Documentation for GIXSGUI - 1.6.2

Zhang Jiang
X-ray Science Division
Advanced Photon Source
Argonne National Laboratory
zjiang@aps.anl.gov
(Dated: May 31, 2015)

CONTENTS

I. OVERVIEW

GIXSGUI (Grazing-incidence X-ray Scattering Graphical User Interface) is a Matlab toolbox that offers both a graphical user interface and a script-based access to visualize and process grazing-incidence X-ray scattering (GIXS) data from nano-structures on surfaces and in thin films. It provides routine surface scattering data reduction methods such as geometric correction, one-dimensional intensity linecut, two-dimensional intensity reshaping, etc. Three-dimensional structure indexing is also implemented to determine the space group and lattice parameters of organized nano-structures in thin films.

While both transmission and reflection geometries are supported, the package is more focused on handling grazing-incidence data. This package was originally designed for the GIXS beam line at Sector 8-ID-E, the Advanced Photon Source, Argonne National Laboratory, but It should also be compatible with GIXS data taken at other synchrotron facilities and laboratory-based instruments.

Currently supported image formats include TIF (TIFF), CBF (Crystallographic Binary File), EDF (European Data Format), FITS (Flexible Image Transport System), and MAT (Matlab).

GIXSGUI toolbox contains two main *.m files: *gixsgui.m* and *gixsdata.m*. The later defines a handle class `gixsdata`. Data and experiment parameters are stored as properties in `gixsdata`, and most data operations are achieved via built-in methods of `gixsdata`. In addition to GUI mode, this design also allows a script mode fully independent of GUI, enabling batch processing and data automation for large data set that often result from in-situ and time-resolved experiments.

It is appreciated that any publication resulting from this toolbox acknowledges its use by citing the following article — Z. Jiang, *GIXSGUI*: a MATLAB toolbox for grazing-incidence X-ray scattering data visualization and reduction, and indexing of buried three-dimensional periodic nanostructured films, ***J. Appl. Crystallogr.*** 48, 917-926 (2015). DOI:10.1107/S1600576715004434

II. INSTALLATION

Matlab 2010a or later versions are required. At least 2GB memory is recommended due to a number of angle and q maps created during data processing for each loaded image file. To install GIXSGUI, execute the unzip the package into your desired folder and add that folder and all of its **subfolders** to Matlab path (Matlab =≫ File =≫ Set Path =≫ Add with Subfolders).

III. GIXSDATA AND SCRIPT MODE

`gixsdata` is a handle class defined in file *gixsdata.m*. Image data, setup and plot parameters, and operation methods are defined in the class. Its properties and methods are listed in appendices A and B. Figure 1 illustrates the data flow of a `gixsdata` object and *GIXSGUI*.

A. Load image

To load an image file, e.g., `demo.tif`,

```
obj=gixsdata('demo.tif');
```

loads the image data and passes it to property `obj.RawData`. `ImFile`, `ImFilePath`, `ImFileName`, `ImFileExt`, and `ImFileInfo` are updated automatically. If `obj` already exists, one can also use

```
obj.ImFile='demo.tif';
```

Alternatively, one can load the image into Matlab workspace first and pass the image to `RawData` to an existing object, e.g.,

```
data=imread('demo.tif');
obj=gixsdata;
obj.RawData=data;
```

which is equivalent to

```
obj=gixsdata(data);
```

Be cautious that `ImFile`, `ImFilePath`, `ImFileName`, `ImFileExt`, and `ImFileInfo` are not updated. The advantage of using this method is that setup and plot parameters, q and angle maps are all preserved, which may save a significant amount of time of recalculating maps if the newly loaded `RawData` has identical dimensions and setup parameters.

After `RawData`, an array of 1043(V)×981(H) for Pilatus 1MF for example, is assigned, several operations are performed automatically in sequence as listed below.

- `ImDim` is set to the dimension of `RawData`, [981,1043].
- `Mask` is initialized as a logical array with value 1 assigned to elements with non-negative (≥ 0) intensities for `RawData`, and 0 for the rest.
- `RawData` is duplicated to `MaskedData` with masked region assigned to NaN.
- If `FlatField` is empty or its dimension does not match `RawData`, a default array of all ones is assigned. It then automatically applies to `MaskedData`. It does not apply to `SolidAngleCorrectedData` at this step.
- If q maps, angle maps, and 1D lists are empty or their dimensions do not match `RawData`, method `qmaps` is called to create new maps and lists. However, this operation is not performed if any of the necessary setup parameters, e.g., `Beam0`, `XEnergy`, etc., are invalid. Method `solidangle_correction` is then called by `qmaps` at the end.
- Given valid setup parameters, e.g., `Beam0` and `XEnergy`, method `solidangle_correction` is called for solid angle, flat field, efficiency, and polarization corrections. The corrected data is stored in `SolidAngleCorrectedData`.

B. Q and angle maps

The convention of q and angles are illustrated in Fig. 2.

- $\boldsymbol{k}_i$ and $\boldsymbol{k}_f$: Incident and exit wave vectors with $|\boldsymbol{k}_i| = |\boldsymbol{k}_f| = 2\pi/\lambda$ in elastic x-ray scattering, where λ is the x-ray wave wave length.
- 2Θ: Oblique angle between $\boldsymbol{k}_i$ and $\boldsymbol{k}_f$.
- $\boldsymbol{q}$: Total wave vector transfer, $\boldsymbol{q} = \boldsymbol{k}_f - \boldsymbol{k}_i$. For reflection geometry, $q = \sqrt{q_z^2 + q_x^2 + q_y^2}$.
- Φ: Polar angle (or azimuthal angle) on the detector plane with respect to the direct beam pixel on detector. Its range can be either $[-180°, 180°)$ or $[0°, 360°)$.
- χ: Polar angle of the reciprocal space with respect to the surface normal, i.e., the angle between q_z direction and $\boldsymbol{q}$.
- α_i and α_f: Incident and exit angles.
- 2θ: In-plane (parallel to surface) scattering angle.
- q_z: Wave vector transfer component normal to the surface, $q_z = k_f \sin\alpha_f + k_i \sin\alpha_i$.
- q_x: In-plane wave vector transfer component in the scattering plane (i.e. the zx plane), $q_x = k_f \cos\alpha_f \cos 2\theta - k_i \cos\alpha_i$.

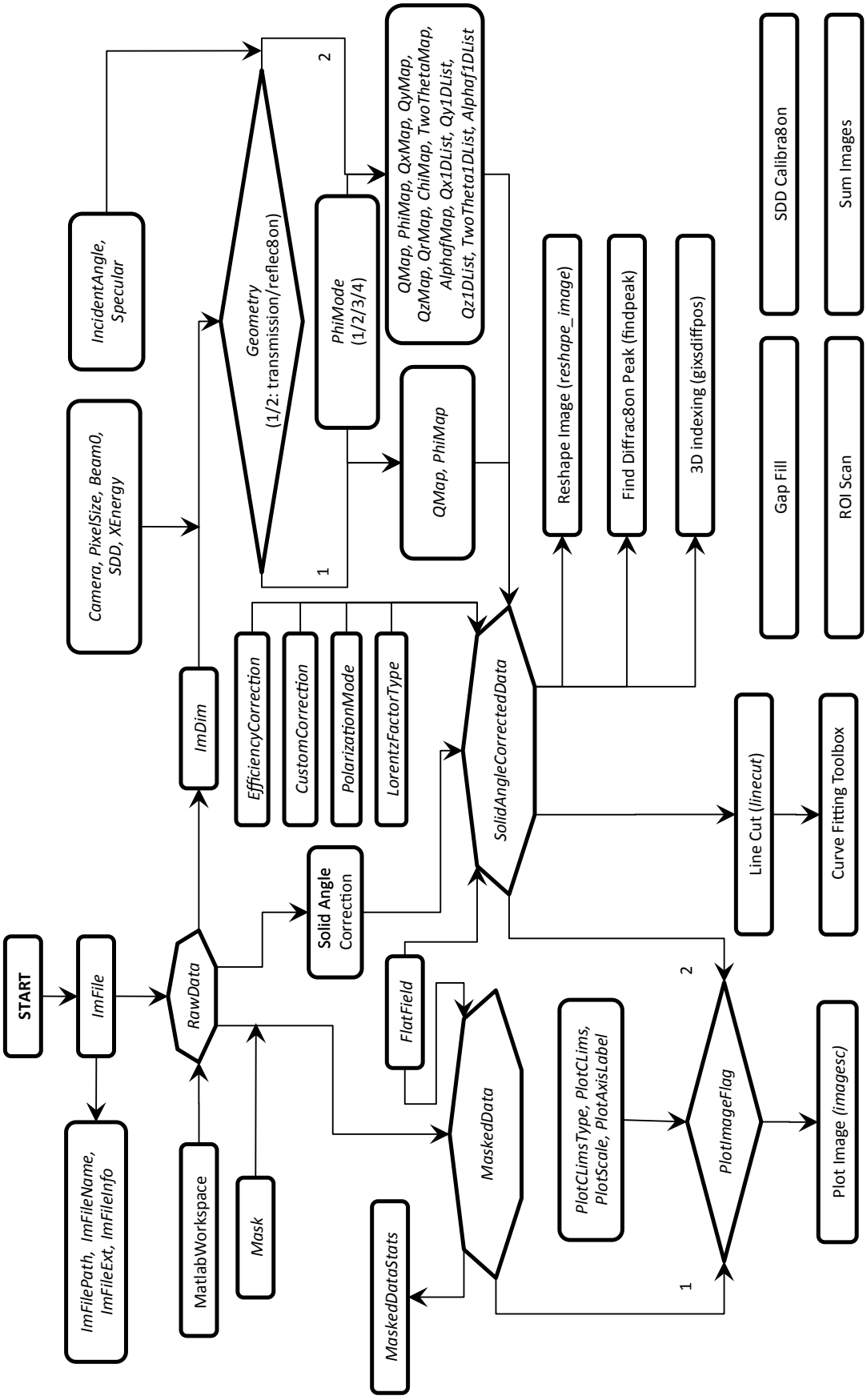

FIG. 1. Data flow in a **gixsdata** object and in *GIXSGUI*.

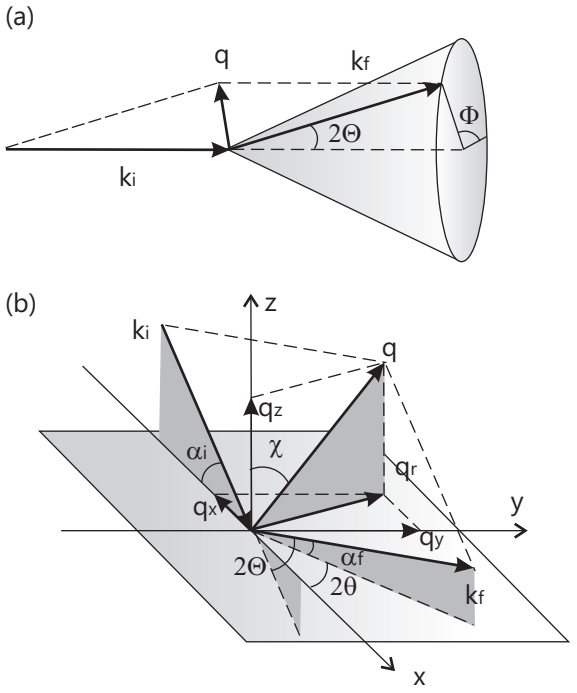

FIG. 2. (a) Transmission geometry. (b) Reflection geometry in the sample surface reference frame.

- q_y: In-plane wave vector transfer component normal to the scattering plane, $q_y = k_f \cos\alpha_f \sin 2\theta$.
- q_r: Total in-plane wave vector transfer, $q_r = \sqrt{q_x^2 + q_y^2}$.

Maps are calculated automatically if all necessary setup parameters exist before `RawData` is assigned. Otherwise, one has to call `qmaps(obj)` manually. All maps have identical dimension to `RawData`. Method `solidangle_correction` is called within method `qmaps` in the end.

C. Corrections

GIXSGUI supports typical GIXS experimental data in which a 2D area detector is fixed at a stationary position with the direct incident beam impinging normal to the detector face. Obtaining the differential scattering cross section of the sample requires a number of data corrections on the 2D GIXS pattern to account for detection efficiency and flat-field, incident X-ray polarization, solid angle variation, etc.

1. *Efficiency correction*

Efficiency correction arises from two sources: medium (e.g. air) path attenuation and detector sensor absorption. Both corrections are especially crucial for

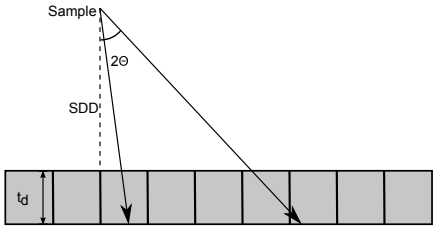

FIG. 3. Photons scattered to larger oblique angles are more attenuated in the path medium and have increased absorption probability in the detector sensor.

experiments with a ultra-short sample-to-detector distance, e.g., GIWAXS. Pixels at larger oblique angles, 2Θ, have larger sample-to-pixel distance, and therefore larger medium attenuation (Fig. 3). This effect can be corrected by multiplying the measured intensity by

$$E_m = \frac{1}{\exp\left[-\mu_m \mathrm{SDD}/\cos 2\Theta_i\right]}, \tag{1}$$

in a pixel-by-pixel manner, where μ_m (unit: 1/mm) is the linear mass attenuation coefficient of the path medium, `SDD` (unit: mm) is the nominal sample-to-detector distance with respect to the direct beam pixel, and $2\Theta_i$ is the oblique scattering angle of pixel (x_i, y_i) given by

$$2\Theta_i = \mathrm{atan}\left[\sqrt{L_x^2(x_i - x_0)^2 + L_y^2(y_i - y_0)^2}/\mathrm{SDD}\right], \tag{2}$$

where L_x and L_y are horizontal and vertical pixel sizes, x_0 and y_0 are direct beam position on detector. E_m is unity for measurements conducted in a vacuum environment all the way from sample to detector. However, the vacuum setup is usually inconvenient for GIWAXS, or may be incompatible for *in-situ* investigations.

On the contrary, the detector sensor absorption correction is due to the variation of X-ray absorption probabilities across pixels and depends on the properties of the detector. For direct-detection pixel-array detectors such as Pilatus (Dectris Inc.), X-rays impacting at different oblique angles result in different path lengths in the detector sensor. The probability of stopping and detecting X-ray photons increases as the angle increases. The correction is given by

$$E_d = \frac{1}{1 - \exp\left[-\mu_d t_d/\cos 2\Theta_i\right]} \tag{3}$$

where μ_d (unit: 1/mm) is the linear mass attenuation coefficient of the sensor material, and t_d (unit: mm) is the thickness of the sensor plate. The efficiency correction is optional depending on the experiment setups and detector types.

