Title
Xi-cam: a versatile interface for data visualization and analysis

Journal
Journal of Synchrotron Radiation, 25(4)

ISSN
0909-0495

Authors
Pandolfi, Ronald J
Allan, Daniel B
Arenholz, Elke
et al.

Publication Date
2018-07-01

DOI
10.1107/s1600577518005787

Peer reviewed

JOURNAL OF SYNCHROTRON RADIATION

ISSN 1600-5775

Received 31 January 2018
Accepted 13 April 2018

Edited by M. Eriksson, Lund University, Sweden

Keywords: scattering; tomography; NEXAFS; XAS; spectroscopy; SAXS; GISAXS; WAXS; GIWAXS; software.

Xi-cam: a versatile interface for data visualization and analysis

Ronald J. Pandolfi,[a]* Daniel B. Allan,[b] Elke Arenholz,[a] Luis Barroso-Luque,[a] Stuart I. Campbell,[b] Thomas A. Caswell,[b] Austin Blair,[a] Francesco De Carlo,[c] Sean Fackler,[a] Amanda P. Fournier,[d] Guillaume Freychet,[a] Masafumi Fukuto,[b] Doğa Gürsoy,[c,e] Zhang Jiang,[c] Harinarayan Krishnan,[a] Dinesh Kumar,[a] R. Joseph Kline,[f] Ruipeng Li,[b] Christopher Liman,[f] Stefano Marchesini,[a] Apurva Mehta,[d] Alpha T. N'Diaye,[a] Dilworth Y. Parkinson,[a] Holden Parks,[a] Lenson A. Pellouchoud,[a] Talita Perciano,[a] Fang Ren,[d] Shreya Sahoo,[a] Joseph Strzalka,[c] Daniel Sunday,[f] Christopher J. Tassone,[a] Daniela Ushizima,[a] Singanallur Venkatakrishnan,[g] Kevin G. Yager,[h] Peter Zwart,[a] James A. Sethian[a] and Alexander Hexemer[a]*

[a]Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA, USA, [b]National Synchrotron Light Source II, Brookhaven National Laboratory, Upton, NY, USA, [c]Advanced Photon Source, Argonne National Laboratory, 9700 South Cass Avenue, Lemont, IL, USA, [d]Stanford Synchrotron Radiation Lightsource, SLAC National Accelerator Laboratory, 2575 Sand Hill Road, Menlo Park, CA, USA, [e]Department of Electrical Engineering and Computer Science, Northwestern University, 2145 Sheridan Road, Evanston, IL, USA, [f]National Institute of Standards and Technology, 100 Bureau Drive, Gaithersburg, MD, USA, [g]Oak Ridge National Laboratory, PO Box 2008, Oak Ridge, TN, USA, and [h]Center for Functional Nanomaterials, Brookhaven National Laboratory, Upton, NY, USA.
*Correspondence e-mail: ronpandolfi@lbl.gov, ahexemer@lbl.gov

Xi-cam is an extensible platform for data management, analysis and visualization. *Xi-cam* aims to provide a flexible and extensible approach to synchrotron data treatment as a solution to rising demands for high-volume/high-throughput processing pipelines. The core of *Xi-cam* is an extensible plugin-based graphical user interface platform which provides users with an interactive interface to processing algorithms. Plugins are available for SAXS/WAXS/GISAXS/GIWAXS, tomography and NEXAFS data. With *Xi-cam*'s 'advanced' mode, data processing steps are designed as a graph-based workflow, which can be executed live, locally or remotely. Remote execution utilizes high-performance computing or de-localized resources, allowing for the effective reduction of high-throughput data. *Xi-cam*'s plugin-based architecture targets cross-facility and cross-technique collaborative development, in support of multi-modal analysis. *Xi-cam* is open-source and cross-platform, and available for download on GitHub.

1. Introduction

Efficient use of synchrotron beamline time is critical to synchrotron user and beamline productivity. With increasing data rates and proliferation of high-throughput sample switching, data analysis and management is increasingly becoming the limiting factor of experiments. Live, intuitive, interactive data viewing and reduction is necessary to provide users with the best data quality feedback as they work.

Xi-cam is a graphical plugin-based platform for organizing, viewing and analyzing X-ray scattering/grazing-incidence X-ray scattering images developed by the Center for Advanced Mathematics for Energy Research Applications (CAMERA). This is a cross-platform open-source Python project licensed under the Berkeley Software Distribution (BSD) license.

There are many existing tools for data reduction, analysis and fitting. For X-ray scattering data, reduction and analysis of area detector images can be performed using *Fit2d* (Hammersley, 2016), *view.gtk* (Yang, 2013) or *GIXSGUI* (Jiang, 2015). Small-angle X-ray scattering or grazing-incidence small-angle X-ray scattering (SAXS or GISAXS) data can be fit using a variety of models, as implemented in *ATSAS* (Franke *et al.*, 2017), *ScatterSim* (Yager *et al.*, 2014), *IsGISAXS* (Lazzari, 2002), *BornAgain* (Durniak *et al.*, 2015) and *HipGISAXS* (Chourou *et al.*, 2013). While many of these software tools allow for batch processing, there is currently no uniform analysis environment allowing users to reduce, explore, analyze and quantitatively fit their data within a single environment. *Xi-cam*'s uniform set of controls and plugin architecture provide just such an environment.

The *Xi-cam* interface is built from a software base designed for portability and responsiveness without sacrificing performance. The front-end interface utilizes Qt4 and the PyQtGraph library for displaying images and plotting. Images and plots can be manipulated and interacted with as data are explored. Expensive calculations have additional multi-core CPU/GPU implementations optionally invoked with supporting hardware/drivers.

Xi-cam provides an interface for browsing data locally or from a Globus data server (Ave *et al.*, 2017). Following a cloud data model, *Xi-cam* has instant remote access to data acquired at Advanced Light Source beamlines; data are streamed to the National Energy Research Scientific Computing Center (NERSC) data center as they are collected, and are instantly available for remote access. Remote data access is provided through Databroker (Databroker Development Team, 2017), Globus (Ave *et al.*, 2017), the SPOT data interface (SPOT Development Team, 2017) and SFTP.

The *Xi-cam* back-end is designed to run both as a client to process local data and a server daemon. As a server daemon, image data can be pre-processed as part of the workflow pipeline for storage. Expensive calculations (*e.g.* remeshing) can be cached and packaged along with the original data in NeXus (Könnecke *et al.*, 2015) files for rapid viewing in the *Xi-cam* interface.

The plugin-based design of *Xi-cam* allows for both extensibility and cross-technique interaction, with plans for cooperative multi-modal analysis. The plugin framework allows developers to implement techniques with unique interfaces and processing while sharing global platform features such as the remote data interface, IPython console, and live processing.

Xi-cam is available for download at: https://github.com/ronpandolfi/Xi-cam. *Xi-cam*'s documentation is available at: http://xi-cam.readthedocs.io/en/latest/.

2. *Xi-cam* platform

2.1. Plugin development

The plugin development process for *Xi-cam* is documented at http://xi-cam.readthedocs.io/. The standard *Xi-cam* plugin requires some knowledge of Qt and object oriented programming. While this is a robust and flexible design, it is often less accessible for scientific users testing custom processing in *Xi-cam*. A convenience plugin factory method is available, which greatly simplifies this task.

