSSIM")

# Plot analysis results
plot_metrics(metrics)
```

For full-reference comparison, local quality information is calculated at multiple scales. Scale is the standard deviation of the filter size used to compute the local quality metric. In other words, the quality at scale = 5 tells you how well objects on the order of 10 pixels are represented. Fig. 9 shows the result of compute_quality with the MS-SSIM metric applied to the 'cameraman' test image which has been deformed in four different ways. The distorted image is plotted in the upper left and a sequence of contour plots of the quality metric are plotted for each scale. Larger plots show information about smaller scales and smaller plots show information about larger scales.

The combination of quality metrics at multiple scales and parameterized functions allows us to generate informative plots about an XCT experimental method. If we wanted to evaluate whether *ART*, *SIRT* or *MLEM* is optimal for reconstructing the Soil phantom (Fig. 8), we can try each of the methods at different numbers of iterations and plot the results. Optimal tuning of configuration parameters of any iterative reconstruction algorithm (*e.g.* regularization parameter, number of iterations, different updating schemes) can also be quantified by calculating this set of metrics.

In Fig. 10 we can see that *ART* only scores best on smaller scales with few iterations. Beyond that, it creates its best reconstruction at around 50 iterations before decreasing in quality again. *MLEM* is the best algorithm for this phantom because the quality reconstruction rapidly increases at all scales faster than *ART* and *SIRT*. The *SIRT* contour looks like the *MLEM* contour but the quality improves at a slower pace. *SIRT* might be a better option if its time per iteration is much smaller than that of *MLEM*.

We can also calculate no reference metrics using standard phantoms. In Fig. 11 we have calculated the MTF for *ART*, *SIRT* and *MLEM* at various numbers of iterations. In this comparison it once again seems that *MLEM* is the best reconstruction method at higher numbers of iterations because the MTF most rapidly approaches unity.

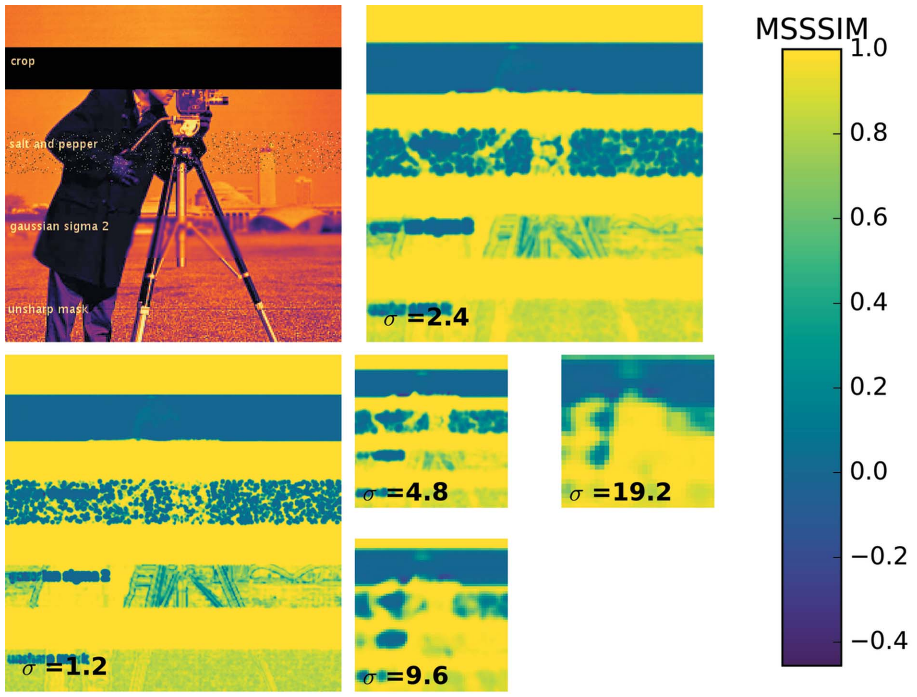


Figure 9
Example output from compute_quality using the MS-SSIM quality metric. The source image is the 'cameraman' image which has been distorted in four ways: crop, salt and pepper, Gaussian smoothing and unsharp masking.