To set up efficiency correction parameters, one needs to construct a 2×3 cell for `EfficiencyCorrection` (defined in appendix A). The total efficiency correction array is calculated by multiplying E_m and E_d on a pixel-by-pixel basis, followed by a normalization to its maximum

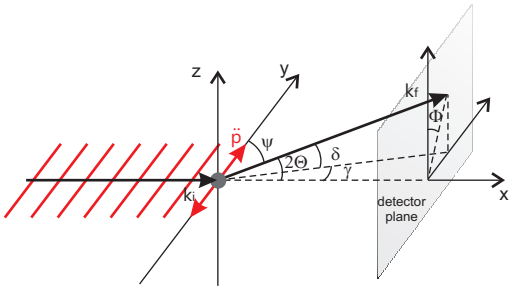

FIG. 4. Electrons in a sample become dipoles oscillating along the direction of the polarization of the incident wave, resulting in less observed scattered intensity at smaller oblique angles. The geometry is in the lab reference frame.

value. The correction then applies by a by multiplication to `RawData` in a pixel-by-pixel basis, and the corrected data is stored in `SolidAngleCorrectedData` (see section III C 4). It does not apply to `MaskedData`.

2. *Polarization correction*

The amplitude of the scattering from an electron in an electromagnetic field is proportional to the observed acceleration of its dipole moment. Synchrotron radiation is linearly polarized in the horizontal plane, i.e., the dipole moment is horizontal and perpendicular to the incident beam direction $\boldsymbol{k}_i$ (Fig. 4). The full acceleration of the electron would be only observed in the xz plane. However, out of the xz plane, only partial acceleration would be observed. Therefore, the amplitude of the observed scattering field reduces as ψ increases, where ψ is the angle between the polarization and the exit wave vector $\boldsymbol{k}_f$. The polarization correction factor for observed intensity is

$$P_h = |\sin\psi|^2 = 1 - \cos^2\delta\sin^2\gamma. \tag{4}$$

For a vertically polarized source, or when the area detector is rotated 90° or 270°, the polarization correction factor is

$$P_v = \cos^2\delta. \tag{5}$$

Vertical polarization correction usually occurs in the following situation: grazing-incidence takes place from a vertically mounted sample, the detector is mounted without rotation, and later during the data analysis, the image is rotated 90° or 270°. Therefore, choosing between the horizontal and vertical polarization corrections depends on the direction of the detector mount with respect to the polarization plane of the synchrotron radiation.

For a horizontally or vertically polarized source with a partial polarization, the correction factor is

$$P_p = \zeta P_h + (1-\zeta)P_v, \tag{6}$$

where $\zeta \in [0,1]$ is the fraction of the radiation that is polarized in the horizontal direction of the beam. For synchrotron radiations, ζ is $\sim 98\%$.

For an unpolarized source, the emitted electric field is calculated by considering the projections of the incident electric field onto the y and z axes separately. The polarization correction factor is then calculated from the average value of the two components so that,

$$\begin{aligned} P_u &= \frac{1}{2}\left(P_h + P_v\right) = \frac{1}{2}\left(1 + \cos^2\delta\cos^2\gamma\right) \\ &= \frac{1}{2}\left(1 + \cos^2 2\Theta\right). \end{aligned} \tag{7}$$

Therefore, the correction has nothing to do the rotation of the area detector. Rather, it only depends on the oblique angle Θ. This is an equivalent case for $\zeta = 0.5$.

There are four options provided: none, horizontal, vertical, and unpolarized. In GISAXS and SAXS experiments with very small δ and γ angles, the polarization correction is negligible. So the "none" option can be selected, i.e., the correction is one,

$$P_n = 1. \tag{8}$$

To perform the polarization correction, the scattering intensity is divided by the corresponding polarization factor on a pixel-by-pixel basis. The corrected data is superimposed on the efficiency correction and is stored in `SolidAngleCorrectedData` (see section III C 4). It does not apply to `MaskedData`.

3. *Flat field correction*

The flat field correction removes artifacts that are caused by variations in the pixel-to-pixel sensitivity of the detector. Both the `MaskedData` and `SolidAngleCorrectedData` are always multiplied by the flat field correction array which is a normalized 2D image pre-calibrated by the detector manufacture at the operation energy. One can assign the flat field data using

```
obj.FlatField = flatfield_data;
```

Figure 5 shows the effect of flat field correction for a Pilatus camera. After correction, higher-intensity lines due to gain variations at the edges of Pilatus modules and chips, as well as abnormal intensities on a number of biased pixels, disappear.

4. *Solid angle correction*

The measured intensity on a pixel is proportional to the solid angle subtended by that pixel and needs to be normalized with respect to a reference, e.g., solid angle of the direct beam pixel. Solid angle correction is purely

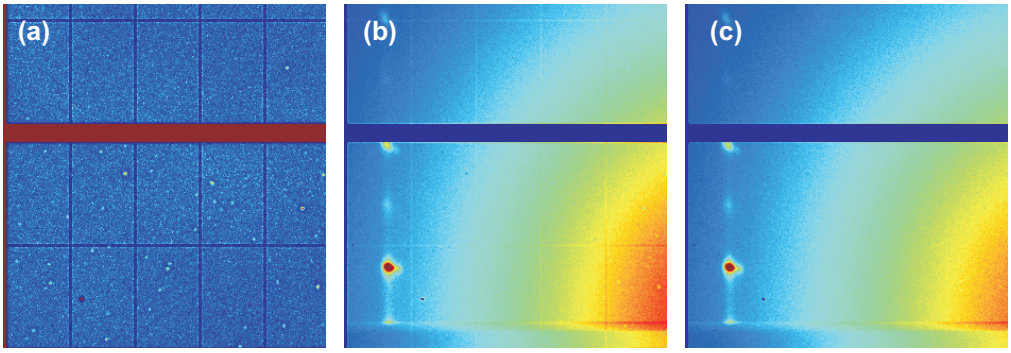

FIG. 5. (a) Flat field data of Pilatus 1MF camera. (b) Before correction. (c) After correction.

a geometrical effect that applies to all types of detectors and is given by

$$C_s = \Delta\Omega_0/\Delta\Omega_i, \tag{9}$$

which is multiplied to `RawData`. Here, $\Delta\Omega_i = L_x L_y/\cos 2\Theta_i$ is the solid angle of pixel (x_i, y_i), and $\Delta\Omega_0$ is the solid angle of the direct beam pixel. Given valid setup parameters, such as `SDD`, `Beam0`, and `PixelSize`, solid angle correction is performed automatically on `RawData`, and cannot be switched off. Flat field (if exists), efficiency and polarization corrections are also automatically performed at the end of method `solidangle_correction`. The corrected data is stored in `SolidAngleCorrectedData`. Solid angle correction does not apply to `MaskedData`.

5. *Lorentz correction*

Lorentz correction is usually used to correct the intensities of X-ray scattering of single crystal diffractometry in order to obtain the real intensities for structure factors. Many single crystal diffraction experiments have been and are performed by spinning the crystal along an axis. One needs to correct the differences in the time that the individual reciprocal lattice point, with different miller indices and therefore at different distance from the origin ([000], point O in Fig. 6), spends on the surface of the Ewald (i.e., Laue condition to give scattering intensities). Similar scenario applies to powder scattering where the orientation randomness of crystallites is in a plane. Lorentz correction should be applied only to scattering from the reciprocal lattices where there is a periodicity with a great number of consecutive repeating units in the crystalline system. Note that lorentz correction destroys the characteristics of the scattering of individual particles and should not be used to produce a peak on scattering data which does not show periodicity or structural peaks without correction [Cser, *J. Appl. Polym. Sci.* **80**, 2300 (2001)]. Apparently, the shape factor of a particle always peaks at zero scattering angle; however, incorrectly applying Lorentz correction gives faulty intensities because the correction reduces the intensities to zero at the zero angle and generates an artificial maximum somewhere at a non-zero angle.

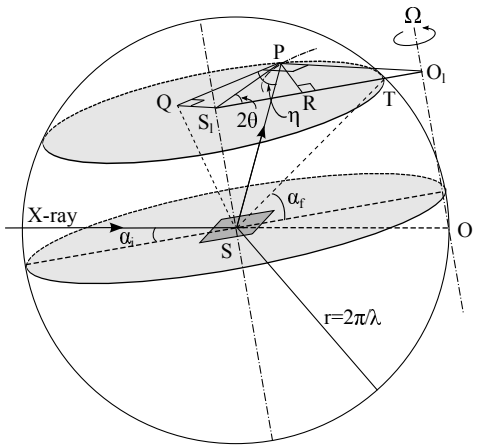

FIG. 6. Illustration of the origin of Lorentz correction with the help of Ewald sphere.

Lorentz correction is an experiment specific correction, and one should understand the sample conditions and experiment setup in order to properly choose Lorentz correction formula. One can calculate the Lorentz factor for the correction by deriving the Jacobian for the transformation of the integration volume (over the differential scattering cross section in order to obtain the total scattering intensity) from the angular expressions in real-space to the *dhdkdl* expressions in reciprocal space [Vlieg, *J. Appl. Cryst.* **30**, 532 (1997)]. Here, we use an alternative approach, i.e., Buerger precession method [Buerger, *Phys.* **26**, 637 (1940); Buerger and Klein, *J. Appl. Phys* **16**, 408 (1945); Waser, *Rev. Sci. Instrum.* **22**, 563 (1951)], to illustrate the origin of the Lorentz correction for an ω scan, where a film sample of 3D structures rotates along the axis normal to its surface (Fig. 6). The Lorentz factor is proportional to the time of diffraction permitted to each diffraction, or inversely proportional to the velocity with which the diffraction passes through the Ewald sphere. For reflection condition $[hkl]$, Lorentz

factor is

$$L_{hkl} \sim \frac{1}{V_{hkl}}. \tag{10}$$

Here, $V_{hkl} = \Omega\overline{PO_l}\cos\eta$ is the velocity of the allowed reflection P to pass through the surface of the Ewald sphere, where Ω is the angular velocity of the rotation with respect to the $\overline{OO_l}$ axis that is normal to sample surface, and η is the angle between the velocity tangential direction of reflection P and $\overline{SP}$. Because $\cos\eta = \overline{PQ}/\overline{SP} = \overline{PQ}/r$, $\overline{PQ}\ \overline{PO_l} = \overline{PR}\ \overline{S_lO_l}$ (twice the area of triangle ΔPS_lO_l), and $\overline{PR} = \overline{S_lP}\sin 2\theta$, we have $V_{hkl} = (\Omega/r)\,\overline{S_lO_l}\ \overline{S_lP}\sin 2\theta$. Because $\overline{S_lO_l} = r\cos\alpha_i$ and $\overline{S_lP} = \overline{S_lT} = r\cos\alpha_f$, we then get $V_{hkl} = \Omega r\cos\alpha_i\cos\alpha_f\sin 2\theta$. Because the angular velocity Ω and Ewald sphere radius r are constants for all reflections, we can simply drop them out and arrive at

$$L_{hkl} \sim \frac{1}{\cos\alpha_i\cos\alpha_f\sin 2\theta}. \tag{11}$$

Here α_i, α_f and 2θ are the incident, exit, and in-plane scattering angles that are also defined in Fig. 2b. In a more general form for oriented samples with its randomness in the surface plane (or an ω scan), Lorentz factor is written as

$$L \sim \frac{1}{S}\frac{1}{\sin 2\theta}, \tag{12}$$

where S is a level-scale factor which is constant for a given level l, but differing in form for different sample systems and experiment geometries. For a three-dimensional system with random orientation in the sample surface plane, $S = \cos\alpha_i\cos\alpha_f$ as shown in Eq. 11. For a two-dimensional system with its grains randomly oriented in the sample surface, $S = 1$ [Als-Nielsen and McMorrow, *Elements of Modern X-ray Physics*, 2nd Edition, page 187, Wiley (2011)].

For a completely random sample such as powders, the scattering depends only on the oblique angle 2Θ (see Fig. 2a) [Cullity, *Elements of X-Ray Diffraction*, 1st Edition, page 128, Addison-Wesley (1956)], and the Lorentz factor is

$$L \sim \frac{1}{4\sin^2\Theta\cos\Theta}. \tag{13}$$

This is often the case for bulk samples measured in a transmission geometry such as SAXS and WAXS.

For a single crystal that randomly rotates across its Bragg angle [Als-Nielsen and McMorrow, *Elements of Modern X-ray Physics*, 2nd Edition, page 182, Wiley (2011); Reynolds, *Clay Clay Miner.* **34**, 359 (1986)], the Lorentz factor is

$$L \sim \frac{1}{\sin 2\Theta_{Bragg}}. \tag{14}$$

To perform the Lorentz correction in `gixsdata`, one needs to specify a Lorentz factor type which is given by `LorentzFactorType` (see appendix A). There are four types of Lorentz corrections provided: (1) no correction, (2) in-plane randomly oriented 3D structure, (3) in-plane randomly oriented 2D structures, and (4) completely random structures. To get the Lorentz correction matrix, use method `lorentzfactor` (see appendix B). These four types are typical situations in GIXS experiments. As we have mentioned above that Lorentz correction only applies to periodic structures. One should be aware that although the Lorentz correction for convenience applies to the entire image in `gixsdata`, only the intensities of pixels representing structure factors should be corrected. Therefore, one can only perform Lorentz correction if he/she needs to analyze structure peak intensities in `gixsdata`. It should also be noted that the four Lorentz factor types provided in `gixsdata` do not cover all situations that a user might encounter with. For more complicated sample orientations or scattering geometries, one needs to apply his/her own Lorentz correction to the image or more conveniently to the one-dimensional line profile. For example, the specular rod $(00l)$ for which the in-plane scattering angle $2\theta = 0$ and the rocking angle is the incident angle α requires Lorentz factor $L \sim 1/\sin 2\alpha$ [Smilgies, *Rev. Sci. Instrum.* **73**, 1706 (2002); Shayduk, *J. Appl. Cryst.* **43**, 1121 (2010)].