The *Xi-cam* EZPlugin provides, as a function, a simple plugin constructor. This constructor allows easy customization of the plugin name, toolbuttons, editable parameters, file operations and visualizations. All the features of the standard plugin are also available.

2.2. Remote execution

Remote execution in *Xi-cam* is handled through two core components: the Paramiko-based[1] (Paramiko Development Team, 2017) remote connection interface and Dask-Distributed (Dask Development Team, 2016), a flexible parallel computing library for analytic computing.

Paramiko, a Python-based ssh library, provides a secure remote connection interface between the *Xi-cam* application and the remote service. Additionally, *Xi-cam* uses Paramiko to handle any hops between client and destination through implicit port forwarding and creating additional ssh tunnels, which is often the case in supercomputing environments. Furthermore, the interface also sets up an appropriate execution environment for the destination service such as ensuring use of batch schedulers for workers, and that the appropriate paths are set for finding remote libraries and executables.

The second component utilizes Dask-Distributed to transfer *Xi-cam* workflows to the remote service. The Dask-Distributed API provides a mechanism to create and execute a local workflow graph remotely and provides an asynchronous event handler to notify whether the remote execution has completed successfully or whether an error has occurred. There are three main Dask integration points in *Xi-cam*:

(i) The local Dask-Distributed interface, which sets up and maintains the connection between the local and remote scheduler.

(ii) The Dask scheduler, which parcels out work and communicates results to the local interface.

(iii) The Dask worker, which executes the Python algorithm sent from *Xi-cam*.

2.3. Hardware compatibility

Hardware profiles for a wide variety of detectors are available through the pyFAI (Ashiotis *et al.*, 2015) API. *Xi-cam* heuristically identifies detector models automatically for uniquely shaped detectors (*i.e.* Pilatus); for less unique detectors, the detector profile can be selected manually in the experimental configuration. For detectors with inactive area

[1] Certain commercial equipment, instruments or materials are identified in this paper in order to specify the experimental procedure adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the materials or equipment identified are necessarily the best available for the purpose.

(Mar, Pilatus, *etc.*), pre-defined detector masks are automatically applied.

2.4. Data formats

Core functionality is provided for loading and saving data in a variety of formats. Interfaces for many formats are provided through the FabIO package (see the FabIO documentation for a list of compatible formats; Knudsen *et al.*, 2013). Further formats are provided through extensions to FabIO. The following additional formats are made available through FabIO extensions:

(i) FITS format, provided by astropy (Robitaille *et al.*, 2013).

(ii) DXchange, provided by dxchange (De Carlo *et al.*, 2014).

(iii) NeXus provided by nexusformat (Könnecke *et al.*, 2015).

(iv) Igor Pro General Binary, .gb (Nika/Irena) (Ilavsky, 2012).

(v) RAW.

(vi) Multi-image tiff.

(vii) Other beamline-specific HDF5 formats.

To implement an additional format, a class inheriting from fabio.fabioimage must be registered with fabio. Such a class requires as minimum: a single method taking a filepath as input and reading array data; a list of compatible file extensions. For full details, see the FabIO documentation (Knudsen *et al.*, 2013), template at fabio/templateimage.py, and examples in pipeline/formats.py of *Xi-cam*.

3. SAXS toolset

Xi-cam features an extensive toolset for SAXS data, including single-frame reduction ('Viewer'), multi-frame series reduction ('Timeline'), batch reduction, forward GIXS simulation ('*HipGISAXS*'), reverse Monte Carlo ('HipRMC') and CDSAXS (critical dimension small-angle X-ray scattering) fitting.

3.1. One-click calibration

As part of an automated data pipeline, a 'hands-free' calibration procedure is provided. First, the beam center is identified by one of three methods: autocorrelation, circular wavelets and circle fitting. A pixel-space azimuthal integration is calculated from that center, from which the first calibrant peak position is identified; a first approximation of sample–detector distance is made from the beam center position and first calibrant peak. This is used as input to DPDAK's refinement algorithm (Benecke *et al.*, 2014), which also fits additional further peaks. Using this procedure, small detector tilt may be corrected in the refinement stage automatically. A heuristic optionally determines the detector model prior to the above calibration steps. A detector model matching the file extension and image size is identified from the available FabIO detector profiles. Pixel shape, inactive areas (*i.e.* module gaps from Pilatus detectors) and binning are loaded from the FabIO profile. For confirmation, an overlay is displayed, simulating the calibrant scattering pattern (shown as green rings in Fig. 1). In case the automatic calibration algorithms are insufficient, calibration parameters may be set and adjusted manually, including detector model.

Figure 1
Automated calibration of silver behenate (AgB) data (center panel) by autocorrelation. The AgB rings (yellow) are directly obscured by the simulated calibrant rings (green), and the first AgB peak is translated to a q value of 0.106 3^{-1}, indicating proper calibration. The reduced one-dimensional spectra is immediately displayed (bottom).

3.1.1. Autocorrelation. With autocorrelation-based center detection, the maxima position of the autocorrelation of a calibrant image provides an estimation of the beam center. This is a fast method when using fast-Fourier-transform-based convolution. It is highly effective when more than 180°of a calibrant ring is visible; a geometry with the beam center near a corner or off the image causes failure. A heuristic algorithm further identifies the order of the first ring.

3.1.2. Circular wavelets. Continuous wavelet transforms (CWTs) are widely used in image processing for pattern recognition (Carmona *et al.*, 1998; Du *et al.*, 2006). A CWT of a signal $s(x)$ can be represented as (Daubechies, 1992)

$$C(\mu,\sigma)=\frac{1}{\sqrt{\sigma}}\int_{-\infty}^{\infty}s(x)\,\psi\left(\frac{x-\mu}{\sigma}\right)\mathrm{d}x, \qquad (1)$$

where $\psi(x)$ is a continuous function of σ and μ, the mother wavelet. The mother wavelet is used to generate daughter wavelets by varying either or both of the independent variables σ representing the width and μ representing the location. As the wavelets are slid across the signal $s(x)$, the value of $\|C(\mu,\sigma)\|_\infty$ changes, with the maximum obtained when the location and width of the wave matches one of the calibrant rings. The second derivative of the Gaussian distribution, also known as the Ricker wave, is a commonly used wavelet. We designed a two-dimensional radial Ricker wavelet, by modifying the orginal, for our purpose,

$$\psi(r,\theta)=\psi\big[\mu(r),\sigma,\theta\big]. \qquad (2)$$

Thus the value of C is maximum when r and σ equal the radius and width of the ring, respectively. Provided that the search

range is appropriate, this technique can fit properly even if the beam center projection is outside the image.

3.2. Tilt auto-calibration

Detectors oriented with some tilt, such that the point of normal incidence is not the point of intersection with the direct beam, require a more detailed procedure to detect the experimental geometry. A general outline of this procedure is:

(i) Generate a point cloud from points along each calibrant ring.

(ii) Segment the point cloud into separate calibrant rings.

(iii) Fit an ellipse to each point cloud.

