Documentation for GIXSGUI - 1.2

Zhang Jiang

X-ray Science Division

Argonne National Laboratory

zjiang@aps.anl.gov

(Dated: July 28, 2011)

CONTENTS

I. Overview 3

II. Installation 3

III. GIXSDATA 3

A. Load image 5

B. Q and angle maps 6

C. Corrections 7

1. Efficiency correction 7

2. Flat field correction 8

3. Solid angle correction 9

D. Linecut 10

E. Reshape image 10

IV. GUI mode (GIXSGUI) 10

Appendices 15

A. List of GIXSDATA Properties 15

B. GIXSDATA Methods 20

C. Frequently Asked Questions (FAQ) 24

D. Use of Third-party Codes 27

E. Future Release 27

I. OVERVIEW

GIXSGUI (Grazing-incidence X-ray Scattering Graphical User Interface) is a Matlab package to visualize and perform preliminary processing on x-ray scattering data. While both transmission and reflection geometries are supported, the package is more focused on handling grazing-incidence data. This package is designed for sector 8-ID-E at the Advanced Photon Source, Argonne National Laboratory. It should also be compatible with GIXS data taken at other places. Currently supported image formats include TIF, CBF (Crystallographic Binary File) and MAT (Matlab).

GIXSGUI package contains two main *.m files: GIXSGUI.M and GIXSDATA.M. GIXSDATA.M defines a handle (analogue of pointer in C/C++) class called gixsdata. GIXSGUI package provides two modes to access and process data: script mode and GUI mode. In the script mode, one can perform operations, such as parameter setup, data import, storage, process, duplication and export, by applying gixsdata class methods to its class properties. Alternatively, one can use GIXSGUI.M, which provides a user-friendly interface to access all these operations.

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, copy the full 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

GIXSDATA is a handle class defined in file GIXDATA.M to visualize and process GIXS data. 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.