6. Other corrections

There are some other corrections not included in `gixsdata`, such as sample absorptions, resolution for beam divergence and energy dispersion, footprint for illuminated sample area, intensity integration of the interception of scattering rod for surface diffuse scattering analysis, etc [Vlieg, *J. Appl. Cryst.* **31**, 198 (1998), and **30**, 532 (1997); Robach et al., *J. Appl. Cryst.* **33**, 1006, (2000); Smilgies, *Rev. Sci. Instrum.* **73**, 1706 (2002); Smilgies, *J. Appl. Cryts.* **42**, 1030, (2009); He, *Two-dimensional X-ray Diffraction*, 1st Edition, chapter 6, Wiley (2009)]. Depending on specific sample types, experiment configurations, and the needs for data analysis, not every correction is needed. For example, one would expect the path of X-rays is longer for large exit angles for a uniform sample so that a sample absorption correction would be [Baker et al., *Langmuir* **26**, 9146 (2010)],

$$A = \frac{1-\exp\left(-\mu t\left[\frac{1}{\sin\alpha_i}+\frac{1}{\sin\delta}\right]\right)}{\mu\left[\frac{1}{\sin\alpha_i}+\frac{1}{\sin\delta}\right]}, \tag{15}$$

where μ is average linear mass attenuation coefficient of the film, α_i is the incident angle, and δ is the angle defined in Fig. 4. However, in many films, the sample absorption is negligibly small and therefore is dropped out here.

One can apply these specific or customized corrections by generating a correction matrix (the same dimension as the `RawData`) and store it in `CustomCorrection`

of *gixsdata* which is to be multiplied automatically to `SolidAngleCorrectedData`. Alternatively, one can more conveniently (and this also make a lot of sense) apply these corrections later to line profiles or discrete reciprocal points during further data analysis or fitting, rather than upfront to the entire image through `CustomCorrection`.

Some distortions to the scattering intensities may also need corrections but cannot be performed straightforwardly by multiplying a correction coefficient. For example, the scattering enhancement near the critical angle of the total external reflection is a dynamical scattering effect and can only be treated with proper surface scattering theories such as distorted wave Born approximation (see section V). In fact, one may be able to take advantage of this intensity distortion to obtain the surface structures with a high resolution normal to the surface [Jiang et al., *Phys. Rev. B* **84**, 075440 (2011)].

D. Calculation of q maps

Q map values can be calculated easily in the lab reference frame and then transformed to the sample reference frame. Assume the incident beam is norm to the detector. In lab frame (Fig. 4), the incident wave and scattered wave vectors, $\boldsymbol{k}_i$ and $\boldsymbol{k}_f$, are

$$\begin{aligned} k_{i,x}^{lab} &= k \\ k_{i,y}^{lab} &= 0 \\ k_{i,z}^{lab} &= 0, \end{aligned} \tag{16}$$

and

$$\begin{aligned} k_{f,x}^{lab} &= k\cos\delta\cos\gamma \\ k_{f,y}^{lab} &= k\cos\delta\sin\gamma \\ k_{f,z}^{lab} &= k\sin\delta, \end{aligned} \tag{17}$$

respectively. Therefore, the wave vector transfer $\boldsymbol{q}^{lab}$ in the lab frame is given by

$$\begin{aligned} q_x^{lab} &= k_{f,x}^{lab} - k_{i,x}^{lab} \\ q_y^{lab} &= k_{f,y}^{lab} - k_{i,y}^{lab} \\ q_z^{lab} &= k_{f,z}^{lab} - k_{i,z}^{lab}. \end{aligned} \tag{18}$$

The sample surface makes an angle with the incident beam of α_i (the incident angle), so the coordinate rotation matrix with a rotation angle of $-\alpha_i$ (right hand rule) with respect to the y axis is

$$R_y(-\alpha_i) = \begin{bmatrix} \cos\alpha_i & 0 & -\sin\alpha_i \\ 0 & 1 & 0 \\ \sin\alpha_i & 0 & \cos\alpha_i \end{bmatrix}. \tag{19}$$

The sample surface may be tilted with respect to the x axis by an angle, let's say ζ, so that the scattering plane

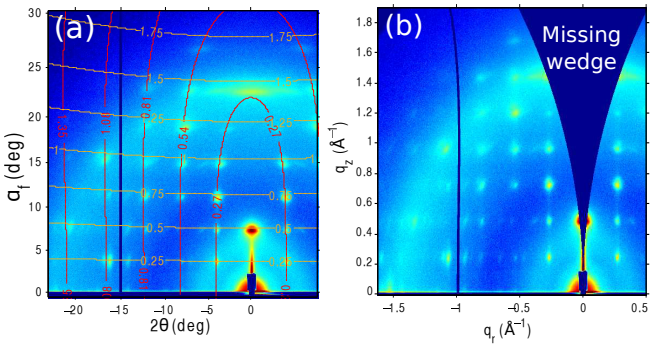

FIG. 7. GIWAXS pattern (a) before and (b) after reshaping with respect to q_z and q_r, whose contour lines are shown in gold and red colors in (a).

is not in alignment with the natural arrangement of the detect pixel arrays. The rotation matrix is

$$R_x(\zeta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\zeta & -\sin\zeta \\ 0 & \sin\zeta & \cos\zeta \end{bmatrix}. \tag{20}$$

So we can rewrite q in the sample frame as

$$\boldsymbol{q} = R_y R_x \boldsymbol{q}^{lab}. \tag{21}$$

E. Linecut

With a built-in `linecut` method of `gixsdata`, one can reduce 2D GIXS data into 1D profiles with respect to any angle or q components prior to further structure modeling (such as for lattice parameter, nano-structure size and shape analysis) with functions either integrated with Matlab Curve Fitting Toolbox or developed by users for specific purposes. Constraints are often added to specify region of interest (ROI) for the linecut. See appendix B for usage. In the GUI mode, the linecut is performed on `SolidAngleCorrectedData`, unless it does not exist.

F. Reshape image

The measured intensity distribution on an area detector does not directly represent the sample's reciprocal space, i.e., the q maps on the detector are not orthogonal for the natural arrangement of the pixel arrays. One rather measures the projection of the reciprocal lattice intercepted by the Ewald sphere onto the detector plane. To do analyses such as for crystalline orientation with respect to surface normal, one can either use the 1D profiles as a function of angles directly obtained with *linecut*, or more conveniently reconstruct the GIXS image to one represented by orthogonal reciprocal space axes (e.g., in q_z and q_r, or in q_z and χ with χ the polar angle of the reciprocal space). Crystalline and texture orientation analysis is often done by analyzing of the intensity distribution in terms of χ. For example, Fig. 7a

shows a raw GIWAXS pattern from a thin film of conjugated block-copolymers, which self-assemble into 2D powder-like nano-structures in the film plane. The contours of constant out-of-plane and in-plane wave vector transfers, q_z and q_r, respectively, in the raw scattering pattern do not follow the columns and rows of pixels and are not orthogonal to each other, especially at high exit and in-plane scattering angles. Reshaping with the built-in method `reshape_image` redistributes the intensities into the natural reciprocal space coordinates of the sample (Fig. 7b). After reshaping, a blank region (a.k.a. missing wedge) appears, which is a q space that cannot be probed at a fixed incident angle. This often causes a problem for crystalline orientation analysis with respect to the film surface using Bragg diffractions on the q_z axis. One solution is to collect a series of images by scanning the incident angle near the the angle for the Bragg diffraction condition to patch up the missing wedge, so called pole figure construction [Baker et al., *Langmuir* **26**, 9146 (2010)].

G. Find diffraction peak

Method `findpeak` provides a routine to find the diffraction peak position of a ROI of `MaskedData` or `SolidAngleCorrectedData` (the latter is forced in the GUI mode unless it does not exit due to incomplete setup parameters.). Two models are implemented: COM (center-of-mass) and BVND (bivariate-normal-distribution). The first model calculates the COM of the user-specified ROI (which can be defined through the GUI interface or the script mode) and returns the peak position in terms of x and y pixels. The BVND model assumes a bivariate normal distribution for the shape of the peak and fits the intensity data in the ROI for the peak position. The probability density function of the bivariate normal distribution is given by

$$f(x,y) = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} \exp\left[-\frac{z}{2\,(1-\rho^2)}\right], \quad (22)$$

where

$$z = \frac{(x-x_0)^2}{\sigma_x^2} - \frac{2\rho(x-x_0)(y-y_0)}{\sigma_x\sigma_y} + \frac{(y-y_0)^2}{\sigma_y^2}, \quad (23)$$

and

$$\rho = \mathrm{cor}(x,y) \quad (24)$$

is the correlation of x and y. Geometrically ρ can be viewed as a parameter related to the rotation angle of the peak, who has finite widths σ_x and σ_y). Most diffraction peaks can be well described by the BVND model. During the fitting in the program, the model function is

$$I = C_x x + C_y y + C + A\exp\left[-\frac{z}{2\,(1-\rho^2)}\right], \quad (25)$$

where A is the normalization and scaling factor, $C_x x + C_y y + C$ is the plane background.

After the peak position is found, the x and y pixel values are rounded off and its map values, i.e., q, Φ, q_z, q_x, q_y, q_r, 2θ, α_f, and χ, are calculated.

IV. GUI MODE (GIXSGUI)

While the properties and methods defined in *gixsdata.m* provides an independent and complete access to the `gixsdata` object in a script mode, one can also work in a graphical user interface provided by *gixsgui.m*, to do the same job. Figure 8 shows the main GUI control panel. Hover the cursor on each GUI object (push buttons, list boxes and etc.) for quick instructions.

- First thing to do is to load a parameter file (Matlab *.mat) by "Load Params" created previously by "Export Params". "Set as Default" sets current parameters, i.e., setup parameters, corrections (efficiency and flat field), plot parameters, masks, as well as maps (if they exist) as the default parameters to be applied to the following images loaded to the image space.

  To be compatible with *gisaxsviewer*, a legendary Matlab package to view Mar165 GISAXS data (2048×2048), one can load *.par parameter file.

- Locate image file path by "Load Path". Supported image files with .tif, .mat, .cbf, .edf, and .fits extensions, as well as subfolders, are displayed in the path list box (upper). Double click image file, or single click followed by "Add", to load into image space listed in the list box (lower).

- "Remove" and "Remove All" push buttons are used to remove loaded images from the image space. Because a loaded image and its maps occupies a lot of memory (up to 100MB for reflection geometry), removing unused image data helps save memory and speeds up data visualization and processing.

- Use "Get beam0" to draw a rectangular box around the direct beam to find the direct beam pixel.

- Use "Get SDD" to calibrate sample-to-detector distance on an image taken from a standard sample. Only silver behenate (AgBe) powder sample is currently supported. SDD is obtained by selecting at least three points along the 1st order (001) diffraction ring. One can alternatively use the specular reflections from a thin film or flat substrate (usually silicon wafer) to calibrate (use "SDD Calibration" in the Misc Tools).

- Use "Get Specular" to define the scattering plane (for reflection geometry), the plane containing the incident beam and perpendicular to the sample surface. Beam zero position has to be defined prior

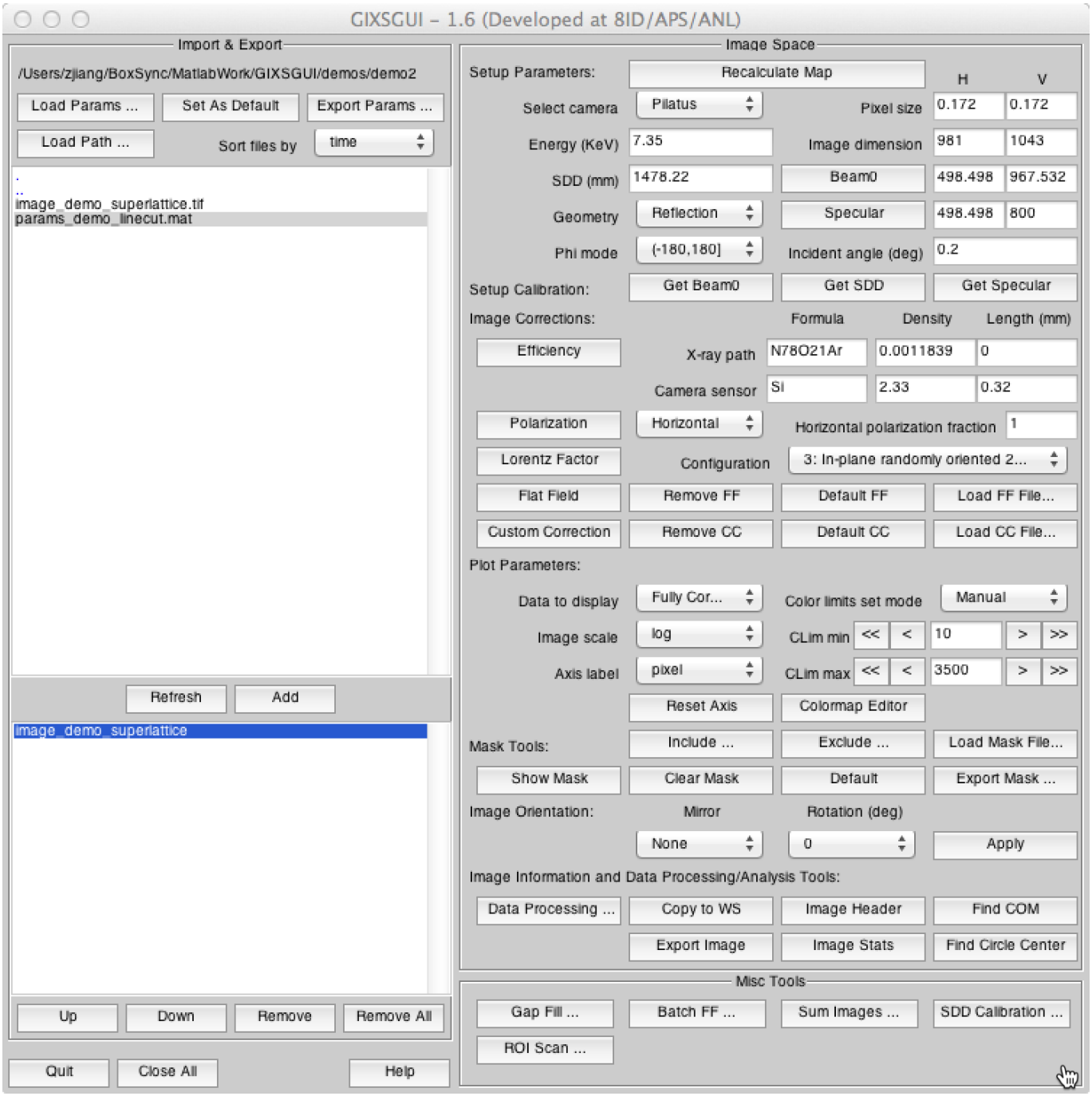

FIG. 8. *GIXSGUI* graphical user interface.

to using "Get Specular". The scattering plane is found by selecting two points across the vertical scattering streak. The x, y pixel coordinates of the center point, which is used along with `Beam0` to define the scattering plane, are found from the position of the FWHMs.