(iv) For each ellipse, calculate major/minor axes, tilt (from eccentricity) and the position of the direct beam.

(v) Cross-correlate parameterized ellipses, defining a single geometry which best matches the set.

It is important to note that (often contrary to intuition), the direct-beam position is not at the center of a calibrant ellipse. Rather, for detector tilt angle α and calibrant ring defined by scattering double-angle 2θ, the direct beam position P between major axis ends A and B is defined such that

$$\frac{\overline{PA}}{\overline{PB}} = \frac{\cos(2\theta - \alpha)}{\cos(2\theta + \alpha)}. \tag{3}$$

3.3. Masking

Four types of masking tools are available for use in *Xi-cam*. A detector active area mask is automatically applied when a frame is read, masking out inactive or anisotropically sensitive regions of the pixel-space image. A zinger/cosmic ray masking tool uses the Astropy cosmics detection algorithm to mask particle tracks from cosmic background. A threshold masking tool allows masking of pixels below a threshold value, including (optionally) a neighborhood size for morphological closing. Finally, a polygon masking tool allows the user to directly define any polygonal shape to mask over the image.

Symmetry operation tools can be used to fill in masked regions of the image using information from symmetrically opposing areas. Other algorithms for data infill (for display purposes) have been considered as future extensions of this functionality, including a heat equation solver, bilinear interpolation and a scattering physics-aware 'healing' algorithm (Liu *et al.*, 2017).

3.4. Timeline

Visualization of reduced data along independent axes provides further insight for differential measurements (see Fig. 2). Timeline is a mode for series data analysis which visualizes changes across a single parameter space. This can be a time-resolved series, or a scan of another parameter such as temperature, energy, incoming angle, solution composition, *etc*. In the default mode, the lower panel of Timeline displays a single one-dimensional plot of the χ^2 difference between consecutive frames. A variety of other functions are included for other modes of analysis, including azimuthal integration. Custom functions are supported, allowing more specialized

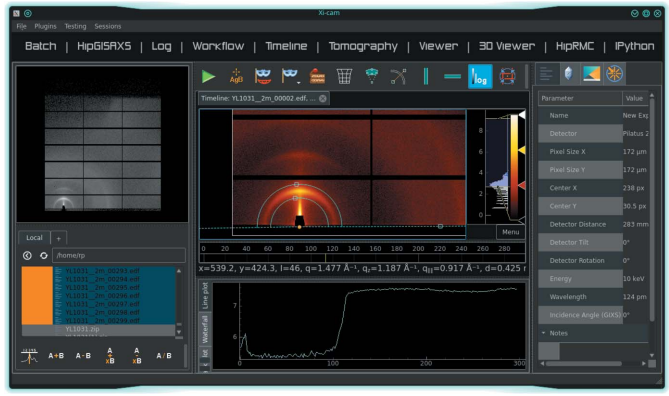

Figure 2
A 300 frame data series from an OPV drying experiment loaded in Timeline. The crystalline peak arc has been selected on the image (cyan). The reduced data displayed in the bottom panel show the integrated intensity in the selected region. This shows that crystallization began around frame 100; however, dynamics continue to drive structural change to a lesser degree, as indicated by a drop in intensity towards frame 250.

metrics. Region-of-interest cuts can also be defined to select the region to process. Peak tracking optionally adjusts the region of interest for each frame as peaks shift by toleranced maximum convolution. The Timeline mode supports custom functions for visualization by adding to the variationoperators.py module. New functions must follow the template input signature, and return a Float, Collections.OrderedDict or one-dimensional numpy.ndarray to generate a line plot, multi-line plot or waterfall plot, respectively.

3.5. CDSAXS

To meet the challenges of sub-10 nm feature characterization, new methods are under development by the semiconductor industry. Critical-dimension small-angle X-ray scattering (CDSAXS) has been targeted as promising for the dimensional control of line gratings with nanometer scale resolution. The transmission scattering geometry enables a 100 µm × 100 µm patterned area to be probed. CDSAXS measurements have been shown to be sensitive to shape and structural properties of line gratings, including the pitch, linewidth, line profile (line height and sidewall angle) and line roughness (Hu *et al.*, 2004; Sunday *et al.*, 2015; Wang *et al.*, 2009).

The fast development of the CDSAXS technique led to the appearance of several analytical models and experimental approaches, developed mostly in synchrotrons during the last few years. Moreover, a laboratory-source instrument was developed by NIST allowing increased private use of CDSAXS. However, there have been no open-source software available to analyze CDSAXS patterns. The development of CDSAXS in *Xi-cam* will provide a new CDSAXS approach, going from the experimental data through the calculation of (q_x, q_z) cartography to the fitting of the line profile, all from a single interface.

Our approach is based on the work of Hannon *et al.* (2016). A genetic algorithm, using a covariance matrix adaptation

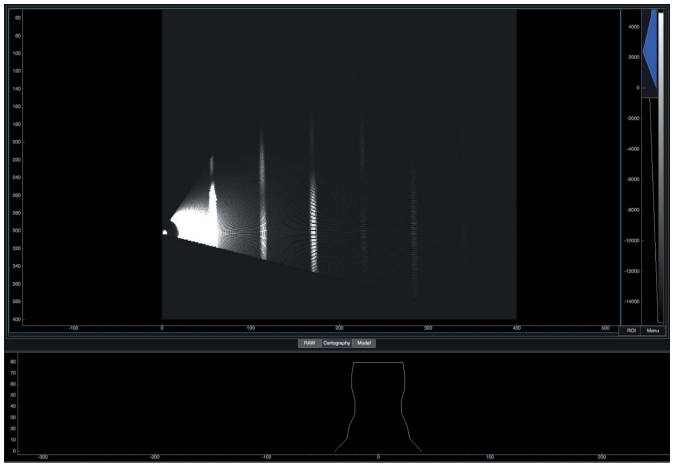

Figure 3
Cartographic (q_x, q_z) transformation from raw CDSAXS data. The vertical lines indicate locations where one-dimensional peak profiles would be extracted for fitting. Fitting these profiles can provide estimations for critical dimensions of the periodic form factor.

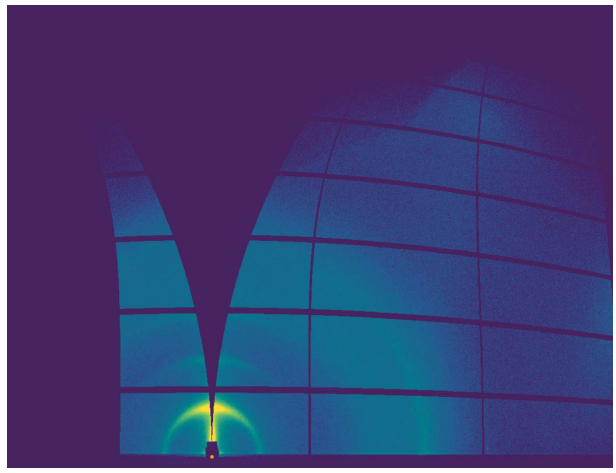

Figure 4
Proper conversion of a GIWAXS image into Q-space results in the missing vertical wedge, as a result of the projection of the Ewald sphere. Reflections within this forbidden region are not accessible.

evolutionary strategy, has been developed in order to minimize the mean-absolute error log goodness between the experimental and simulated data. The user-friendly GUI interface allows every user to extract the full information from experimental images, by only giving the initial guess of dimension of the line profile. The CDSAXS plugin takes as input raw data generated at each angle. After procedural alignment of the sample tilt, the raw data are converted to (q_x, q_z) automatically (see Fig. 3). From initial parameters given by the users, such as initial height, linewidth and sidewall angle, the minimization algorithm runs until the fit converges, and returns the fitted parameters as well as the accuracy of the fit.