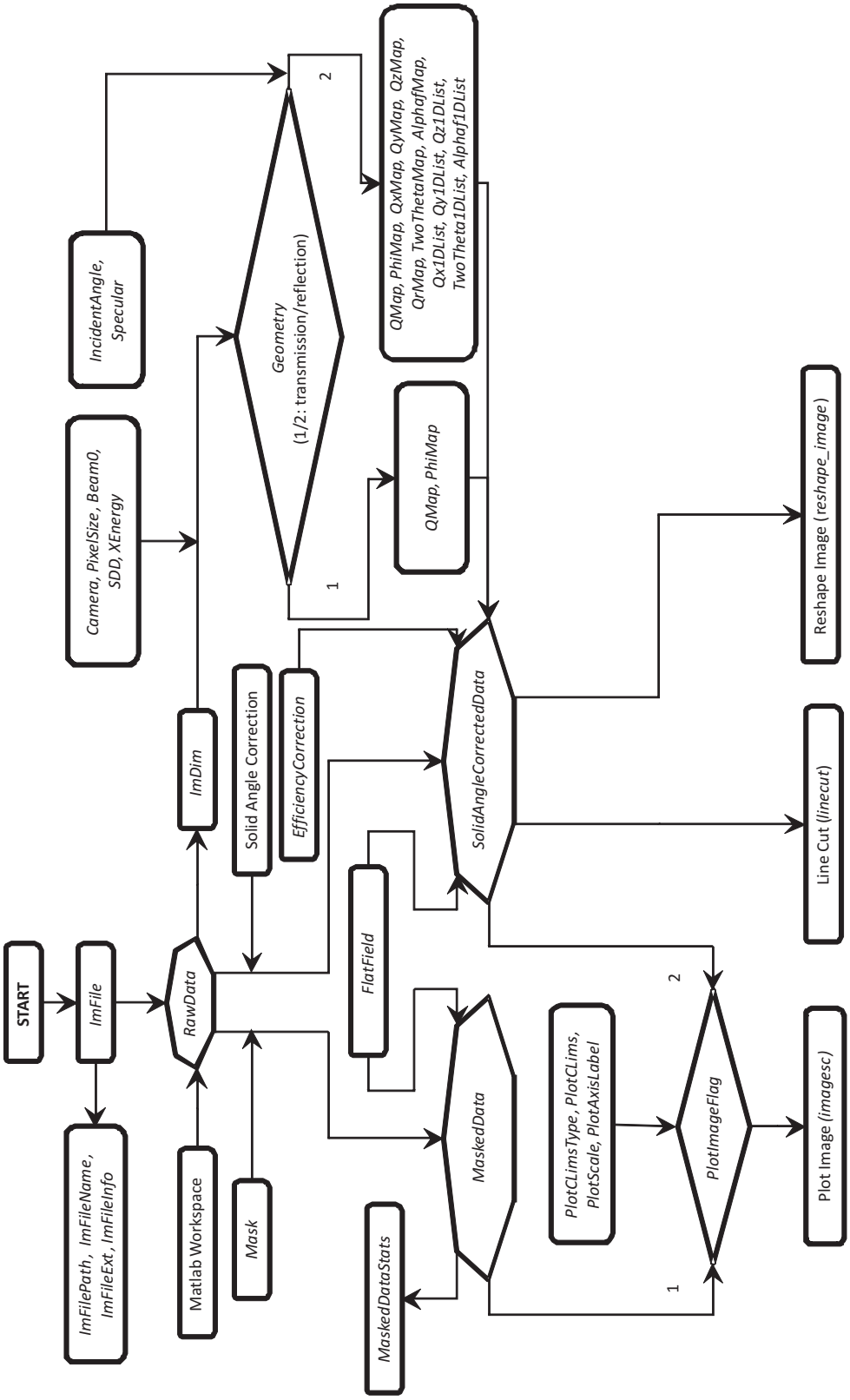

FIG. 1. Data flow in a gixsdata object.

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.

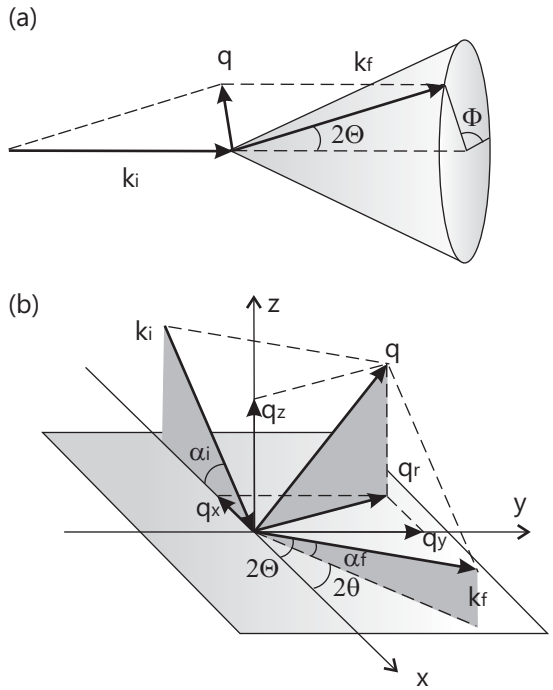

FIG. 2. (a) Transmission geometry. (b) Reflection geometry.

- 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, and efficiency 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 $\mid \boldsymbol{k}_i \mid = \mid \boldsymbol{k}_f \mid = 2\pi/\lambda$ 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, $q = \sqrt{q_z^2 + q_x^2 + q_y^2}$.

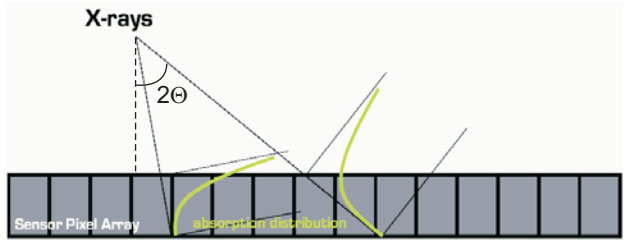

FIG. 3. Pixels with larger oblique angles have more absorption probability. Figure source: Dectris Ltd.