6. Future works and proposed features

6.1. Future works

Here are some features which we are currently working on to add to *XDesign*.

6.1.1. Wave propagation. *XDesign* only simulates beam attenuation, and it does not have a detector object. Therefore, phase contrast and other reconstruction methods which use beam scattering or wave interference cannot be simulated at this time. In a future release, we are planning to simulate Fresnel multislice wave propagation.

6.1.2. Geometric flexibility. As a pre-release, *XDesign* only supports two-dimensional geometries. We are currently working to enable the definition of higher-dimensional phantoms which will enable freedom of problem definition, and allow for more flexible data acquisition, *e.g.* three-dimensional

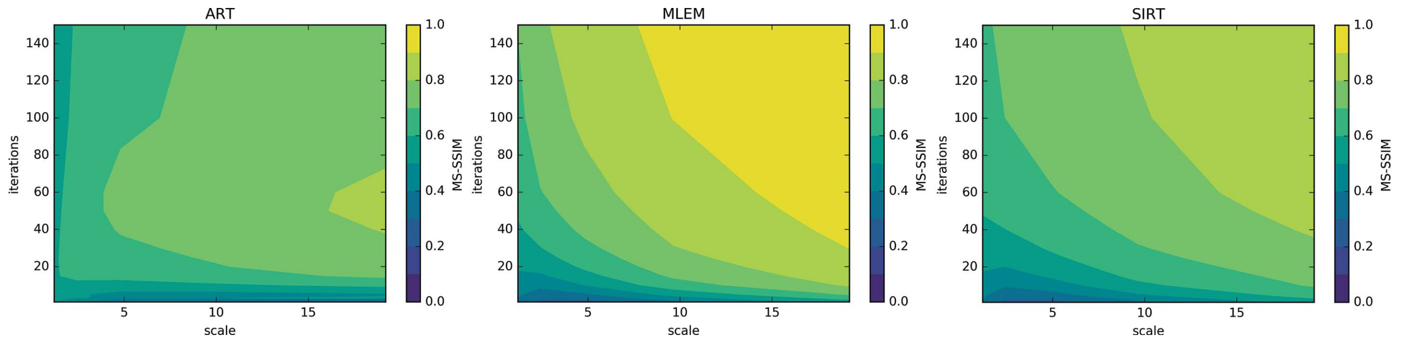


Figure 10
MS-SSIM quality contours for the reconstruction of the Soil phantom in Fig. 8 using *ART*, *MLEM* and *SIRT* reconstruction.

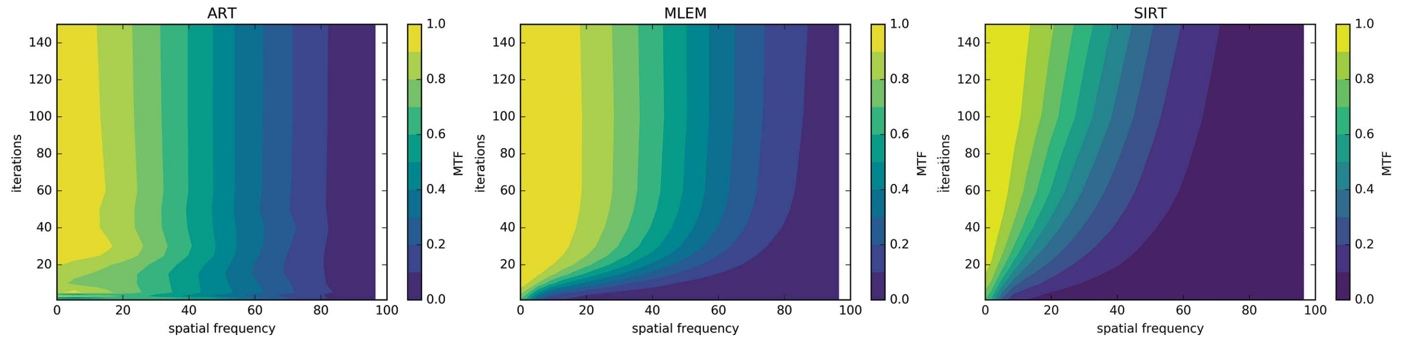


Figure 11
MTF quality contours from the unit circle phantom in Fig. 6 using *ART*, *MLEM* and *SIRT* reconstruction. MTF values at zero-frequency are not normalized to unity according to Friedman *et al.* (2013).