- Next define parameters for image corrections. X-ray path length is not always equal to `SDD`. Instead, it is the length not in vacuum between sample and detector (direct beam position). Refer to appendix A for instruction of setting efficiency corrections.

- Next define parameters for plotting images. "Data to plot" specifies either "Uncorrected" or "Corrected" image data to plot. Flat field correction applies to both "Uncorrected" and "Corrected", and the difference lies in efficiency and solid angle corrections.

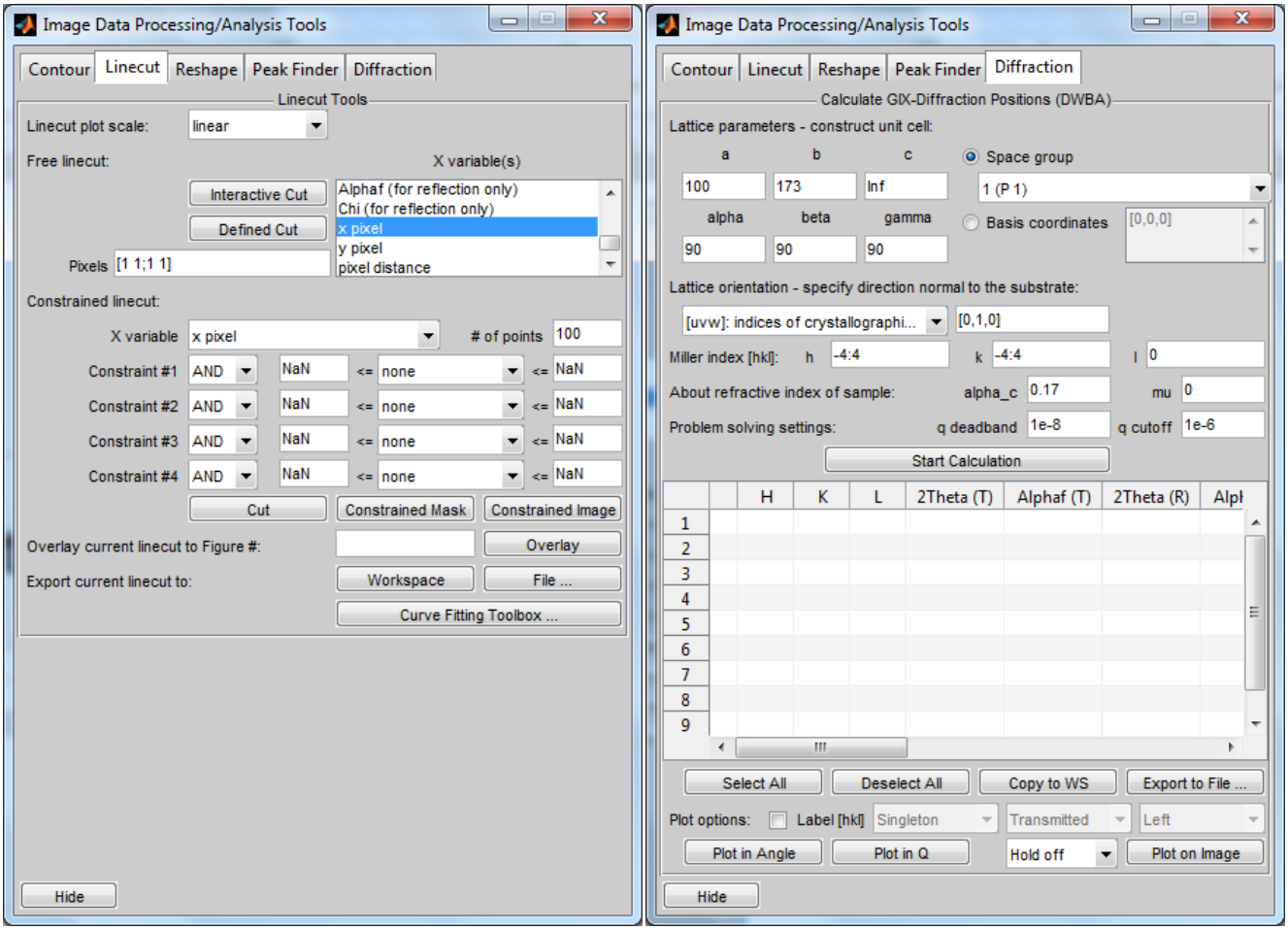

FIG. 9. Image data processing tools. Left: Linecut; Right: 3D nano-structure indexing.

Uncorrected: `MaskedData` (with with flat field correction).

Corrected: masked `SolidAngleCorrectedData` (with flat field, polarization, efficiency, and solid angle corrections).

- Next define `Mask`, which automatically applies to `RawData` and assigns the masked image data to `MaskedData`.

- Mirror and rotation operations on the current image can be performed. Because `gixsdata` properties will be altered, these operations should be done after other setup parameters are complete, e.g., direct beam and specular positions, pixel sizes ("Other" camera should be used if pixel sizes are asymmetric), flat field correction, and mask.

- A bunch of tools are provided to view current image information and to process image data such as linecut and image reshaping. "Find Circle Center" finds the geometrical center of a circle defined by at least three points in the current selected image."Image Header" displays `ImFileInfo` of the current selected image. "Image Stats" Displays the statistics of `MaskedData` of the current selected image.

- Use "Data Processing ..." to open a new GUI (Fig. 9) to draw contour lines, perform linecuts, reshape images, find diffraction peak positions, and index 3D structures.

- Two ways of reducing 2D data to 1D line profiles are implemented: free linecut and constrained linecut. The former is a pixel based linecut, where the pixels are defined either by mouse clicking ("Interactive Cut") or by typing the pixel coordinates ("Defined Cut"). The result of the free or the constrained linecuts can be plotted and overlayed to a figure with specified figure number that contains other linecuts. After linecut, the data can be either exported to file or workspace, or can be passed directly to Matlab Curve Fitting Toolbox for further fitting and modeling (license required for the toolbox).

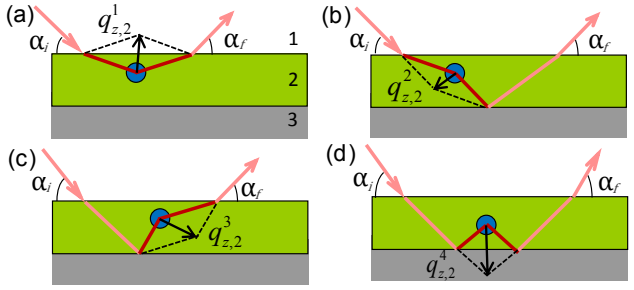

FIG. 10. Schematic of the four scattering events inside a supported film as a result of the combination of refraction, diffraction and reflection.

GIXSGUI provides a number of miscellaneous tools to facilitate image processing and visualization.

- "Gap Fill": Fills Pilatus 1MF module gaps by patching two or more images taken at different camera positions.
- "Batch FF": Applies flat field correction to images in a batch mode.
- "Sum Images": Sums or averages images and exports to desired data type.
- "SDD Calibration": Rather than using silver behenate (AgBe) standard, one can use a few specular reflections from a flat wafer to fit for the sample-to-distance.
- "ROI Scans": Extracts integrated intensity over a user-defined ROI for a series of 2D image scans. For example, one can monitor the scattering intensity of a stationary ROI during a *in-situ* process; or obtain the reflectivity curve by tracking a moving ROI that always fulfilling the specular reflection. A built-in function `roiscan` provides a more flexible script mode.

V. 3D NANO-STRUCTURE INDEXING

GIXS is nowadays routinely performed to solve for the nano-scopic crystalline structures in assembled thin films consisting of nanoparticles, macromolecules and their composite materials. 3D nano-structure indexing is highly demanded for fast characterization of the morphology and quality of the assembly in these films. The indexing method is slightly different from that applied in traditional crystallography on powders, due to the refraction and reflection effects in the presence of interfaces (e.g. film/air and film/substrate).

In the framework of distorted wave Born approximation (DWBA) [Sinha et al., *Phys. Rev. B* **38**, 2297 (1988); Lee et al., *Macromolecules* **38**, 4311 (2005); Tate et al., *J. Phys. Chem. B* **110**, 9882 (2006); Jiang et al., *Phys. Rev. B* **84**, 075440 (2011)], the differential scattering cross-section is give by

$$\left(\frac{d\sigma}{d\Omega}\right)_{\rm diff} \approx r_e^2 |\Delta\rho|^2 \left|\mathcal{F}\left(q_r, k^i_{z,1}, k^f_{z,1}\right)\right|^2, \tag{26}$$

where r_e is Thompson scattering length of electron, $\Delta\rho$ is the electron density contrast of the nano-structures in the film, $k^i_{z,1}$ and $k^f_{z,1}$ are the z components of the incident and exit wave vectors in medium 1 above the film (e.g. air or vacuum), respectively, and

$$\begin{aligned}\mathcal{F}\left(q_r, k^i_{z,1}, k^f_{z,1}\right) &= T_2\left(k^f_{z,2}\right) T_2\left(k^i_{z,2}\right) F\left(q_r, q^1_{z,2}\right) \\ &+ R_2\left(k^f_{z,2}\right) T_2\left(k^i_{z,2}\right) F\left(q_r, q^2_{z,2}\right) \\ &+ T_2\left(k^f_{z,2}\right) R_2\left(k^i_{z,2}\right) F\left(q_r, q^3_{z,2}\right) \\ &+ R_2\left(k^f_{z,2}\right) R_2\left(k^i_{z,2}\right) F\left(q_r, q^4_{z,2}\right)\end{aligned} \tag{27}$$

Here F is the Fourier transform of the electron density variation in the film and is simply a form factor in the case of dilutely distributed objects. The four terms in Eq. (27) have clear physical meanings and represent four scattering events (see Fig. 10) as a result of refraction at the film surface, reflection at the film/substrate interface, as well as diffraction of the nano-structure. Each term is weighted by combinations of incident and exit wave vector (i.e. angle) dependent T_2 and R_2, which are complex amplitudes of transmitted and reflected waves in medium 2 (film) and can be calculated by Parratt's recursion method [Parratt, *Phys. Rev.* **95**, 359 (1954); Tolan, *X-Ray Scattering from Soft-Matter Thin Films*. Berlin: Springer (1999)]. The z components of the wave vector transfer of the four events are respectively given by

$$\begin{aligned} q^1_{z,2} &= k^f_{z,2} - k^i_{z,2}, \\ q^2_{z,2} &= -k^f_{z,2} - k^i_{z,2}, \\ q^3_{z,2} &= k^f_{z,2} + k^i_{z,2}, \\ q^4_{z,2} &= -k^f_{z,2} + k^i_{z,2}. \end{aligned} \tag{28}$$

Here, $k^i_{z,2} = k_2 \sin\alpha^i_2$ and $k^f_{z,2} = k_2 \sin\alpha^f_2$ (with k_2 the wave vector amplitude of the X-ray in medium 2), are z components of the incident and exit wave vectors in medium 2, respectively, and can be determined via Snell's law $\cos\alpha^{(i,f)}_1 = n_2 \cos\alpha^{(i,f)}_2$ (with n_2 the index of refraction in medium 2). At a fixed incident angle, scattering for a given wave vector transfer (q_z, q_r), may exit the film surface at two angles because of the refraction and reflection effects,

$$\alpha_f = \arcsin\sqrt{\left(\frac{q_z}{k}\right)^2 + \sin^2\alpha_i \mp \frac{2q_z}{k}\sqrt{n^2 - 1 + \sin^2\alpha_i}}, \tag{29}$$

where the $-$ channel is for terms 1 and 2, and the $+$ channel is for terms 3 and 4 in Eq. (27), respectively.

The in-plane angle of scattering is found to be

$$2\theta = \arccos\left(\frac{\cos^2\alpha_f + \cos^2\alpha_i - \left(\frac{q_r}{k}\right)^2}{2\cos\alpha_f\cos\alpha_i}\right). \quad (30)$$

As implied in Fig. 10a and 10b, the − channel occurs with the incident beam directly scattered from the structure without first being reflected from the substrate, and is therefore named the transmission channel; while in Fig. 10c and 10d, the + channel occurs with the incident beam reflected from the substrate before scattering, and is therefore named the reflection channel.

When the Laue condition $\boldsymbol{q} = \boldsymbol{G}$ ($\boldsymbol{G}$ the reciprocal lattice vectors of the nano-structures), is fulfilled, diffraction can be observed in the transmission and reflection channels whose exit angles angles can be obtained through Ewald sphere construction described as follows. The Ewald sphere is defined in the lab frame (coordinate system defined in Fig. 4), whereas the reciprocal space is conveniently defined in the sample frame (coordinate system defined in Fig. 2). The two frames are related by an incident-angle dependent rotation matrix

$$R_y(\alpha_i) = \begin{bmatrix} \cos\alpha_i & 0 & \sin\alpha_i \\ 0 & 1 & 0 \\ -\sin\alpha_i & 0 & \cos\alpha_i \end{bmatrix}. \quad (31)$$

The reciprocal lattice vector in the lab frame is then rewritten as

$$\begin{aligned} \boldsymbol{G}_{lab} &= (G_x, G_y, G_z) = R_y\boldsymbol{G}_{sample} \\ &= R_y(h\boldsymbol{a}_1^* + k\boldsymbol{a}_2^* + l\boldsymbol{a}_3^*), \end{aligned} \quad (32)$$

where (h, k, l) are Miller indices, and $(\boldsymbol{a}_1^*, \boldsymbol{a}_2^*, \boldsymbol{a}_3^*)$ are the reciprocal space basis vectors. Laue condition requires that lattice points fall on Ewald sphere surface so as to be observed,

$$|\boldsymbol{G}_{lab} - \boldsymbol{k}|^2 - |\boldsymbol{k}|^2 = (G_x - k)^2 + G_y^2 + G_z^2 - k^2 \leq \Delta G^2, \quad (33)$$

where ΔG accounts for the finite size of the diffractions in the reciprocal space and any mosaicity of the domains.

Domains of substrate supported nano-structures often have a statistically distributed orientation with respect to the surface normal. If the orientation is the isotropic in the plane, which is often the case for most self-assembled structures on flat substrates, the reciprocal lattice becomes a set of q_r-dependent rings (2D powder rings). Therefore, $\boldsymbol{G}_{lab}$ fulfilling Laue condition is related to the wave vector transfer (q_x, q_y, q_z), which is defined in the sample frame, by

$$G_x^2 + G_y^2 + G_z^2 - q_r^2 + q_z^2 + q_z(G_x\sin\alpha_i - G_z\cos\alpha_i) = 0, \quad (34)$$

and

$$-G_x\sin\alpha_i + G_z\cos\alpha_i = q_z. \quad (35)$$

The diffraction angles from nano-structures of given lattice type and parameters are then calculated by substituting (q_r, q_z) values obtained by simultaneously solving Eqs. (33-35) into Eqs. (29-30).