- Φ: Pixel azimuthal angle with respect to the direct beam pixel on detector.
- 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, $q_x = k_f \cos\alpha_f \cos 2\theta - k_i \cos\alpha_i$.
- 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

1. *Efficiency correction*

The efficiency correction arises from two parts: medium (e.g. air) path attenuation and detector sensor absorption. Both are important and strongly recommended for GIXS measurements with a short sample-to-detector distance, e.g., grazing-incidence wide-angle x-ray scattering (GIWAXS) and grazing-incidence x-ray diffraction (GIXD).

Pixels with larger oblique angles, 2Θ, have larger path length from sample to detector. The medium attenuation of the x-rays is thus larger, and this effect has to be corrected by normalizing the measured pixel intensity with respect to the pixel path length. The correction array is calculated by

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

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

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

where L_x and L_y are horizontal and vertical pixel sizes, x_0 and y_0 are direct beam position on detector.

Similarly, the detector sensor absorption correction comes from various x-ray absorption probabilities across the pixels of sensor arrays. X-rays impacting at different oblique angles have different path lengths in the detector sensor. Therefore, the probability of stopping and detecting x-rays increases as the angle increases (Fig. 3). The correction array is calculated by

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

where μ_d is the linear mass attenuation coefficient of the detector sensor material (unit: 1/mm), and t_d is the thickness of the sensor (unit: mm).

To set up efficiency correction parameters, one needs to construct a 2×3 cell for `EfficiencyCorrection` (defined in section A). The total efficiency correction array is calculated by multiplying E_m and E_d pixel by pixel, followed by a normalization to its maximum 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 3). It does not apply `MaskedData`.

2. 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

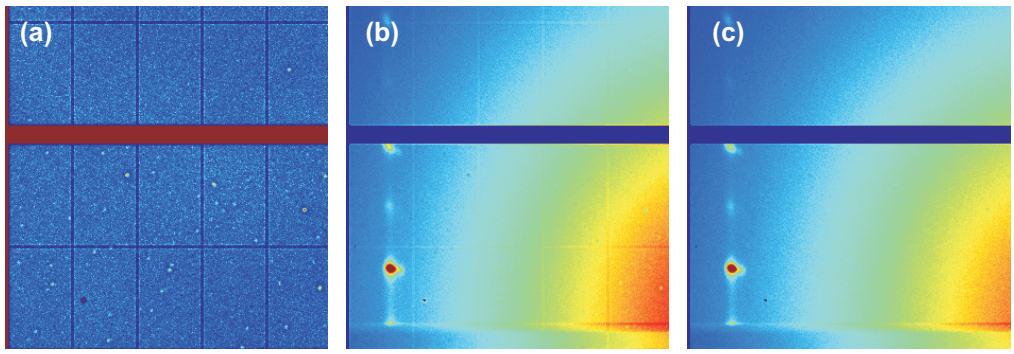

FIG. 4. (a) Flat field data of Pilatus 1MF camera. (b) Before correction. (c) After correction.

always multiplied by the flat field correction array. One can assign the flat field data using

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

Figure 4 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.

3. Solid angle correction

The measured intensity on a pixel is proportional to the solid angle subtended by the pixel. Pixels of larger oblique angle have smaller solid angles, therefore, the intensities on all the pixels should be rescaled to an identical solid angle, e.g., that of direct beam pixel. This correction is purely a geometrical effect. The correction array is given by

$$C_s = \Delta\Omega_0/\Delta\Omega_i,$$

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 and efficiency corrections, if they exist, 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`.