In demonstration, silicon line gratings were studied. Line gratings were designed as dense arrays of line/space patterns with a constant periodicity of 86 nm and a line width of 40 nm. From the experimental profile obtained from -60° to 18°, with 0.5° step, line profiles were extracted from a model of seven stacked trapezoids, illustrated in Fig. 3. From the precision of the model, uncertainty was calculated; the fitted model was confirmed by critical-dimension scanning electron microscopy.

4. GIS(W)AXS toolset

4.1. GIWAXS remeshing

Remeshing is necessary for an accurate representation of a grazing-incidence wide-angle X-ray scattering (GIWAXS) image with orthogonal reciprocal space axes. This transformation redistributes pixel intensities by a backwards geometric mapping, transforming the data into proper q-space. A missing wedge on the remapped image is a consequence of the reflection geometry (see Fig. 4) (Baker *et al.*, 2010; Jiang, 2015; Hexemer & Müller-Buschbaum, 2015). For high data-rate processing, a GPU version of this algorithm is also included in *Xi-cam*.

4.2. Crystal simulator

GIS(W)AXS is routinely employed to determine the nanoscopic crystalline structures in assembled thin films consisting of nanoparticles, phase-separated block-copolymers and their nanocomposites. With user inputs of lattice parameters, space group (or customized lattice construction), relative crystal orientation, materials properties and experimental parameters (incident angle, energy, *etc.*), the crystal simulator toolset calculates diffraction peak locations in terms of scattering angles on an area detector. When solving for the scattering angles that satisfy the Laue condition within the framework of the distorted-wave Born approximation, two diffraction channels (reflection and transmission) are expected (Jiang, 2015), a unique phenomenon of the grazing-angle scattering geometry. The expected diffraction peaks can be superimposed on top of the experiment scattering pattern for a quick exploration and indexing of three-dimensional nanostructures of thin films (see Fig. 5). For a complete description of the technical approach, see Jiang (2015).

4.3. *HipGISAXS*

HipGISAXS is a massively parallel GISAXS simulator, based on the distorted wave born approximation, developed at the Lawrence Berkeley Laboratory (Chourou *et al.*, 2013). It can scale from everyday laptops to multi-CPU and multi-GPU clusters [for performance metrics, see Chourou *et al.* (2013)].

HipGISAXS has a very large input deck, even for relatively simpler problems. It grows proportionally, as the complexity of the problem. *Xi-cam*'s flexible and modular plugin infrastructure provides an excellent way to construct an intuitive user interface for running *HipGISAXS* and viewing the output. It was the integration of *HipGISAXS* into *Xi-cam* that pushed its remote execution capabilities to the current state.

Fig. 6 shows a screenshot of the *HipGISAXS* interface. The left-hand panel has tools to create hierarchical structures,

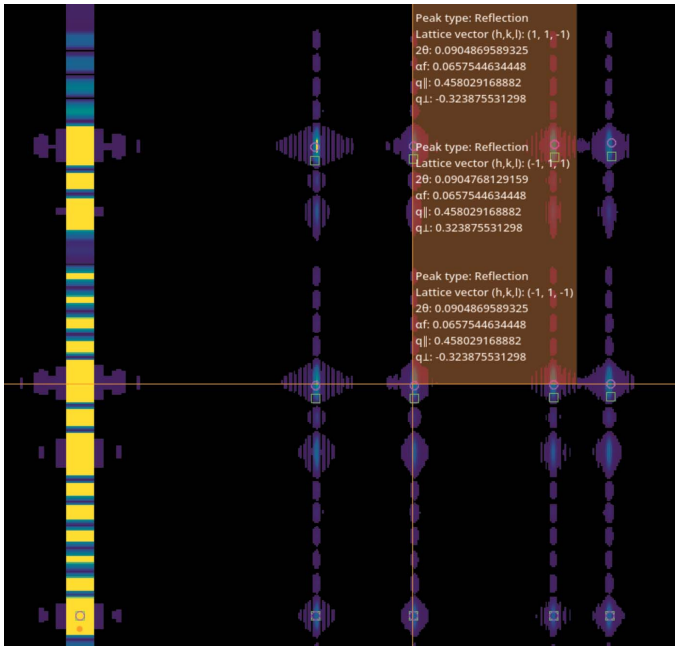

Figure 5
Screenshot of simulated data (cubic structure) with transmission (green/square) and reflection (magenta/circle) peaks overlayed. Hovering over a peak indicator displays the properties of the peak. The mouse has place the crosshair over the (1,1,1) reflection peak.

using analytical shapes such as spheres, cylinders, boxes, *etc*, and arranging the shapes into various geometric lattices (see Fig. 6). The right-hand panel has user input parameters to modify the structures. The center panel displays a 3D representation of structure. Once the user is satisfied with the input structure, and beam parameters, they can run the simulator locally or on any remote machine they can access using ssh. The scattering pattern is displayed in the center panel after the simulation is finished (see Fig. 7).

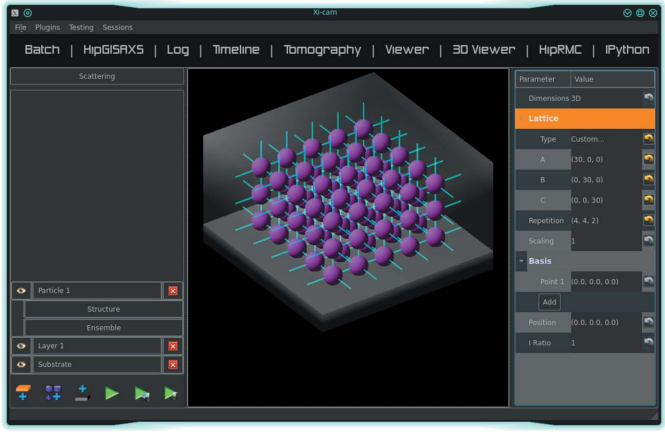

Figure 6
User interface for *HipGISAXS* in *Xi-cam* showing a cubic lattice of spherical particles embedded in a layer. The user may configure parameters for the scattering geometry, particle, structure, ensemble, layers and substrate. The green 'play' buttons run the described simulation.