objects can be used to model three-dimensional objects, but they can also be used for modeling a dynamic two-dimensional object. We also plan to add a component class to the phantom hierarchy. Components will spatially attach features of different properties together to allow for easier geometric manipulation.

6.1.3. Performance. Python is slower than compiled languages. We are working to optimize computationally intensive portions of the data acquisition module for faster run-times. We plan to implement bottleneck functions in compiled languages such as C or Fortran, and also explore use of graphical processing units for vectorizing and further speeding-up calculation of slow processes if needed.

6.2. Proposed features

Here are some feature which have been suggested, but are not yet assigned to anyone for development. Features in this section are not currently planned because they either require greater involvement from the community or more man hours. For additional feature suggestions or to participate in our development process, please head to our GitHub page and create an issue.

6.2.1. Distortions. Some processing methods are targeted at reducing a specific type of noise, *e.g.* motion blur or beam drift. At this time, simulated distortions like these are not implemented, but they could be simulated by adding random noise to methods in the data acquisition module.

6.2.2. Materials definitions. Currently, feature properties must be defined manually. We could define a materials superclass of feature for common compounds such that materials properties of these compounds are auto-populated from tabulated data of the National Institute of Standards and Technology (Berger & Hubbell, 1987) or the Center for X-ray Optics (Henke *et al.*, 1993).

6.2.3. Interfaces. Import and export of experimentation acquisition systems for better determining collection parameters is important. We could implement import and export interfaces to common file formats like *stl* for three-dimensional meshes and *svg* for two-dimensional phantoms. This would improve portability of phantoms and acquisition geometry between researchers. It could also allow for the visualization of experimental setups by using third-party commercial tools and make integration with existing tools easier.

6.2.4. Improved quality measures. Some algorithms are specialized for reconstructing the shape only, and thus the dynamic range of the phantom may not match the dynamic range of the reconstruction. Full reference quality metrics that can compare images of different dynamic ranges will be useful if they can be implemented. We could expand the metrics library to evaluate special reconstructing methods based on community interest.

6.2.5. Community repository. A collection of free-to-use materials phantoms generated by the community would help algorithms developers test their methods more robustly. We are working with synchrotron users to build up a parametrized phantoms library and encourage anyone to contribute their custom parameterized phantoms to the *XDesign* GitHub repository.

Acknowledgements

This work is supported by the US Department of Energy under Contract No. DE-AC02-06CH11357.

References

Bergamaschi, A., Medjoubi, K., Messaoudi, C., Marco, S. & Somogyi, A. (2016). *J. Synchrotron Rad.* **23**, 783–794.

Berger, M. J. & Hubbell, J. H. (1987). *Natl Bur. Stand. Publ.* 87-3597.

Filippidis, I., Dathathri, S., Livingston, S. C., Ozay, N. & Murray, R. M. (2016). *Proceedings of Multi-Conference on Systems and Control*, Buenos Aires, Argentina. IEEE. (Tutorial paper.)

Friedman, S. N., Fung, G. S., Siewerdsen, J. H. & Tsui, B. M. (2013). *Med. Phys.* **40**, 051907.

Gürsoy, D., De Carlo, F., Xiao, X. & Jacobsen, C. (2014). *J. Synchrotron Rad.* **21**, 1188–1193.

Henke, B. L., Gullikson, E. M. & Davis, J. C. (1993). *At. Data Nucl. Data Tables*, **54**, 181–342.

Holman, B. F., Cuplov, V., Hutton, B. F., Groves, A. M. & Thielemans, K. (2016). *Phys. Med. Biol.* **61**, 3148–3163.

Hsieh, J. (2003). *Computed Tomography: Principles, Design, Artifacts and Recent Advances*, *Volume 114 of SPIE Press Monograph.* Bellingham: SPIE.