In a diffraction pattern, all those symmetric extinctions or absences can be traced back to the symmetry elements that have caused them, providing a very useful method for space-group identification. In *GIXSGUI*, one can calculate the diffraction positions of any of the 230 space groups and superimpose them over the experiment GIXS patterns to determine the space group of the sample. For simple lattices one can straightforwardly construct the unit cell by assigning the fractional basis coordinates.

VI. DEMONSTRATIONS

Image and parameter files as well as scrip-mode codes for demonstrations can be found in `~\GIXSGUI\demos\`. Users are encouraged to try them out.

A. Demo 1: SDD Calibration

One can use a SAXS pattern from a AgBe standard to calibrate the sample-to-detector distance. This is often accurate enough for GISAXS/SAXS experiments. If the direct beam position (Beam0) is already provided, the program will use that value to calculate SDD. Otherwise, both SDD and Beam0 are fitted.

For GIWAXS experiments with an ultra-short SDD, the positioning of the standard may be a challendge. One can alternatively is to use the reflections of a thin film or a flat substrate (typically a silicon wafer) to calibrate. At least two incident angles (θ) are needed for the calibration. "SDD Calibrate ..." takes the pixel sizes and Beam0 positions defined in *GIXSGUI* main control panel, and user input of incident angles and corresponding x, y pixel positions, and fit (Matlab Optimization Toolbox is required) to a geometrical function

$$r = \mathrm{SDD}\tan\left(2\left(\theta + \Delta\theta\right)\right), \quad (36)$$

where r is the distance between the specular reflection to Beam0, and $\Delta\theta$ is the offset angle for θ alignment (another benefit of using reflections to calibrate). One can use "Find COM" to locate the center of mass of the specular reflections.

B. Demo 2: Linecut

Fig. 12a is GISAXS pattern from a silicon-supported film of spherical gold nanoparticles ordered in a two-dimensional superlattice. The superlattice is prepared via droplet evaporation of colloidal suspension in toluene, in which the nanoparticles self-assemble into a monolayer superlattice at the interface [Narayanan et al., *Phys. Rev. Lett.* **93**, 135503 (2004); Jiang et al., Nano Lett. **10**, 799 (2010)]. With reciprocal space q maps automatically assigned in *GIXSGUI*, the q_y values at the 2D Bragg peaks

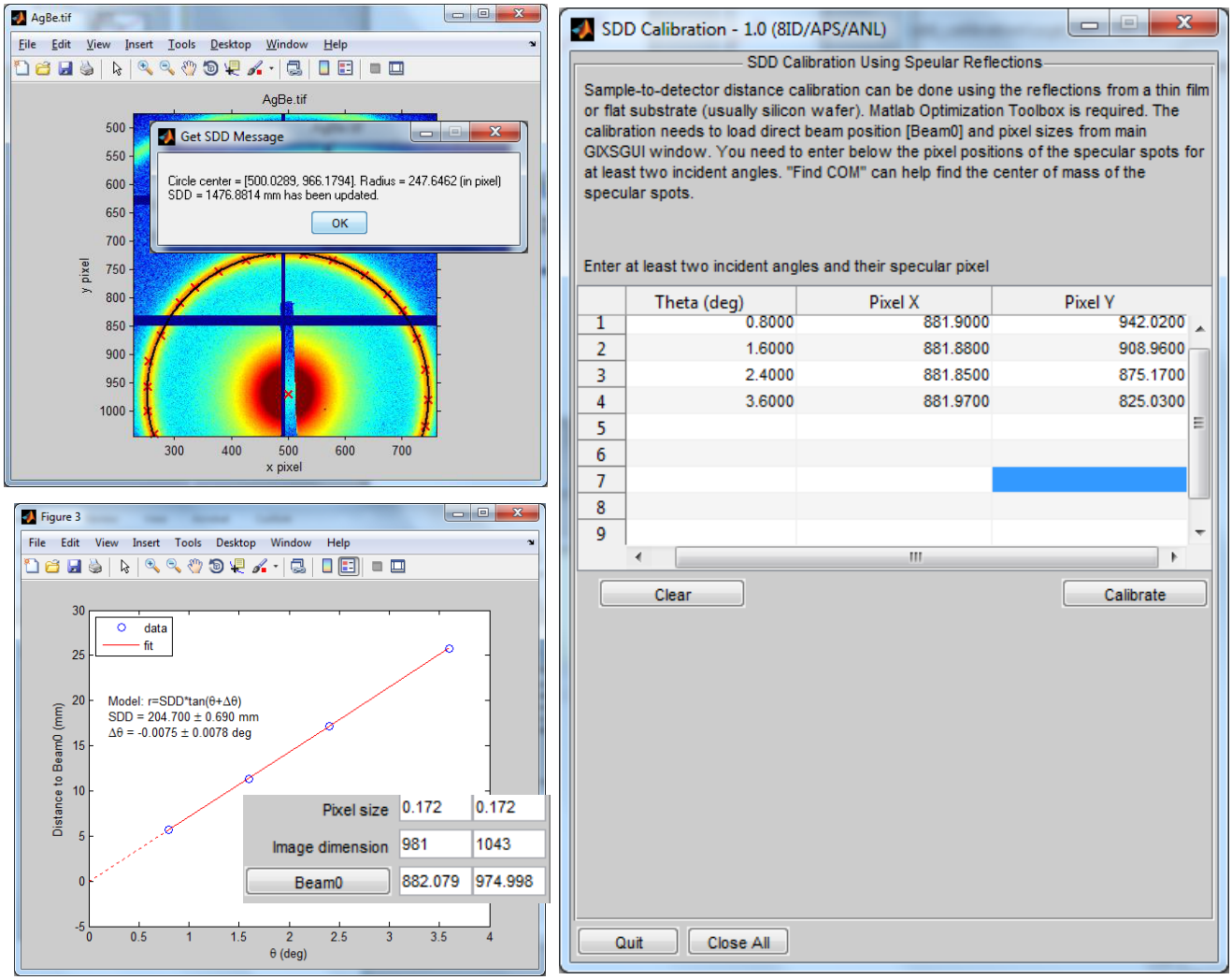

FIG. 11. Two ways to calibrate the sample-to-detector distance. Top left panel: Use AgBe standard to calibrate. Other panels: Use the reflections of a flat silicon wafer to calibrate. Please note the two calibration examples were performed at different times with different SDDs.

(vertical rods) were found to scale as $1 : \sqrt{3} : 2$, indicating a 2D hexagonally close-packed (HCP) lattice. For further analysis, line profile along q_y (Fig. 12b) was obtained by *linecut* control panel and was directly passed to Matlab Curve Fitting Toolbox (license required) to fit the (10) Bragg peak using a Lorentzian function (provided in *GIXSGUI*; enter `type lorentzian` in Matlab command window to see the definition) with a linear background. Optimization returns a (10) peak position at $a = 0.08763$ Å^{-1} and FWHM of $2b = 0.0054$ Å^{-1}. Therefore, the nearest-neighbor (or inter-particle) distance is $\left(2/\sqrt{(3)}\right)(2\pi/a) = 82.8$ Å, and the coherent domain size is estimated to be π/FWHM = 582 Å via Scherrer's relation.

Linecut can be also be performed from script mode. See appendix B for `linecut` usage. A commented script file `~\GIXSGUI\demos\demo2\scpt_demo_linecut.m` is provided for illustration.

C. Demo 3: Image Reshaping

Fig. 13 shows how to reshape an image in the reshape GUI panel to q_z vs q_r or q_z vs χ, the latter is also known as polar figure for crystallinity, texture, and crystalline orientation analysis with respect to the film normal.

Reshaping can also be done from a script. See appendix B for `reshape_image` usage. A commented script file `~\GIXSGUI\demos\demo3\scpt_demo_reshape.m` is provided for illustration.

D. Demo 4: ROI Scan

With an area detector, it is still feasible to carry out point-detector scans. One simply needs to define a virtual slit using a ROI on the area detector, so called ROI scan, which enables the conventional scans such as reflec-

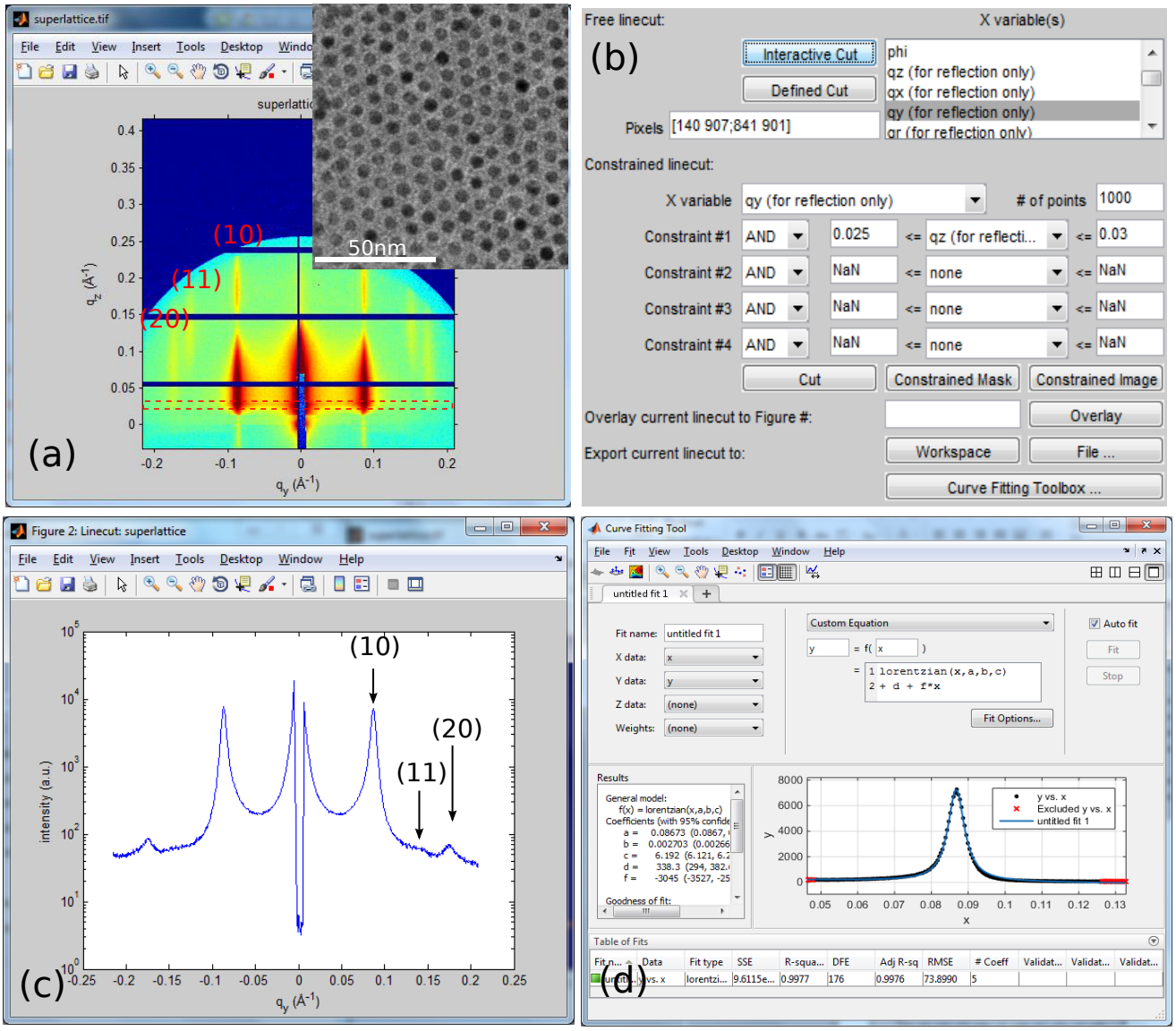

FIG. 12. (a) GISAXS pattern from a film of gold nanoparticles monolayer superlattice taken with 7.35 kev X-rays incident at 0.2°. Inset shows a TEM image of the monolayer. (b) shows a portion of the linecut panel. (c) is the 1D q_y integrated intensity line profile of the red dashed line in (a) with its constrained linecut settings shown in (b). (d) is fitting of the (10) diffraction peak to Lorentzian function in Matlab Curve Fitting Toolbox.

tivity, rocking (transverse diffuse) curve scan etc. with an area detector. One can also fix the ROI to study the time-dependent of the intensity variation of a given ROI for the time-resolved experiments.

One can start "ROI Scan" *GIXSGUI* to perform ROI scans. In GUI mode, constraints defined ROI remains unchanged during the scan. However, for certain scans such as reflectivity with a position-fixed area detector, the virtual detractor slit (ROI) moves along with the incident angle in order to keep tracking of specular reflection. To update ROI for each scan point during the scan, one has to use the `roiscan` function in the script mode (see appendix D). A commented script file `~\GIXSGUI \demos\demo4\scpt_demo_roiscan_reflectivity.m` is provided to illustrate how to reduce a set of 2D GIXS images to 1D reflectivity profile.