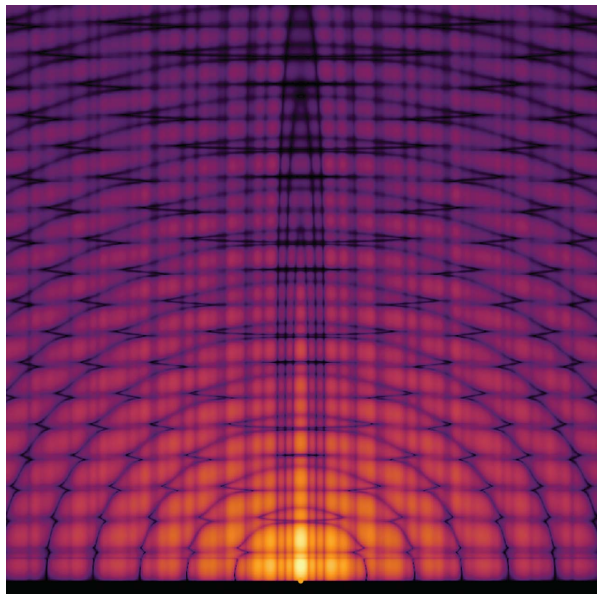

Figure 7
An example simulated GISAXS image generated from the material described in Fig. 6.

5. Tomography toolset

The *Xi-cam* tomography plugin provides an easy-to-use interface for viewing data, constructing workflows and performing reconstructions on raw tomography datasets. It offers a highly customizable workflow editor, as well as several features to precisely tailor this workflow to individual datasets. Features include quick reconstructions of individual sinograms and downsamplings of the full dataset. It also provides a built-in overlay visualization tool for manually detecting the centers of rotation when automatic center detections fail. An example of this feature is shown in Fig. 8. The combination of a customizable workflow editor and reconstruction previews allow users of varying levels of tomography expertise to find optimal pipelines for their unique datasets. Custom processing piplines can then be saved and subsequently used, either as-is or as a starting point, for other datasets.

The *Xi-cam* tomography plugin provides a graphical user interface supporting the full capabilities of TomoPy (Gürsoy *et al.*, 2014; Pelt *et al.*, 2016; Vogelgesang *et al.*, 2012; Pelt & Andrade, 2016; Pelt & Batenburg, 2015), an open-source Python library for tomographic data processing and image reconstruction. The plugin includes all required functionality for pre-processing, filtering and reconstructing datasets by exposing the algorithms of three popular tomographic processing libraries. For data processing and reconstructions, the plugin uses the ASTRA toolbox (Pelt *et al.*, 2016), a library of highly parallelized reconstruction algorithms, and TomoPy. Users may also insert their own custom functions. For reading and writing datasets, the plugin uses DXchange (De Carlo *et al.*, 2014). Data must be provided in either HDF5 or tif formats.

The left side of the GUI window contains both the workflow editor and the filebrowser (Fig. 9). Users open datasets either

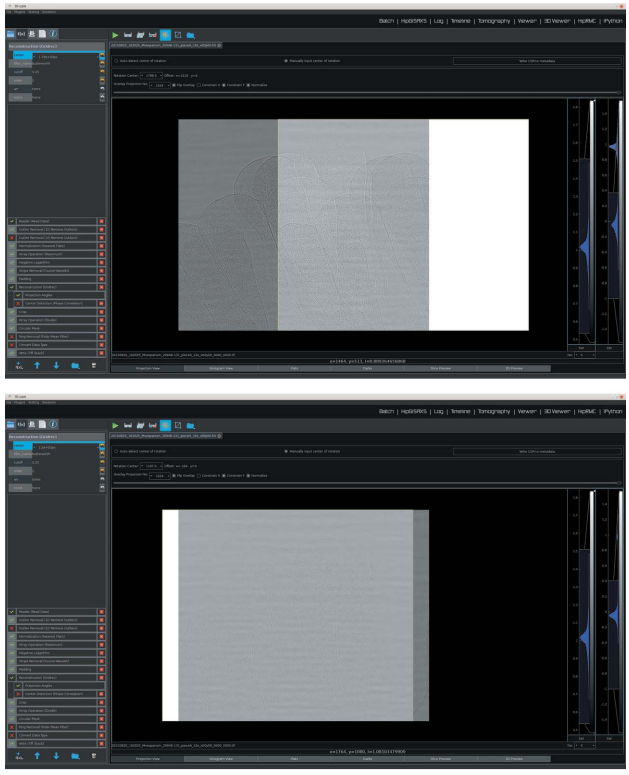

Figure 8
Plugin interface showing the built-in overlay visualization for finding centers of rotation. The tool overlays the first projection in the dataset with a mirror image of the last projection (top). By moving one of the projections, the user can match features from both projections (bottom). The plugin uses the relative horizontal displacement of both images to calculate the center of rotation of the dataset.

by double-clicking the desired dataset in the browser or by dragging and dropping the file into the central workspace. Upon loading the dataset, the raw projections and sinograms (Fig. 10) are immediately viewable in the center window. The toolbar at the top of the plugin (Fig. 11) contains buttons for reconstruction previews. The third-to-last button on the right enables the center of rotation overlay tool. Fig. 12 shows a single slice reconstruction preview using the workflow displayed on the left side of the GUI. Fig. 13 shows a three-dimensional reconstruction preview of a downsampled version of the dataset. By using the workflow editor, users can find the optimal workflow for their datasets and reconstruct the entire dataset. A single slice of this full reconstruction is shown in Fig. 14.

The standard use case requires minimal or no user input, as the plugin comes with a default processing workflow. More advanced users can tailor the workflow to their specific datasets by adding new or custom functions, changing function parameters, or changing the relative order of the functions. In addition, the plugin provides a tool to handle problematic parameters, such as the center of rotation or filter-specific parameters: by right-clicking a parameter in the workflow editor, users may provide a range of values for the parameter to be used in a series of single-sinogram preview reconstructions.

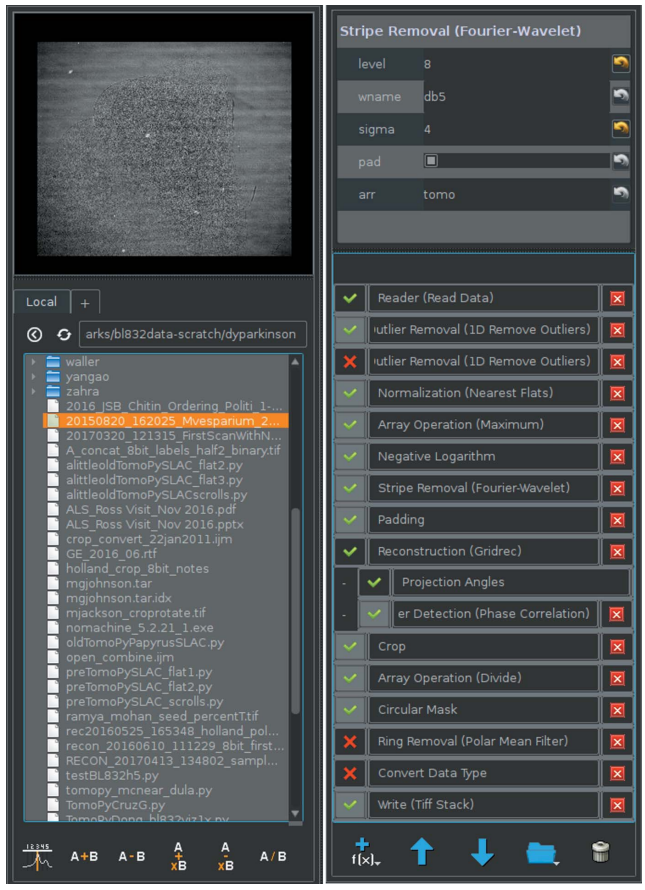

Figure 9
Filebrowser on the left GUI window (left). Workflow editor on the left GUI window with default pipeline (right).