Hsieh, J., Londt, J., Vass, M., Li, J., Tang, X. & Okerlund, D. (2006). *Med. Phys.* **33**, 4236–4248.

Hunter, J. D. (2007). *Comput. Sci. Eng.* **9**, 90–95.

Jan, S., Santin, G., Strul, D., Staelens, S., Assié, K., Autret, D., Avner, S., Barbier, R., Bardiès, M., Bloomfield, P. M., Brasse, D., Breton, V., Bruyndonckx, P., Buvat, I., Chatziioannou, A. F., Choi, Y., Chung, Y. H., Comtat, C., Donnarieix, D., Ferrer, L., Glick, S. J., Groiselle, C. J., Guez, D., Honore, P. F., Kerhoas-Cavata, S., Kirov, A. S., Kohli, V., Koole, M., Krieguer, M., van der Laan, D. J., Lamare, F., Largeron, G., Lartizien, C., Lazaro, D., Maas, M. C., Maigne, L., Mayet, F., Melot, F., Merheb, C., Pennacchio, E., Perez, J., Pietrzyk, U., Rannou, F. R., Rey, M., Schaart, D. R., Schmidtlein, C. R., Simon, L., Song, T. Y., Vieira, J. M., Visvikis, D., Van de Walle, R., Wieërs, E. & Morel, C. (2004). *Phys. Med. Biol.* **49**, 4543–4561.

Liu, Y., Meirer, F., Williams, P. A., Wang, J., Andrews, J. C. & Pianetta, P. (2012). *J. Synchrotron Rad.* **19**, 281–287.

Marchesini, S., Krishnan, H., Daurer, B. J., Shapiro, D. A., Perciano, T., Sethian, J. A. & Maia, F. R.N.C. (2016). *arXiv*:1602.01448.

Narter, M. & Brusseau, M. L. (2010). *Water Resour. Res.* **46**, W08602.

Palenstijn, W. J., Batenburg, K. J. & Sijbers, J. (2013). *13th International Conference on Computational and Mathematical Methods in Science and Engineering (CMMSE 2013)*, 23 27 June 2013, Almeria, Spain.

Patterson, B. M., Cordes, N. L., Henderson, K., Williams, J., Stannard, T., Singh, S. S., Ovejero, A. R., Xiao, X., Robinson, M. & Chawla, N. (2016). *J. Mater. Sci.* **51**, 171–187.

Schlüter, S., Sheppard, A., Brown, K. & Wildenschild, D. (2014). *Water Resour. Res.* **50**, 3615–3639.

Sheikh, H. R. & Bovik, A. C. (2006). *IEEE Trans. Image Process.* **15**, 430–444.

Shepp, L. A. & Logan, B. F. (1974). *IEEE Trans. Nucl. Sci.* **21**, 21–43.

Thielemans, K., Mustafovic, S. & Tsoumpas, C. (2006). *IEEE Nucl. Sci. Symp. Conf. Rec.* **4**, 2174–2176.

Treeby, B. E. & Cox, B. T. (2010). *J. Biomed. Opt.* **15**, 021314.

Vogelgesang, M., Farago, T., Morgeneyer, T. F., Helfen, L., dos Santos Rolo, T., Myagotin, A. & Baumbach, T. (2016). *J. Synchrotron Rad.* **23**, 1254–1263.

Wang, Z., Simoncelli, E. P. & Bovik, A. C. (2003). *Conference Record of the Thirty-Seventh Asilomar Conference on Signals, Systems and Computers.* IEEE.

Zanaga, D., Bleichrodt, F., Altantzis, T., Winckelmans, N., Palenstijn, W. J., Sijbers, J., de Nijs, B., van Huis, M. A., Sánchez-Iglesias, A., Liz-Marzán, L. M., van Blaaderen, A., Batenburg, K. J., Bals, S. & Van Tendelooa, G. (2016). *Nanoscale*, **8**, 292–299.

Zhang, L., Zhang, L., Mou, X. & Zhang, D. (2011). *IEEE Trans. Image Process.* **20**, 2378–2386.