E. Demo 5: 3D Nano-structure Indexing

Diffractions in Fig. 14 arise from a self-assembled thin film of spherical gold nanoparticles blended in block copolymer (BCP)-based supermolecules (poly(styrene)-b-poly(4-vinylpyridine)-(3-pentadecylphenol)$_r$ or PS-b-P4VP(PDP)$_r$ in short with the subscript r the ratio of PDP to 4VP unit) [Kao et al., *Nano Lett.* **12**, 2610 (2012)]. After solvent annealed under chloroform, with the help of the assembly of PS(19)-b-

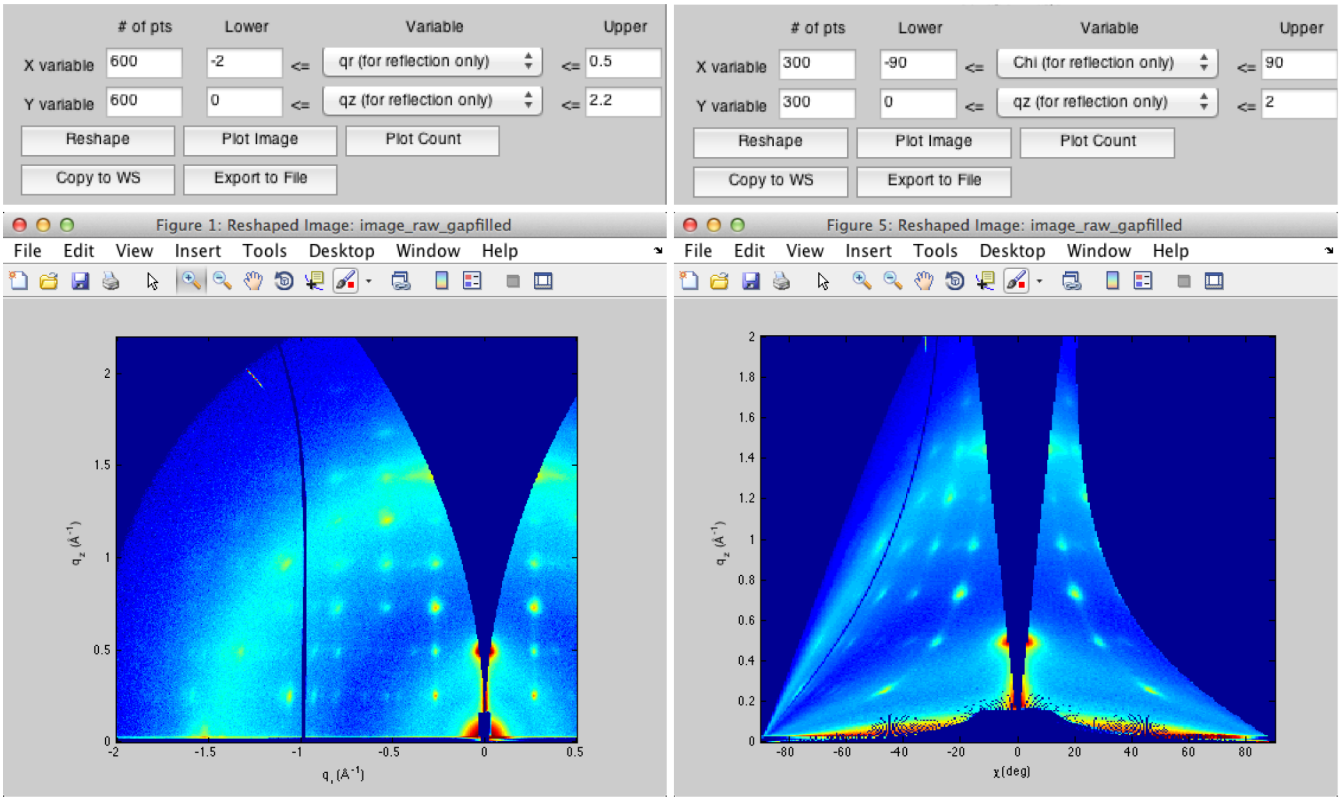

FIG. 13. Image of Fig. 7a reshaped image to q_z vs. q_r (left) and q_z vs. χ (right).

P4VP(5.2)$(PDP)_{1.7}$, Au nanoparticles (6 vol% loading) diffused into the dented-rectangular interstitial sites among PS cylinders (parallel to the film surface) to form nanoparticle chains. X-rays of 7.35 keV are incident at the sample surface at a grazing angle of 0.23 deg. The diffraction peaks are best fit to a rectangular centered lattice of cylinders parallel to the film surface which can be constructed as a non-primitive unit cell with lattice parameters a = 330 Å, b = 425 Å and c = Inf, $\alpha = \beta = \gamma = 90$ deg, basis fractional coordinates $[0, 0, 0]$ and $[\frac{1}{2}, \frac{1}{2}, 0]$, and $[uvw]$ = [010] (b axis) perpendicular to the film. Since the cylinders pack into a 2D pattern (cross-section of the film alike everywhere) of point group $mm2$, one can alternatively use space group no. 35 $Cmm2$ to calculate the diffraction pattern. Note that two diffraction sets (white for reflection and red for transmission channels) appear as a result of the refraction and reflection effects, although they arise from the same structure. The extent of the vertical separation of the two channels depends on the difference of the incident angle above the critical angle, whereas they usually merge when the incident angle is less than the critical angle. One therefore should be careful during the structure indexing on GIXS patterns in order to avoid assigning the two channels to different lattice planes.

The allowed diffraction miller indices and their corresponding exit and in-plane positions (i.e., q_z and q_r) can be calculated in the script mode using function *gixsdiffpos* (see appendix C). A commented script file `~\GIXSGUI\demos\demo5\scpt_demo_indexing_np_bcp.m` is provided for illustration.

Appendices

Appendix A: List of gixsdata Properties

In Matlab command window, type `gixsdata` to view the default properties of a `gixsdata` object.

```
                    Camera : 'Pilatus'
                 PixelSize : [0.172 0.172]
                     Beam0 : [NaN NaN]
                       SDD : NaN
                   XEnergy : 7.35
                  Geometry : 2
                   PhiMode : 1
            PolarizationMode : 2
HorizontalPolarizationFraction : 1
             IncidentAngle : NaN
                  Specular : [NaN NaN]
             PlotCLimsType : 1
                 PlotCLims : [1 2000]
```

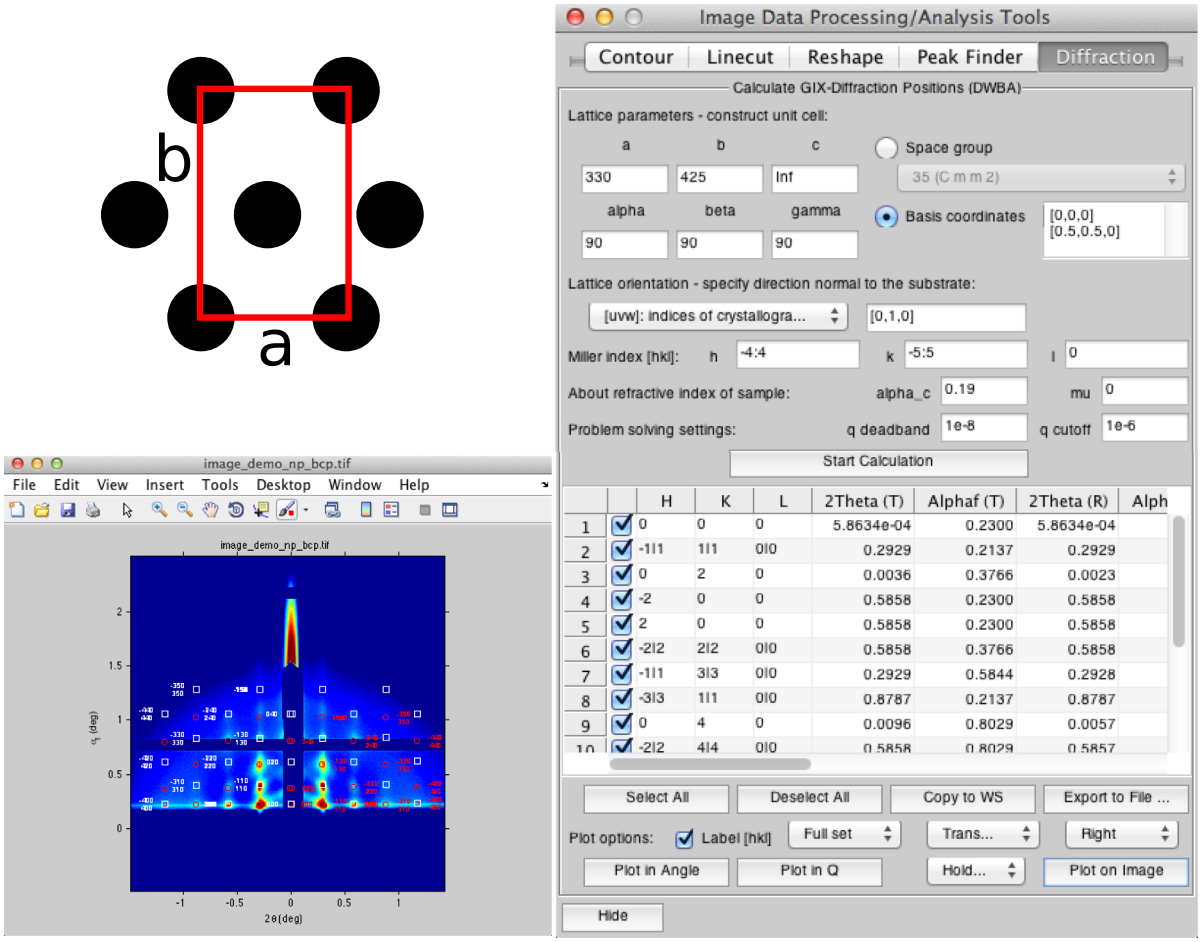

FIG. 14. Bottom left: GISAXS pattern from a film of gold nanoparticle loaded block-polymer-based supermolecular nanocomposites. Top left: schematic view of the film's cross-section showing a rectangular centered lattice consisting of PS cylinders and nanoparticle chains parallel to the film surface. Red circles and white squares labeled with Miller indices are calculations using lattice parameters given in the Right panel for the transmission and reflection channels, respectively. Space group no. 35 *Cmm*2 and manually constructed unit cell both give identical calculations of the diffraction peaks.

PlotScale : 2
PlotAxisLabel : 1
PlotImageFlag : 1
ImFile : ‘’
RawData : [ ]
LorentzFactorType : 1
FlatField : [ ]
CustomCorrection : [ ]
EfficiencyCorrection : [ ]
Mask : [ ]
QMap : [ ]
QxMap : [ ]
QzMap : [ ]
QyMap : [ ]
QrMap : [ ]
PhiMap : [ ]

TwoThetaMap : [ ]
AlphafMap : [ ]
ChiMap : [ ]
Qx1DList : [ ]
Qy1DList : [ ]
Qz1DList : [ ]
TwoTheta1DList : [ ]
Alphaf1DList : [ ]
SolidAngleCorrectedData : [ ]
FigHandle : [ ]
ImFilePath : ‘’
ImFileName : ‘’
ImFileExt : ‘’
ImFileInfo : ‘’
MaskedData : [ ]
MaskedDataStats : [1x1 struct]

ImDim : [0 0]
XWavelength : 1.6869

Public properties that are independent and can be set or accessed by users are

- `Camera`: Type of camera. String must match 'Pilatus', 'MAR165' or 'Other'. Default: 'Pilatus'.
- `PixelSize`: Horizontal and vertical pixel lengths. They are automatically set for Pilatus and MAR165. Unit: mm.
  - Pilatus: [0.172,0.172]
  - MAR165: [0.079138,0.079138]
  - Other: user defined.
- `Beam0`: Direct beam position. Unit: pixel.
- `SDD`: Sample-to-detector distance. Unit: mm.
- `XEnergy`: X-ray energy. Default: 7.35. Unit: KeV.
- `Geometry`: Scattering geometry. 1/[2] for transmission/[reflection].
- `PhiMode`: Range for pixel azimuthal angle with respect to the direct beam. [1]/2/3/4 for $[(-180°, 180°]]/[0°, 360°)/[-270°, 90°)/[-90°, 270°)$.
- `PolarizationMode`: Incident beam polarization. 1/[2]/3/4 for none/[horizontal] /vertical/unpolarized.
- `HorizontalPolarizationFraction`: Fraction that is polarized in the horizontal direction of the beam takes a value between 0 and 1 with 1 for fully horizontally polarized source and 0 for fully vertically polarized source. It applies only to horizontal or vertical polarizations (`PolarizationMode`=2 or 3). Default: 1.
- `IncidentAngle`: Incident angle for reflection geometry. It is not used for transmission geometry, and can set to NaN. Unit: degree.
- `Specular`: Position on the image to define the scattering plane. It can be any point above `Beam0` along the scattering streak. `Specular` is not used for transmission geometry and can be set to [NaN, NaN]. Unit: pixel.
- `PlotCLimsType`: Color-limits-set type for image display. [1]/2 for [manual]/automatic.
- `PlotCLims`: Color limits for image display. Values are set automatically when `PlotCLimsType=2`. Default: [1,2000].
- `PlotScale`: Image display scale option. 1/[2] for linear/[log].
- `PlotAxisLabel`: Axis label option for image display. [1]/2/3 for [pixel]/q/angle.
- `PlotImageFlag`: Flag to specify either `MaskedData` or `SolidAngleCorrectedData` to display. [1]/2 for [`MaskedData`]/`SolidAngleCorrectedData`.
- `ImFile`: Full image file name.
- `RawData`: Raw image data. Dimension is $m \times n$, where m and n are the row and column numbers, respectively.
- `LorentzFactorType`: Type of Lorentz factor for the correction. [1]/2/3/4 for no correction, in-plane (sample surface) randomly oriented 3D structures (ω scan), in-plane randomly oriented 2D structures, and completely randomly oriented objects (such as powders; usually for SAXS and WAXS). Types 2 and 3 are for reflection geometry only.
- `CustomCorrection`: Customized correction matrix to be multiplied to the image data. The correction automatically applies to `SolidAngleCorrectedData`, but not to `MaskedData`. Default is an array of all ones with dimension equal to `RawData`, i.e, `ones(m,n)`.
- `FlatField`: Flat field data to be multiplied to the image data. The correction automatically applies to both `MaskedData` and `SolidAngleCorrectedData`. Default is an array of all ones with dimension equal to `RawData`, i.e, `ones(m,n)`.
- `EfficiencyCorrection`: Parameters for efficiency corrections due to medium (e.g., air) path attenuation and detector sensor absorption. The correction array calculated from `EfficiencyCorrection` is multiplied to `RawData` and the result along with other applied corrections is saved into `SolidAngleCorrectedData`. Default is set for 'Pilatus' camera of 0.32 mm deep silicon sensor and no air absorption:

'N78O21Ar1'	0.0011839	0
'Si'	2.33	0.32

  It is 2×3 cell with the 1st row for air absorption and 2nd row for sensor efficiency. The 1st column specifies the chemical formula, e.g., 'N78O21Ar1' for air. The 2nd column is the mass density in g/cm^3. Last column is the medium path length and sensor depth in mm. Use 'N/A' (not 'NaN', which means sodium and nitrogen) in the chemical formula if no correction is desired.
- `Mask`: Array (*logical*) to define mask (ROI). 1 for ROI or 0 for the rest.
- `QMap`: Q map with respect to the direct beam. All Q maps are in Å^{-1} unit.