6. XAS toolset

X-ray absorption spectroscopy (XAS) is defined as the fine structure of absorption spectra associated with inner shell excitation by different energy X-rays or electrons. The technique is described by many other acronyms including EXAFS (extended X-ray absorption fine structure), XAFS (X-ray

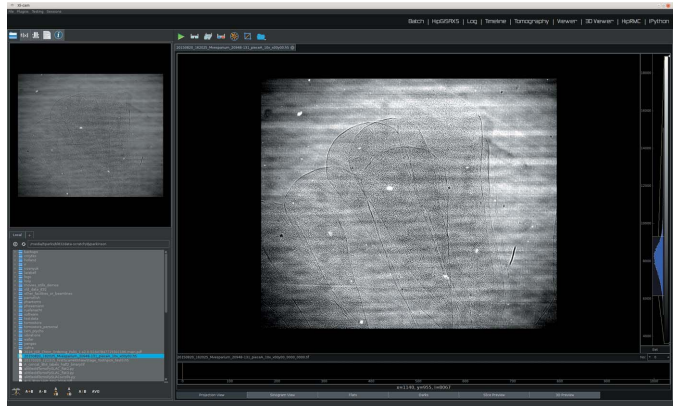

Figure 10
Plugin interface with workflow on the left and the raw projection data on the right. Users can use 'Timeline' below the image to look through projections or sinograms.

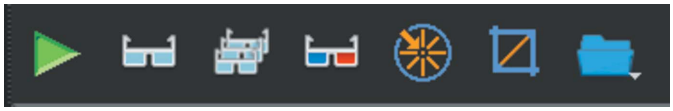

Figure 11
Toolbar at the top of the plugin window. Buttons are for, in order: full reconstructions, single-sinogram previews, multiple-sinogram previews, downsampled three-dimensional previews, enabling the center of rotation detection tool, cropping the dataset, and loading new flats/darks for normalization.

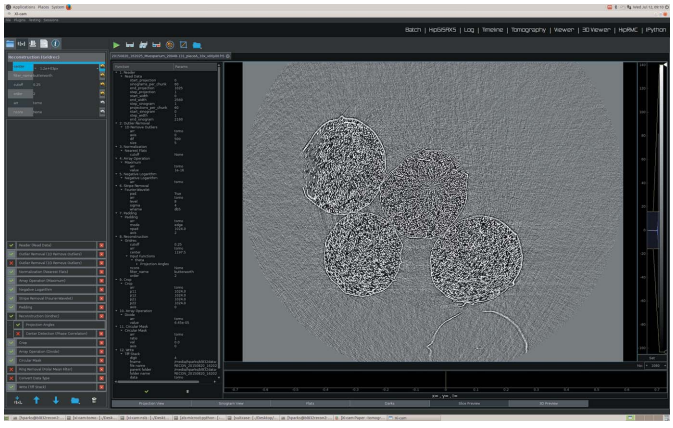

Figure 12
Plugin interface with a single slice reconstruction preview (right side).

absorption fine structure), XANES (X-ray absorption near-edge spectroscopy) and NEXAFS (near-edge X-ray absorption fine structure). The spectra are typically measured at synchrotron facilities and the community has seen considerable growth in the past decades with the increasing availability of synchrotron radiation worldwide. One issue facing the XAS community is choosing which software to use to analyze the XAS data. GUIs include *Sixpack*, *Athena* and *Artemis* which are all based on the *IFEFFIT* code base developed by Matt Newville and Bruce Ravel. Other efforts include a GUI based in the *IGOR Pro* language (*XMCD Panel*, Anders Glans), *LabVIEW*-based program (Matthew Marcus, Advanced Light Source), and multiple efforts at the Stanford Synchrotron Radiation Light Source (SSRL) including *XAS-Collect* (Martin J. George) and *EXAFSPAK* (Graham George). Alternatively, a researcher can use any programming language like MATLAB, Mathematica or Python to carry out the calculations themselves. The purpose of providing yet another XAS analysis tool is to streamline and unify XAS measurement, analysis, interpretation and figure-making into a single interface. To this end, newcomers to the field have a clear choice in software. For example, beamline control is currently operated by *LabVIEW* or *spec* at many beamlines. The data are exported to .csv files and then imported into the desired analysis program for normalization and possibly fitting. Modeling and producing journal-quality figures would be accomplished using two other pieces of software with likely different computational resources (*e.g.* supercomputer).

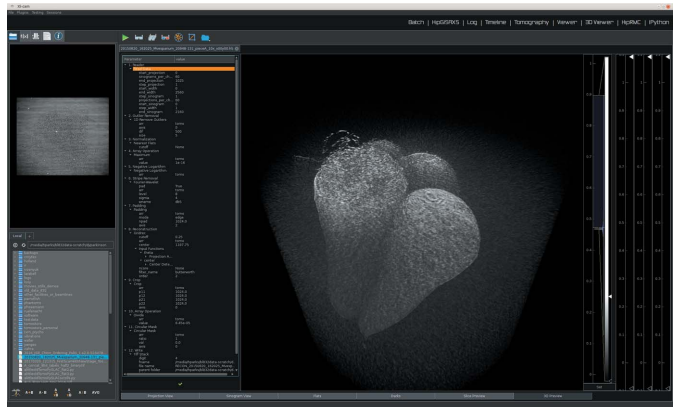

Figure 13
Plugin interface with a three-dimensional reconstruction preview of the dataset shown in Fig. 12.

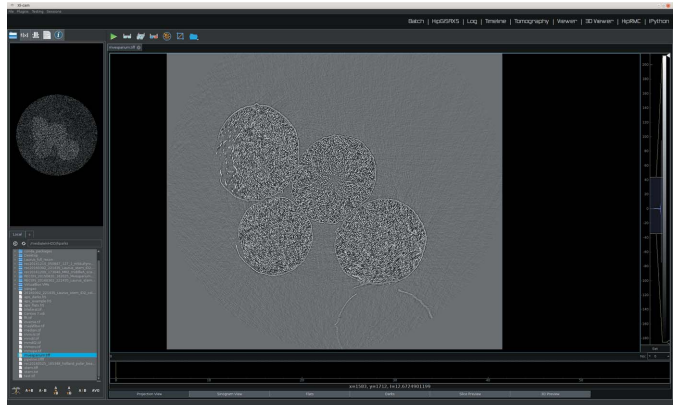

Figure 14
Full reconstruction of dataset previewed in Fig. 12 and Fig. 13.

Popular community supported code bases developed in Python can support all these features in a seamless work-flow.

The novelty of such a Python-based XAS plugin capable of so many functionalities is that a researcher can see commands 'under the hood' and directly manipulate data correspondingly. This capability combines the small learning curve of a GUI with the flexibility of having access to the command line. In addition, the thriving Python community is a powerful resource to leverage for rapid development, incorporation and testing of new or previously developed functionality.