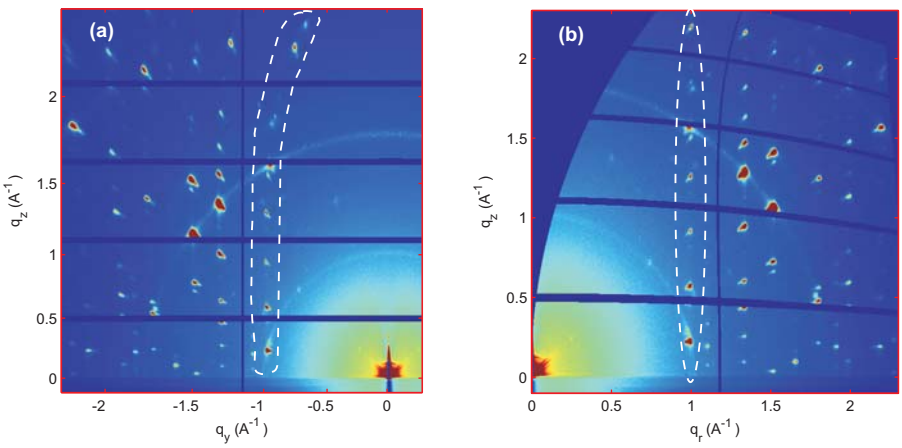

FIG. 5. (a) Intensity image in (q_z, q_y). (b) Reshaped intensity image in (q_z, q_r).

D. Linecut

2D image data is usually reduced to 1D line profiles for further analysis. Class method `linecut` can be used do this. Constraints are often added to specify region of interest (ROI) for the linecut. See appendix B for usage.

E. Reshape image

The intensity measured on an area detector is a projection of the Ewald sphere. This results in distortions. The 2D intensity image coordinates, e.g., (q_z, q_y), are not necessarily Cartesian, especially for smaller sample-to-detector distance. For example, Bragg peaks at q_y may not appear as straight lines in the image. Method `reshape_image` provides a routine to convert the image into one with Cartesian coordinates. For example, a film sample with in-plane orderings has diffractions peaks with a constant q_r value. However, these peaks align on a curve, corresponding to the constant q_r, when the intensity image is displayed in (q_z, q_y), as shown in Fig. 5a. After reshaping the image into a Cartesian coordinate system of (q_z, q_r), these peaks align vertically (Fig. 5b).