- `QxMap`, `QzMap`, `QyMap`, and `QrMap`: Qx, Qz, Qy, and Qr maps. Defined only for reflection geometry.
- `PhiMap`: Polar angle (or azimuthal angle) (Φ) map on detector with respect to the direct beam. All angle maps are in degree unit.
- `TwoThetaMap` and `AlphafMap`: Out-of-plane (parallel to sample surface) angle (2θ) and exit angle (α_f) maps. Defined only for reflection geometry.
- `ChiMap`: Polar angle (χ)of the reciprocal space with respect to the q_z direction (the angle between q_z direction and $\boldsymbol{q}$.
- `Qx1DList`, `Qy1DList`, `Qz1DList`, `TwoTheta1DList`, and `Alphaf1DList`: 1D lists used for axis label. For *gixsdata.m* internal use only.
- `SolidAngleCorrectedData`: Image data after solid angle correction. Intensities of all pixels are corrected regardless of `Mask`. Flat field (if exists), efficiency, and polarization corrections, are also apply automatically and the result is stored in `SolidAngleCorrectedData`.

$$\texttt{SolidAngleCorrectedData} = \texttt{RawData}.*C_s.*\frac{E_m.*E_d}{\max(E_m.*E_d)}.*\frac{1}{P}.*\frac{1}{L}.*\texttt{FlatField}.*\texttt{CustomCorrection},$$

where C_s is the solid angle correction array, E_m and E_d are the efficiency correction for medium attenuation and detector sensor absorption, P is polarization correction factor, and $.*$ is the array multiplication on a pixel-by-pixel basis.

Properties that are dependent on other properties are listed below. Modifications to these properties are forbidden. One should not attempt to modify them.

- `FigHandle`: Figure handle to display image.
- `ImFilePath`, `ImFileName`, and `ImFileExt`: Image file path, name and extension. These properties are derived directly from `ImFile`.
- `ImFileInfo`: Information about the image file. It is obtained from the header of a TIF/TIFF or CBF image file. This property is invalid for other image formats.
- `MaskedData`: `RawData` after applying `Mask`, and `FlatField` (if exists).

  `MaskedData` = `RawData`(masked). * `FlatField`.
- `MaskedDataStats`: Statistics of `MaskedData`. Masked region is not considered. It is a structure with fields:
  - `MinValue` and `MaxValue`: Min and max counts.
  - `MinPixel` and `MaxPixel`: Pixel positions of `MinValue` and `MaxValue`.
  - `TotalValue`: Total count.
  - `MeanValue`: Mean count.
  - `STDValue`: Standard deviation.
  - `MedianValue`: Median count.
  - `NumOfNonZeros`: Number of pixels of non-zero count.
- `ImDim`: Image dimension. It is automatically determined when `RawData` is assigned.
- `XWavelength`: X-ray wavelength. It is automatically determined from `XEnergy`. Unit: Å^{-1}.

To save memory, all image data and maps are stored and handled in *single* data type precision (64 bytes).

Appendix B: List of gixsdata Methods

In addition to Matlab built-in dynamic methods such as `delete`, there are several dynamic and static methods defined for `gixsdata` class. Dynamic methods are called as regular Matlab function. To call static methods, use `obj.solidangle_correction(obj)`. Listed below are all the methods defined for a `gixsdata` object.

- `gixsdata` (dynamic): Create new `gixsdata` object and returns the object handle.

  `obj = gixsdata('image file')` loads image from file into `gixsdata` and return handle `obj`.

  `obj = gixsdata` initializes `gixsdata` with default parameters.

  If handle `obj` exists, image `data` can be directly passed by `obj.RawData = data`.

  `clear obj` only removes the handle. Use `delete(obj)` to delete the `gixsdata` object before clearing the handle `obj`.
- `copyhobj` (dynamic): Duplicates `gixsdata` object.

  `obj2 = copyhobj(obj1)` duplicates `gixsdata` object to another object with a new handle `obj2`. `FigHandle`, `ImFile`, `ImFilePath`, `ImFIleName`,

ImFileExt, and ImFileInfo properties are not duplicated.

Since gixsdata returns the object handle to workspace (or whatever space it is called from), using obj2=obj1 duplicates the handle instead of the object, and both obj1 and obj2 will point to the same object.

- delete (dynamic): Deletes gixsdata object. delete method deletes the object but does not clear the handle. For a complete removal, delete the object and then clear the handle, i.e., delete(obj); clear(obj).

- mask_initialize (static): Initializes the *logical* Mask from RawData. Non-negative elements are denoted by 1 and the rest by 0. User-defined Mask will be lost.

  obj.mask_initialize(obj), where obj is the gixsdata handle.

- qmaps (dynamic): Calculates q and angle maps, 1D lists and SolidAngleCorrectedData. It clears these properties if any of the setup parameters are incomplete or invalid. qmapsneeds to be manually called when setup parameters are changed or updated.

  qmaps(obj), where obj is the gixsdata handle. No output argument is necessary.

- qmaps_initialize (static): Clears all maps, 1D lists, and SolidAngleCorrectedData.

  obj.qmaps_initialize(obj), where obj is the gixsdata handle.

- efficiency_correction (static): Calculates efficiency correction array from property EfficiencyCorreciton. It requires function REFRAC, provided by an independent package *XRAYREFRACTION* (which is also included in *GIXSGUI* distributions), to calculate the attenuation coefficients of the path and detector sensor materials. If the package does not exist or cannot called successfully, an array of all ones is returned without warnings.

  c = obj.efficiency_correction(obj), where obj is the gixsdata handle and c is the correction array having the same dimension as the image data.

- polarization_correction (static): Calculates polarization correction array using polarization parameters from PolarizationMode and HorizontalPolarizationFraction.

  c = obj.polarization_correction(obj), where obj is the gixsdata handle and c is the correction array having the same dimension as the image data.

- lorentzfactor (dynamic): Returns a matrix of Lorentz factor for Lorentz correction. The type of correction has to be specified in LorentzFactorType.

  c_lf = obj.lorentzfactor or c_lf = lorentzfactor(obj), where obj is the gixsdata handle and c_lf is the Lorentz factor array having the same dimension as the image data.

- solidangle_correction (static): Performs solid angle correction to RawData, regardless the Mask. Flat field and efficiency corrections, if defined, are also applied. Given valid setup parameters, the correction is done automatically when RawData is updated. If RawData is set to empty, SolidAngleCorrectedData will be cleared.

  obj.solidangle_correction(obj), where obj is the gixsdata handle.

- imagesc (dynamic): Scales and displays image data.

  imagesc(obj) displays MaskedData or SolidAngleCorrectedData (if exists) in current figure window using plot parameters PlotImageFlag, PlotCLimsType, PlotCLims, PlotScale, and PlotAxisLabel. It overloads Matlab built-in function IMAGESC.

- post_imagesc (dynamic): Updates the current figure with updated image data (either MaskedData or SolidAngleCorrectedData depending on PlotImageFlag) or updated plot parameters.

  obj.post_imagesc(obj), where obj is the gixsdata handle.

- linecut (dynamic): Performs line cut on image data.

  [xdata,ydata] = linecut(obj,xflag,const, nofpts,dataflag) performs line cut on data specified by dataflag for gixsdata of handle obj with predictor parameter xflag, constraints const, total number of linecut points nofpts.

  xflag specifies the predictor parameter. It must be an integer scalar. xflag chart:

  1. q (QMap)
  2. Φ (PhiMap)
  3. q_z (QzMap)
  4. q_x (QxMap)
  5. q_y (QyMap)
  6. q_r (QrMap)
  7. 2θ (TwoThetaMap)
  8. α_f (AlphafMap)
  9. χ (ChiMap)

10. x (horizontal pixel)
11. y (vertical pixel)

`const` specifies the constraints for the linecut. It must be a $m \times 4$ array, where m is the number of constraints. The constraints create a mask superimposed onto `obj.Mask` using logical `AND` operation. If `const=[]` (empty), only `Mask` is used. Each constraint (a row of `const`) has 4 elements: `const = [operator,flag,lower,upper]`, where `operator` has a value of 1 or 2, to specify either `AND` or `OR` logical operation is used for that constraint. `flag` specifies the name of the constraint. It has the same chart as `xflag`. In addition, `flag` can also be 12, meaning that the constraint on that row is not in use. `operator` must be `AND` when `flag=12`. `lower` and `upper` are lower and upper limits of the constraint. They are ignored when `flag=12`. Constraints defined in `const` apply in the following order. First, a mask array of all ones (each element equals 1) is created, which is then applied to the 1st constraint (on the 1st row) using the 1st row's `operator`. This generates a 2nd mask which is then applied to the 2nd constraint on the 2nd row using the 2nd row's `operator`. This goes on and on, till the $(m+1)$th mask is created and applied to the main mask `Mask` using `AND`, generating the final mask which is then used on the image data for the linecut.

`nofpts` is the total number of predictor points for the linecut.

`dataflag=1` or `dataflag=2` specifies either `MaskedData` or `SolidAngleCorrectedData` is used for the linecut. If `SolidAngleCorrectedData` does not exist, i.e., no maps, `MaskedData` will be used regardless of `dataflag`, and the linecut can only be performed with `xflag` having a value 10 or 11.

Example: `[xdata,ydata] = linecut(obj,3, const,500,2)` performs a linecut against q_z (`xflag=3`) on `obj.SolidAngleCorrectedData`. There are 500 points in the 1D line profile (number of points might be less if the step size in the xdata spanning is too small that some xdata points are not covered in the map). The constraints, `const`, are

1	5	0.08	0.09
2	8	0	0.5
1	12	NaN	NaN

There are three constraints defined in `const`, with the last one not in use (`flag=12`). The linecut is performed in regions with $0.08\text{Å}^{-1} \leq q_y \leq 0.09\text{Å}^{-1}$ `or` $0^\circ \leq \alpha_f \leq 0.5^\circ$.

- `get_cmask` (static): Calculates mask defined by linecut constraints `AND Mask`.

  `cmask = obj.get_cmask(obj,const)`, where `obj` is the `gixsdata` handle, and `const` is the linecut constraints. It returns a logical array `cmask` to be used for the linecut.

- `reshape_image` (dynamic): Converts image into Cartesian coordinate system.

  `[xdata,ydata,imgdata,countdata] = reshape _image(obj,params,dataflag)` converts image data specified by `dataflag` for a `gixsdata` oject of handle `obj` using reshape parameters `params` to a new image named `imgdata` whose x and y data are given by `xdata` and `ydata`. `countdata` is an array to store the number of pixels in the source image data array that have the x and y step values specified in `params` for the new reshaped image. `countdata` has the same dimension as `imgdata`.

  `params` is a structure with fields `X`, `Y`, `XNOfPts`, `YNOfPts`, `XRange` and `YRange`. Here, `X` and `Y` use the same chart as `xflag` does in method `linecut` to specify the names of the x and y axis. `XNOfPts` and `YNOfPts` are the numbers of points in `xdata` and `ydata`. `XRange` and `YRange` are the ranges of `xdata` and `ydata`. Here is a usage example: $\gg$`params`

  X : 6
  Y : 3
  XNOfPts : 500
  YNOfPts : 600
  XRange : [0, 2.3]
  YRange : 0, 1

  will reshape the source image data into a 600 (V) $\times$ 500 (H) image with $0\text{Å}^{-1} \leq q_r \leq 2.3\text{Å}^{-1}$ and $0\text{Å}^{-1} \leq q_z \leq 1\text{Å}^{-1}$ being the x and y axis, respectively.

  `dataflag` has the same usage as in method `linecut`.

- `findpeak` (dynamic): Finds 2D diffraction peak positions.

  `[peak,fresult,vertex] = findpeak(obj, vertex,dataflag,fitflag,startpoint,opts)` finds peak position of a region of interest (ROI) defined in a polygon. The vertexes of the polygon (x and y pixels) are defined by `vertex` (two column vector with the 1st and 2nd columns for x and y pixels).

  `dataflag` has the same usage as in method `linecut`.

  `fitflag=1` or `fitflag=2` specifies either COM or BVND model is used to determine the peak position.

  `startpoint` (1x9 vector) gives the initial values of the BVND parameters for the fitting. These

parameters are `[A, Cx, Cy, C, Rho, Sigma_x, Sigam_y, X0, Y0]`. `startpoint` can be `[]` (empty) for automatic assignment. It is not used for the COM model.

`opts` gives the optimization parameters for the BVND fittings. It can be defined using optimset function. Refer to `fminsearch` (Matlab built-in function) and `fminsearchbnd` (by John D'Errico and included in the package) for usage. It can be `[]` (empty) for automatic assignment. It is not used for COM model.

`findpeak` returns the peak position (in x and y pixels) and its map values in `peak` (a structure). The fitting parameters and results are given in `fresult` (structure). The vertexes for the ROI polygon are passed to `vertex`.

- `recount` (dynamic): Calculates the intensity of the pseduo point counter defined by the constraints and mask (ROI).

  `[I, roimap, mapname, pcounts, cmask] = roicount(obj, const, dataflag)` returns the total intensity I (scalar), where `obj` is the `gixsdata` handle and `const` for is the constraints. Use `[]` for `const` if no constraints are applied. `dataflag` has the same usage as in `linecut`.

  On the output side, `I` is the integrated intensity over the ROI. `roimap` is a list (4x1 or 11x1 depending on the scattering geometry of either transmission and reflection) of the averaged map values in the ROI. The order of the list follows `xflag` defined in method `linecut`. `mapname` is the list of corresponding map names. `pcounts` is the number of pixels in the ROI. `cmask` is the logical matrix representing the ROI (constrained mask).

Appendix C: List of Functions for 3D Nano-structure Indexing

3D nano-structure indexing is a sub-package independent of *GIXSGUI* toolbox which search for allowed grazing-incidence x-ray diffraction positions of polycrystalline (2D in-plane powders) films of a given space group or unit cell. On the other hand it also integrated with *GIXSGUI* so that the indexing can be done directly on GIXS patterns from *GIXSGUI*. Listed below are functions in the indexing sub-package. The conventions of the terminology follow the book by Marc De Graef and Michael E. McHenry, *Structure of Materials: An Introduction to Crystallography, Diffraction and Symmetry*, 1st Edition, Cambridge University Press (2007).

- `crystaldirectmetrictensor`: Calculate the crystal metric tensor.

  `g = crystaldirectmetrictensor(a, b, c, alpha, beta, gamma)` calculates the metric tensor for a three dimensional crystal system. `(a,b,c)` are the length of three basis vectors of a Bravais lattice. `alpha` is the angle between `b` and `c`; `beta` is the angle between `a` and `c`; `gamma` is the angle between `a` and `b`. All angles are in degree.

- `crystalreciprocalmetrictensor`: Calculate the reciprocal metric tensor.

  `g = crystalreciprocalmetrictensor(a, b, c, alpha, beta, gamma)` calculates the reciprocal metric tensor for all three dimensional crystal systems in the reciprocal space.

- `crystaldirectstructruematrix`: Calculate the direct structure matrix.