The *Xi-cam* XAS plugin uses Larch (Newville, 2013), an evolution of the trusted *IFEFFIT* codebase being developed in Python by Matt Newville. The beamline system control uses the Experimental Physics and Industrial Control System (EPICS) to automate measurements for high-throughput XAS experimentation.

Data input supports .csv files with headers from beamlines 6.3.1, 6.3.2 and 11.0.1.2 at the Advanced Light Source. The requirements for basic functionality are a column for incident X-ray energy, initial beam intensity (I0) and absorption value (Channeltron, TEY). Parameters include automatic or manual setting of white-line energy, energy for unit normalization and two pre-edge values for leveling.

7. Batch data processing toolset (PAWS)

Xi-cam includes a 'Batch' plugin for applying a variety of data processing workflows to batches of files selected from the application's filesystem browser. *Xi-cam*'s batch processing workflows are built on PAWS (the Platform for Automated Workflows by SSRL).

PAWS is an abstraction layer for unifying lower-level Python APIs for data processing, from very mature packages (*e.g.* the scipy stack) to in-house scripts written for specific applications. These lower-level APIs are wrapped into PAWS Operations (for objects or functions that are only called upon when a workflow is executed) or PAWS Plugins (for objects that should persist for multiple workflow executions or be used concurrently by multiple workflows). Using the PAWS API, the Operations and Plugins are assembled into a PAWS Workflow. The Workflow is a Python object containing input data, any number of input parameters, and all of the functions and objects needed to produce relevant output data. *Xi-cam*'s workflows are written such that they can be copied or serialized, so that the (potentially laborious) batch processing can be run on remote machines or in parallel threads on a local machine.

As an ongoing development, the functionalities of all of the plugins described above are being adapted to PAWS workflows that can be executed on batches of files from the 'Batch' plugin.

The following example illustrates the use of the 'Batch' plugin to process a set of SAXS spectra. In §3.1 and Fig. 1, we saw how a SAXS geometry can be calibrated in the *Xi-cam* 'Viewer' plugin. After performing this process, the calibration parameters become immediately available to the other plugins. By switching to 'Batch', the calibration parameters can be used to integrate any number of SAXS spectra that share the same geometry. The procedure is as follows:

(i) In the 'Viewer' plugin frame, load a calibrant image (*e.g.* a silver behenate diffraction pattern).

(ii) See §3.1 to solve the SAXS beam–sample–detector geometry from this calibrant image.

(iii) Switch to the 'Batch' plugin frame.

(iv) Select a list of files from the application's filesystem browser. Open the files to copy the list of file paths to the 'Batch' input files list.

(v) Select the 'SAXS integrator' workflow from the 'Batch' workflow selector.

(vi) Click the 'Run' button. The 'SAXS integrator' workflow will read the same calibration parameters that were generated in steps (i)–(ii) above.

(vii) As the batch is processed, browse the results by clicking through the Workflow viewer. Clicking an Operation output in the Workflow viewer causes PAWS to generate a widget that visualizes the output data. Data-appropriate widgets are generated to display objects as simple as strings or integers (text printouts), or as complex as multidimensional arrays (interactive plot frames).

(viii) Adjust the settings of the Workflow by entering values into the Workflow viewer. In this way the user can control most or all details of the Workflow (*e.g.* limits in q-space, q-resolution and output filenames, to name a few).

(ix) After finding workable settings, enable the output Operations so that the integrated results will be saved to the filesystem. Adjust the settings of the output Operations in the Workflow viewer, to control the output directory, file type, file name, data column headers, delimiters, *etc*.

(x) Run the Workflow again, and watch the output files appear on the filesystem as the Workflow runs.

Since batch processing is likely to generate large amounts of data, *Xi-cam*'s workflows are written to save nothing. The outputs can be visualized as they are computed, but after this they are de-referenced to free up memory. The end point of 'Batch' workflows is generally to produce a file containing the results. In the example above, a set of diffraction images (.tif files, for example) produces integrated diffraction data (intensity *versus* q), and the integrated data can be inspected directly and/or saved to a .csv file.

To illustrate the versatility of the batch processing plugin, consider taking the above procedure one step further. After integrating the SAXS spectra to a set of .csv files containing intensity *versus* q, select this batch of .csv files and open them for input for another batch-processing workflow. Select 'saxs guinier-porod fitter' from the workflow selector, and run the workflow. The integrated spectra are read in from the .csv files and fit to a Guinier–Porod scattering equation (Hammouda, 2010) with a noise floor [*via* the *XRSDKit* package (XRSDKit Development Team, 2017)]. The Guinier–Porod fit spectra can be written out in a separate .csv file, and the user can compare the measured data *versus* the Guinier–Porod fit.

8. Discussions and conclusions

Xi-cam provides a high-level GUI platform for users to manage, visualize and reduce synchrotron data from scattering, tomography and NEXAFS beamlines. From the user perspective, *Xi-cam*'s development emphasizes interactive data exploration, robust new algorithms and reactive design. While a deeper, flexible functionality is also provided, users require no programming or algorithmic knowledge to process large data. Synchrotron users can employ *Xi-cam*'s plugin-based framework to treat data from any facility, beamline and technique in this single application. A variety of unique analysis algorithms and visualizations are also included, providing users with new approches to looking at their data. Only minimal hardware is required; however, some processing components scale well to multi-core/multi-GPU architectures, with the availability of remote computing.

From the perspective of a scientific software developer, *Xi-cam* provides a means to improve exposure of algorithmic code across user facilities in a way that is agnostic to platform, data format and instrument. As a plugin-based platform, *Xi-cam* is easily extensible. *Xi-cam*'s rapid prototyping tools allow even novice developers to present a user-friendly interface on top of their reduction/treatment algorithm. Naturally, this model supports open source, collaborative and consolidated development efforts; continuing development of

Xi-cam will also emphasize these ideas. Further techniques and algorithms are continuously being added to the *Xi-cam* framework.

Acknowledgements

The example data shown was acquired at beamlines 7.3.3 (Hexemer *et al.*, 2010) and 8.3.2 (MacDowell *et al.*, 2012) at the Advanced Light Source (ALS), a Division of Lawrence Berkeley National Laboratory.

Funding information

Funding for this research was provided by: Lawrence Berkeley National Laboratory (grant No. TReXS LDRD to AH); US Department of Energy (award No. Early Career Award to AH; contract No. DE-SC0012704; contract No. DE-AC02-06CH11357; contract No. DE-AC02-76SF00515; contract No. DE-AC02-05CH11231); Center for Advanced Mathematics in Energy Research Applications.

References

Ashiotis, G., Deschildre, A., Nawaz, Z., Wright, J. P., Karkoulis, D., Picca, F. E. & Kieffer, J. (2015). *J. Appl. Cryst.* **48**, 510–519.

Ave, S. E., Chard, K., Foster, I. & Tuecke, S. (2017). *Proceedings of the Practice and Experience in Advanced Research Computing 2017 on Sustainability, Success and Impact Article*, pp. 26:1–26:5, New Orleans, USA.

Baker, J. L., Jimison, L. H., Mannsfeld, S., Volkman, S., Yin, S., Subramanian, V., Salleo, A., Alivisatos, a, P. & Toney, M. F. (2010). *Langmuir*, **26**, 9146–9151.