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 use GIXSGUI,

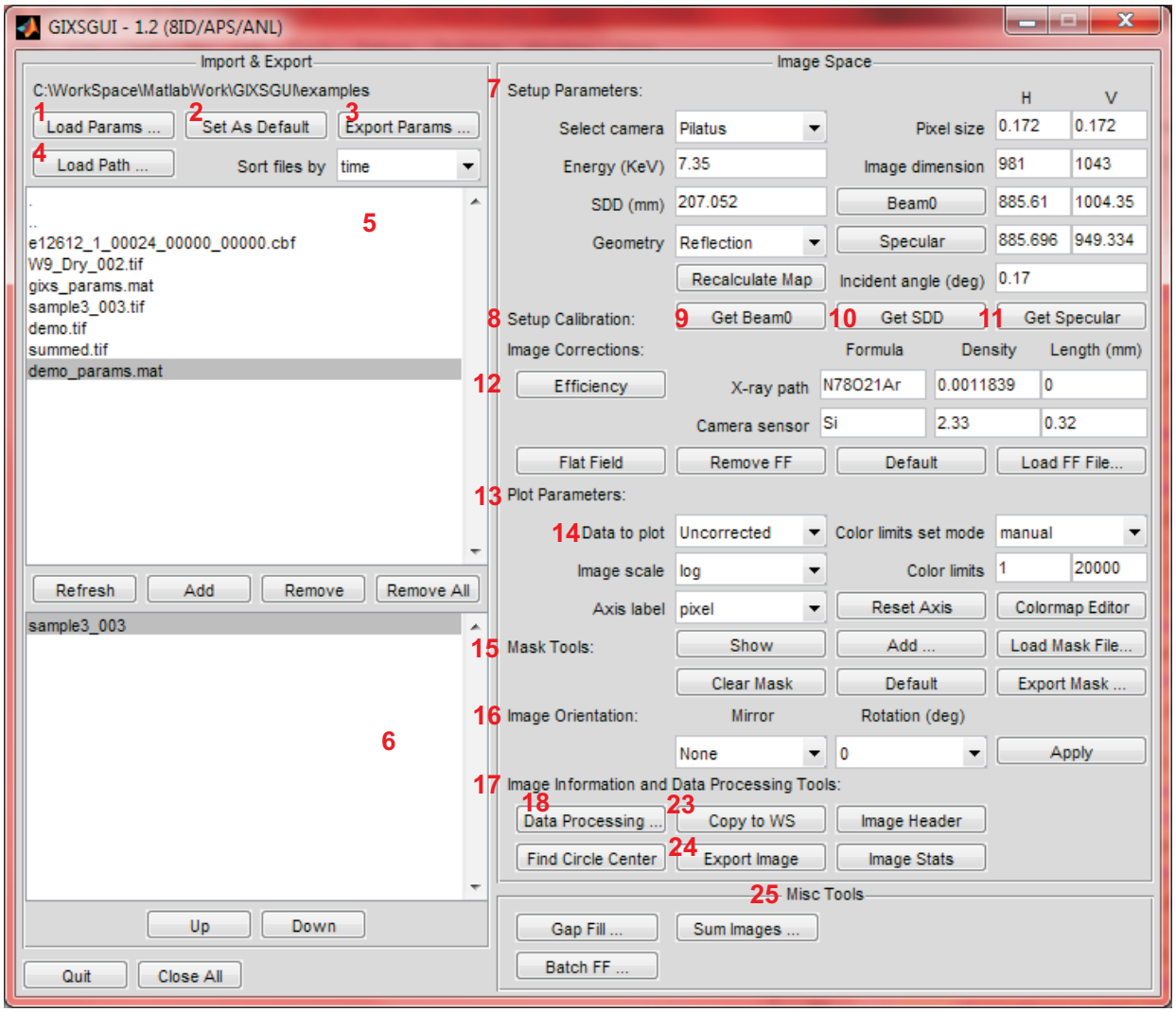

FIG. 6. GIXSGUI graphical user interface.

a graphical user interface provided by GIXSGUI.M, to do the same job. Figure 6 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" (label 1) created previously by "Export Params" (label 3). "Set as Default" (label 2) 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 historic Matlab package to view Mar165 GISAXS data (2048×2048), one can load *.par parameter file.

- Locate image file path by "Load Path" (label 4). Supported image files with .tif, .mat, and .cbf extensions, as well as subfolders, are displayed in the path list box (label 5). Double click image file, or single click followed by "Add", to load into image space listed in the list box (label 6).

- "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" (label 9) to draw a rectangular box around the direct beam to find the direct beam pixel.

- Use "Get SDD" (label 10) 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.

- Use "Get Specular" (label 11) 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 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.

- Region 12 defines 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.

- Region 13 defines parameters for plotting images. "Data to plot" (label 14) 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.

Uncorrected : `MaskedData` (including flat field correction)

Corrected : masked `SolidAngleCorrectedData` (including flat field, efficiency and solid angle corrections)

- Region 15 defines `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 in region 16. 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.
- Region 17 provides tools 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 ..." (label 18) to open a new GUI (Fig. 7) to do linecut and image reshaping.
- Two ways of reducing 2D data to 1D line profiles are implemented: free linecut (region 19) and constrained linecut (region 20). 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 (region 21).
- Region 22 is used to reshape image to one in orthogonal Cartesian coordinates.
- Duplicate `gixsdata` of the current image into Matlab workspace by "Copy to WS" (label 23). "Export Image" (label 24) exports `MaskedData` of the current image to an int32 tif file or Matlab mat file.