  `A=crystaldirectstructruematrix(a, b, c, alpha, beta, gamma)` calculates the direct structure matrix for the coordinate transformation from Bravais lattice to Cartesian coordinate frame. Then a lattice vector `p`=$[uvw]$ in the Bravais lattice can be expressed as `A*p'` in the Cartesian coordinate frame.

- `crystallatticeangle`: Calculate the angle between lattice vectors or planes.

  `theta = crystallatticeangle(p1, p2, g)` calculates the angle between two lattice vectors with coordinates `p1` and `p2`, or lattice planes with plane index `p1` and `p2`. For angle between vectors, `p`=$[u, v, w]$, and `g` is the real space metric tensor of the Bravais lattice. For angle between planes, `p` is the miller index (h, k, l) for the plane, and `g` is the reciprocal space metric tensor.

- `crystalplanedistance`: Calculate the crystal plane distance.

  `d=crystalplanedistance(p, g)` calculates the plane distance of parallel planes with miller index `p`=(h, k, l). `g` is the reciprocal metric tensor.

- `crystalvectorlength`: Calculate the length of a vector.

  `t=crystalvectorlength(p,g)` calculates the length of vector. In real space, `p`=$[u, v, w]$ is the coordinate of the vector and `g` is the metric tensor. In reciprocal space, `p`=(h, k, l) is the miller index and `g` is the reciprocal metric tensor.

- `gvector`: Calculate the reciprocal lattice vectors.

  `G = gvector(a, miller)` calculates the reciprocal lattice vectors of `miller` indices (1x3 vector) for a real space lattice with lattice vectors `a`=(a_1, a_2, a_3) whose coordinates are defined in a Cartesian reference system.

- `latticetype`: Return the lattice type (one of the 7 types) for a space group.

- `sgname`: Return the name of the space group. For multiple cell/axes choices, only one choice is used which is explicitly specified.

- `sgreflection`: Determine the reflection condition of a miller index set.

  `b = sgreflection(miller, sg)` determines if the diffraction is allowed for a miller index set (h, k, l) for space group number `sg`. It returns 1 if reflection is allowed and 0 if forbidden. Reflection conditions are taken from International Tables for Crystallography Volume A. Only general rules are applied. In presence of multiple cell/axes choices, the choice is specified is `sgname`.

- `gixsdiffpos`: Calculate the grazing-incidence x-ray diffraction positions of polycrystalline films.

  `result = gixsdiffpos(lattice, sguvw, orientation, orientationmethod, hlist , klist, llist, k0, alpha_i, nfilm, qdeadband, qcutoff)` calculates the GIXS diffraction positions from a nano-structred film. Polycrystals in the film need to have no preferred orientation in the plane of the substrate. Both Born approximation (BA) and distorted wave Born approximation (DWBA) are used to calculate the angle and q positions of allowed diffractions.

  `lattice = [a, b, c, alpha, beta, gamma]` defines the unit cell.

  `sguvw` defines the symmetry of the lattice. It is either an integer (1∼230) for the space group, or an array (nx3) to define the lattice basis with n lattice "atoms" positioned at fractional coordinates [uvw] in the unit cell.

  `orientation`(1x3) and `orientationmethod` define the unit cell direction that is normal to the substrate surface. `orientationmethod`=1 uses the Cartesian sample reference frame to define this direction. The initial orientation of the unit cell is layer out with respect to the substrate surface and the forward direction of the incident beam such that the `ab` plane is on the substate plane and the lattice vector `a` and `b` are along and perpendicular to the forward direction of the incident beam, respectively, i.e., along x and y directions defined in the sample reference frame (Fig. 2). After the `orientation` is defined by an $[x, y, z]$ direction in the Cartesian sample reference frame, the lattice will be automatically rotated so that the defined direction is along the substrate normal, and then the orientation of polycrystals is averaged over all in-plane angles for completely random in-plane orientations. A more convenient method to define the unit cell orientation is to do it in the lattice reference frame, i.e., `orientationmethod`=2, where `orientation` is specified by $[uvw]$, the indices of the crystallographic direction of a lattice.

  `hlist`, `klist`, and `llist` are lists of h, k, l values, specifying the range to calculate the diffraction positions. `k0` is the wave vector of the incident beam. `alpha_i` is the incident angle in degree. `nfilm` is the average index of the refraction of the film, which contains both real and imaginary parts for refraction and absorption. `qdeadband` specifies the q range within with the diffractions are considered a multiplication, i.e., one diffraction corresponds to multiple (hkl) values. Typical values are $1e{-}8$ Å^{-1}. `cutoff` specifies the q cutoff below which the solved absolute $q = (qx, qy, qz)$ values are set to zero. This is useful for large lattices that may give many very small qx, qy or qz values as a result of the precision of equation solving algorithm. Typical values is $1e-6$ Å^{-1}.

  The fields of the output `result` structure are described below:

  - `miller_ba_full`: cell (nx1)containing $[hkl]$ values of allowed reflections calculated in Born approximation (BA). Multiple equivalent $[hkl]$ values may exit and are listed within each cell element.
  - `q_ba_full`: corresponding list (nx3) of q values (column 1/2/3 for $q_x/q_y/q_z$, respectively).
  - `q_ba`: `q_ba_full` with negative q_z values excluded.
  - `miller`: cell (nx1)containing $[hkl]$ values of allowed reflections calculated in distorted wave Born approximation (DWBA). Multiple equivalent $[hkl]$ values may exit and are listed within each cell element.
  - `angle_t` and `angle_r`: list (nx2) of corresponding angles (column 1/2 for in-plane/exit angles)for transmission and reflection channels, respectively.
  - `q_dwba_t` and `q_dwba_r`: list (nx3) of corresponding q (column 1/2 for $q_x/q_y/q_z$)for transmission and reflection channels, respectively.

Appendix D: Function for ROI Scan

- `roiscan`: Extract integrated intensity over a user-defined ROI (or virtual detector slit) over a set of images.

  `scanresult = roiscan(filepath, filename, scanvarname, scanvarlist, params, const , dataflag, plotflag, cmaskdisplayflag)` extracts ROI intensity for a set of images. `roiscan` calls the built-in method `roicount` of `gixsdata`. `filepath` and `filename` are the path and list of names (in a nx1 cell array). `scanvarname` is the variable during the scan (e.g., `IncidentAngle`),

which has to math the property name of `gixsdata` if it is actively assigned during the ROI scan. `scanvarlist` is the corresponding list (nx1) of the scan variable (e.g, incident angle values), the total number of which has to match that of `filename` on a one-to-one basis. `params` is the *gixsdata* parameter. `const` is the constraint to define the ROI, and it is an nx1 cell array, following the usage of `linecut` method. `dataflag=1` or `dataflag=2` specifies either `MaskedData` or `SolidAngleCorrectedData`. `plotflag=0/1` specifies whether the scanned image and reduced scan data are displayed and updated during the ROI scan data reduction. `cmaskdisplayflag=0/1` specifies whether a box inclosing ROI is drawn.

`scanresult` is the output consisting of the following fields:

- `X` and `I`: lists (nx1) of the scan dependent variable `scanvarlist`, and ROI intensity, respectively.
- `mapname`: list of map names for the experiment geometry. It is 4x1 for transmission or 11x1 for reflection.
- `roimap`: list (nx4 or nx11) of averaged map values in the ROI for each image.
- `pcounts`: list (nx1) of the number of pixels in the ROI for each image.

Appendix E: Frequently Asked Questions (FAQ)

This section is contributed by Dr. Joseph Strzalka at 8ID/APS.

1. **Q**: How to load a *.mat parameter file and assign the parameters to a `gixsdata` object in a script mode?

   A: One has to first load a parameter file, which is a `gixsdata` object with empty `RawData`, `MaskedData` and `SolidAngleCorrectedData`. Duplicate the parameter `gixsdata` object and assign a image file to its `ImFile` or pass the image data directly to `RawData`.

```
foo=load('gixs_params.mat); % load parameter file
obj=copyhobj(foo.params);   % duplicate object
obj.ImFile='demo.tif';      % load image file
obj.RawData=data;           % or pass image data
```

2. **Q**: I used the Gap Fill tool to merge data from two images and saved the gap-filled image, but when I load the data into *GIXSGUI*, I still see the gaps. How do I view the data without gaps?

   A: After you load the gap-filled data, click the button "Clear Mask" in the "Mask Tools" area. If the data still does not display correctly, it is because the flat field correction is being applied to the data that was already flat-field corrected in the gap fill procedure. Click the button "Remove FF" in the "Image Corrections" area.

3. **Q**: How do I use the Gap Fill tool?

   A: Click the "Gap Fill" button on *GIXSGUI* to open the Gap Fill tool.

Load the Flat Field: `~/GIXSGUI/pilatus_1mf/flatfield/FF_p1m0108_E11p9_T9p9_vrf_m0p3.tif`

Load the Gap Mask: `~/GIXSGUI/pilatus_1mf/gapfill/gapmask_extra5row.tif`

Load the Bad Pixel Mask: `~/GIXSGUI/pilatus_1mf/gapfill/pilatus1mf_badpix_20110110.tif`

For the Base Image, load an image with the Pilatus in the "pildown" position. For the Vertical Offset Image, load the corresponding image with the Pilatus in the "pilup" position. Change the offset to X=0, Y= 23 (The offset between the "pilup" and "pildown" positions is 3.956 mm=23 pixels×0.172 mm/pixel). Click the "Fill" button. Export the data to a file or to the Matlab workspace using the buttons in the Export area.

4. **Q**: How do I apply the flat field correction?

   A: The flat field correction is described in section III C 3. When you start *GIXSGUI*, the initial flat field data is a dummy file that is completely uniform, an array with every element equaling 1.

You can load a non-uniform flat field data in two ways:

(a) Use the "Load FF File ..." button.

(b) Use the "Load Params ... " button and load a parameter file containing flat field data.

There are four buttons in a row that affect the flat field correction.

- "Flat Field" generates a plot of the flat field data currently being used. This lets you check what data you are using (or whether you have flat field data loaded. The dummy uniform data will appear as a blank image).
- "Remove FF" removes the flat field data, replacing it with an array where every element is 1.
- "Default" is the inverse of "Remove FF" in the following sense. If you loaded flat field data and set it as part of the default parameters (using the "Set As Default" button), then "Default" restores that flat field data as the current flat field data.
- "Load FF File ..." initiates a dialog that lets you select the flat field data.

The flat field correction is always applied and the data you see always has a flat field correction applied. When the current flat field data is just the uniform dummy file, the correction does not change the data and the image appears the same as it would uncorrected.

In the *GIXSGUI* distribution, the folder `~/GIXSGUI/pilatus_1mf/flatfield` contains three files provided by the manufacturer. `FF_p1m0108_E11p9_T9p9_vrf_m0p3.tif` is the best one to use for the standard energy of 7.35 keV at 8ID/APS/ANL.

The flat field correction is also important for the Gap Fill image tool, which can merge together two images collected with a detector offset, so that the inactive pixels in the gaps between modules can be filled. This only works properly when each image is flat-field corrected before the composite gapfilled image is created. Then, since the gapfilled data has already been flat-field corrected, you will want to revert to the dummy uniform flat field data when you view it (see FAQ #2).

5. **Q**: What are the mask functions for and how do I use them?

   A: Fundamentally, you can think of the mask as letting you exclude parts of the data you don't care about (like the inactive pixels in the gaps between modules), and/or, you can think of the mask as letting you focus on the parts of the data that you do care about (for instance, your scattering is forming a ring and you only want to look at that ring).

   The Mask Tools are accessed from main GUI window (see Fig. 8).

   - "Show Mask" displays the current mask — similar to "Flat Field" button for the Flat Field Tools.
   - "Clear Mask" sets the current mask to a dummy mask, an array with all elements equal to one — similar to "Remove FF".
   - "Default" resets the current mask to the file stored as the default — similar to "Default" for the Flat Field Tools.
   - "Load Mask File ..." initiates a dialog to let you input a file with mask data.
   - "Export Mask ..." initiates a dialog to let you save the current mask data to a file.
   - "Include ..." lets you use the mouse to define a mask interactively. Click the "Include ..." button and use the mouse to left-click on a series of points that define the region of interest. For the last point, you can either left-click and then use the "Enter" button on your keyboard, or you can right-click on it. The display will change to show you the data with the excluded region shown in blue.
   - "Exclude ..." works in a similar way as "Include ...", but excludes user-defined region.

   Usually, bad pixels and inactive pixels in the gap are marked with negative values so that they are easy to mask out from the data.

6. **Q**: How do I automate the fitting for diffraction peak positions?

   A: You have to do it in the script mode. First, you need to locate the peak positions. This is can be done by defining estimated boxes/ROIs around the peaks you believe to be diffractions. The vertices of these ROIs are then fed into `findpeak` (see Appendix B for usage) for automatic fitting for peak positions (the output is corresponding q and angle values of the peaks). Secondly, you try to guess the space group and lattice parameters in *GIXSGUI*. Once the scope is narrowed, you can use `gixsdiffpos` in your script to calculate the peak positions to compare with the experiment values you have determined in the first step. Recursive comparisons can be done in a fitting code using standard optimization algorithms such as *lsqcurvefit* and *fminsearch* in the Matlab Optimization Toolbox, or *ga*, *patternsearch*, and *simulannealbnd* in the Matlab Global Optimization Toolbox (additional licenses are required).

7. **Q**: Which Matlab distributions are compatible with *GIXSGUI*?

   A: *GIXSGUI* works with Matlab 7.10 (R2010a) and later versions. No extended Matlab Toolboxes are required except the SDD Calibration and 3D Indexing functions (require Matlab Optimization Toolbox). In addition, Matlab Curve Fitting Toolbox is required if the linecut data is to be directly fitted in Matlab. Early versions do not support all the functions used in *GIXSGUI*. The R2011a release has a bug in the `imfinfo` function. While it is expected to be fixed in future Matlab distributions, Matlab has issued a patch as a work around, which can be acquired upon request from 8ID/APS/ANL staff. The graphical compatibility issue with Matlab R2014b has been resolved.

8. **Q**: Do you provide any templates or examples for the script mode?

   A: Yes. There are several commented *.m* script files in `~\GIXSGUI\script` to show you how to work in the script mode.

9. **Q**: What if I run into problems with the software or it does not have the feature I need?

   A: You can always contact us. We actively maintain the package and will be happy to implement new features upon the community's need.

Appendix F: Use of Third-party Codes

The following codes were downloaded from Matlab File Exchange and their usage are subjects to BSD licenses.

- `cmapline.m` Copyright (c) 2010, Andrew Stevens. All rights reserved.
- `CircleFitByTaubin.m` Copyright (c) 2009, Nikolai Chernov. All rights reserved.
- `fminsearchbnd.m` Copyright (c) 2006, John D'Errico. All rights reserved.