Benecke, G., Wagermaier, W., Li, C., Schwartzkopf, M., Flucke, G., Hoerth, R., Zizak, I., Burghammer, M., Metwalli, E., Müller-Buschbaum, P., Trebbin, M., Förster, S., Paris, O., Roth, S. V. & Fratzl, P. (2014). *J. Appl. Cryst.* **47**, 1797–1803.

Carmona, R., Hwang, W. & Torrésani, B. (1998). *Practical Time-Frequency Analysis: Gabor and Wavelet Transforms with an Implementation in S. Wavelet Analysis and Its Applications Series*. Academic Press.

Chourou, S. T., Sarje, A., Li, X. S., Chan, E. R. & Hexemer, A. (2013). *J. Appl. Cryst.* **46**, 1781–1795.

Dask Development Team (2016). *Dask: Library for dynamic task scheduling*, http://dask.pydata.org.

Databroker Development Team (2017). *Databroker*, https://github.com/NSLS-II/databroker.

Daubechies, I. (1992). *Ten Lectures on Wavelets.* Philadelphia: Society for Industrial and Applied Mathematics.

De Carlo, F., Gürsoy, D., Marone, F., Rivers, M., Parkinson, D. Y., Khan, F., Schwarz, N., Vine, D. J., Vogt, S., Gleber, S.-C., Narayanan, S., Newville, M., Lanzirotti, T., Sun, Y., Hong, Y. P. & Jacobsen, C. (2014). *J. Synchrotron Rad.* **21**, 1224–1230.

Du, P., Kibbe, W. A. & Lin, S. M. (2006). *Bioinformatics*, **22**, 2059–2065.

Durniak, C., Ganeva, M., Pospelov, G. & Van Herck, W., (2015). *BornAgain – Software for simulating and fitting X-ray and neutron small-angle scattering at grazing incidence*, http://www.bornagain project.org.

Franke, D., Petoukhov, M. V., Konarev, P. V., Panjkovich, A., Tuukkanen, A., Mertens, H. D. T., Kikhney, A. G., Hajizadeh, N. R., Franklin, J. M., Jeffries, C. M. & Svergun, D. I. (2017). *J. Appl. Cryst.* **50**, 1212–1225.

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

Hammersley, A. P. (2016). *J. Appl. Cryst.* **49**, 646–652.

Hammouda, B. (2010). *J. Appl. Cryst.* **43**, 716–719.

Hannon, A. F., Sunday, D. F., Windover, D. & Kline, R. J. (2016). *J. Micro. Nanolithogr. MEMS MOEMS*, **15**, 034001.

Hexemer, A., Bras, W., Glossinger, J., Schaible, E., Gann, E., Kirian, R., MacDowell, A., Church, M., Rude, B. & Padmore, H. (2010). *J. Phys. Conf. Ser.* **247**, 012007.

Hexemer, A. & Müller-Buschbaum, P. (2015). *IUCrJ*, **2**, 106–125.

Hu, T., Jones, R. L., Wu, W. L., Lin, E. K., Lin, Q., Keane, D., Weigand, S. & Quintana, J. (2004). *J. Appl. Phys.* **96**, 1983–1987.

Ilavsky, J. (2012). *J. Appl. Cryst.* **45**, 324–328.

Jiang, Z. (2015). *J. Appl. Cryst.* **48**, 917–926.

Knudsen, E. B., Sørensen, H. O., Wright, J. P., Goret, G. & Kieffer, J. (2013). *J. Appl. Cryst.* **46**, 537–539.

Könnecke, M., Akeroyd, F. A., Bernstein, H. J., Brewster, A. S., Campbell, S. I., Clausen, B., Cottrell, S., Hoffmann, J. U., Jemian, P. R., Männicke, D., Osborn, R., Peterson, P. F., Richter, T., Suzuki, J., Watts, B., Wintersberger, E. & Wuttke, J. (2015). *J. Appl. Cryst.* **48**, 301–305.

Lazzari, R. (2002). *J. Appl. Cryst.* **35**, 406–421.

Liu, J., Lhermitte, J., Tian, Y., Zhang, Z., Yu, D. & Yager, K. G. (2017). *IUCrJ*, **4**, 455–465.

MacDowell, A. A., Parkinson, D. Y., Haboub, A., Schaible, E., Nasiatka, J. R., Yee, C. A., Jameson, J. R., Ajo-Franklin, J. B., Brodersen, C. R. & McElrone, A. J. (2012). *Proc. SPIE*, **8506**, 850618.

Newville, M. (2013). *J. Phys. Conf. Ser.* **430**, 012007.

Paramiko Development Team (2017). *Paramiko*, http://www.paramikoorg/.

Pelt, D. M. & Andrade, V. D. (2016). *Adv. Struct. Chem. Imaging*, **2**, 17.

Pelt, D. M. & Batenburg, K. J. (2015). *Proceedings of the 13th International Meeting on Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine*, pp. 158–161.

Pelt, D. M., Gürsoy, D., Palenstijn, W. J., Sijbers, J., De Carlo, F. & Batenburg, K. J. (2016). *J. Synchrotron Rad.* **23**, 842–849.

SPOT Development Team (2017). *SPOT*, http://spot.nersc.gov/index.php.

Sunday, D. F., List, S., Chawla, J. S. & Kline, R. J. (2015). *J. Appl. Cryst.* **48**, 1355–1363.

Robitaille, T. P., Tollerud, E. J., Greenfield, P., Droettboom, M., Bray, E., Aldcroft, T., Davis, M., Ginsburg, A., Price-Whelan, A. M., Kerzendorf, W. E., Conley, A., Crighton, N., Barbary, K., Muna, D., Ferguson, H., Grollier, F., Parikh, M. M., Nair, P. H., Günther, H. M., Deil, C., Woillez, J., Conseil, S., Kramer, R., Turner, J. E. H., Singer, L., Fox, R., Weaver, B. A., Zabalza, V., Edwards, Z. I., Bostroem, K. A., Burke, D. J., Casey, A. R., Crawford, S. M., Dencheva, N., Ely, J., Jenness, T., Labrie, K., Lim, P. L., Pierfederici, F., Pontzen, A., Ptak, A., Refsdal, B., Servillat, M. & Streicher, O. (2013). *Astron. Astrophys.* **558**, A33.

Vogelgesang, M., Chilingaryan, S., Rolo, T. D. S. & Kopmann, A. (2012). *Proceedings of the 14th IEEE International Conference on High Performance Computing and Communications (HPCC-2012) and the 9th IEEE International Conference on Embedded Software and Systems (ICESS-2012)*, pp. 824–829.

Wang, C., Fu, W. E., Li, B., Huang, H., Soles, C., Lin, E. K., Wu, W., Ho, P. S. & Cresswell, M. W. (2009). *Thin Solid Films*, **517**, 5844–5847.

XRSDKit Development Team (2017). *XRSDKit*, https://github.com/scattering-central/xrsdkit.

Yager, K. G., Zhang, Y., Lu, F. & Gang, O. (2014). *J. Appl. Cryst.* **47**, 118–129.

Yang, L. (2013). *J. Synchrotron Rad.* **20**, 211–218.