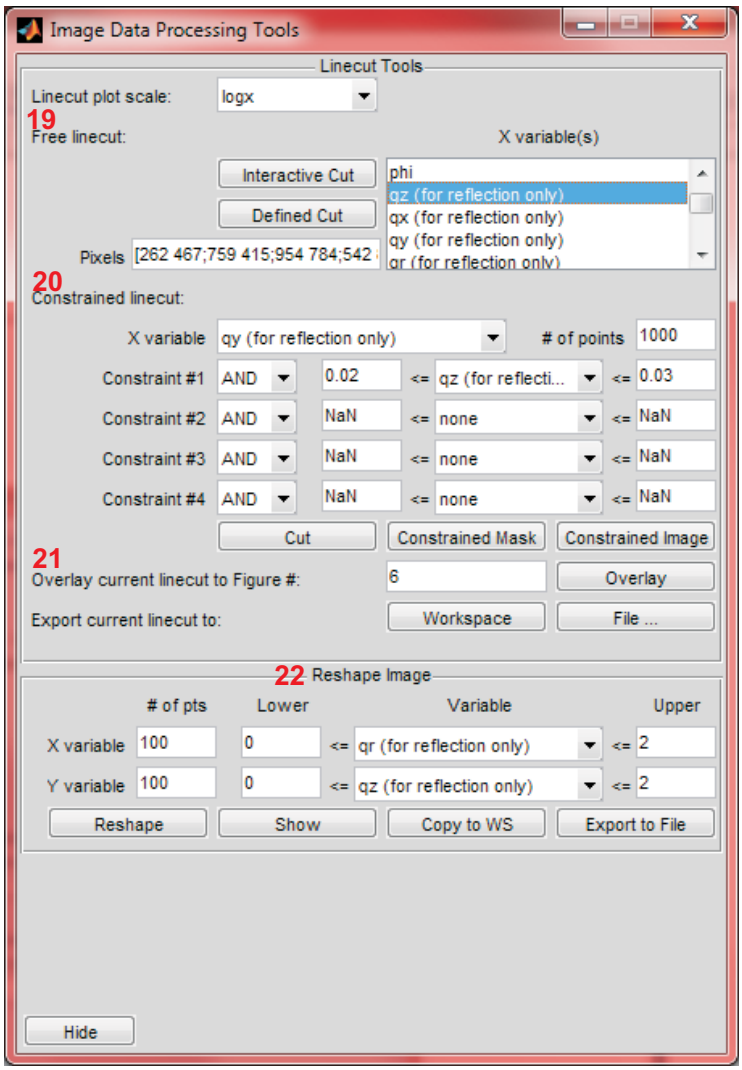

FIG. 7. GUI for image data processing tools.

- "Close All" closes all figures derived from GIXSGUI.

GIXSGUI provides a number of miscellaneous tools (region 25) 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.

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
  IncidentAngle : NaN
     Specular : [NaN NaN]
  PlotCLimsType : 1
     PlotCLims : [1 2000]
    PlotScale : 2
  PlotAxisLabel : 1
  PlotImageFlag : 1
       ImFile : ''
      RawData : [ ]
     FlatField : [ ]
  EfficiencyCorrection : [ ]
         Mask : [ ]
         QMap : [ ]
         QxMap : [ ]
         QzMap : [ ]
         QyMap : [ ]
```

QrMap : [ ]

PhiMap : [ ]

TwoThetaMap : [ ]

AlphafMap : [ ]

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].

- `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.

- `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 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`: 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.

- `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 and/or efficiency corrections, if they exist, are also apply automatically and the result is stored in `SolidAngleCorrectedData`.

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

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, and $.*$ is the array multiplication in 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 MAT image file.
- `MaskedData`: `RawData` after applying `Mask`, and `FlatField` (if exists).

$$\texttt{MaskedData} = \texttt{RawData}(\text{masked}).*\texttt{FlatField}.$$

- `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: 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. `qmaps`needs 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.

- `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. x (horizontal pixel)
10. 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 11, meaning that the constraint on that row is not in use. `operator` must be `AND` when `flag=11`. `lower` and `upper` are lower and upper limits of the constraint. They are ignored when `flag=11`. 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 9 or 10.

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	11	NaN	NaN

There are three constraints defined in `const`, with the last one not in use (`flag=11`). 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: ≫`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`.

Appendix C: 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 2. 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 by six buttons in region 15 of the GUI (as defined on Fig. 6).

   - "Show" 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.
- "Add ..." lets you use the mouse to define a mask interactively. Click the "Add ..." 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.

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**: Which Matlab distributions are compatible with GIXSGUI?

   A: GIXSGUI works with Matlab 7.10 (R2010a) and later versions. No extended Matlab Toolboxes are required. 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.

Appendix D: Use of Third-party Codes

Codes in `cmapline.m` (Copyright (c) 2010, Andrew Stevens, All rights reserved) and `CircleFitByTaubin.m` (Copyright (c) 2009, Nikolai Chernov, All rights reserved) were obtained from Matlab File Exchange and are protected by BSD license.

Appendix E: Future Release

- Q and phi partition with respect to the reflected beam.