JOURNAL OF APPLIED CRYSTALLOGRAPHY

ISSN 1600-5767

Received 5 December 2018
Accepted 18 April 2019

Edited by A. Barty, DESY, Hamburg, Germany

Keywords: serial crystallography; X-ray free-electron lasers; XFELs; data visualization; data analysis; Python.

ClickX: a visualization-based program for preprocessing of serial crystallography data

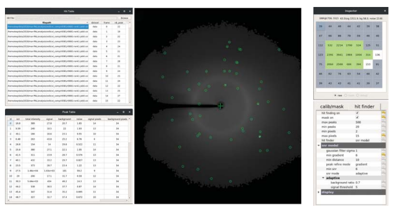
Xuanxuan Li,[a,b] Chufeng Li[c] and Haiguang Liu[b]*

[a]Department of Engineering Physics, Tsinghua University, Beijing 100084, People's Republic of China, [b]Complex Systems Division, Beijing Computational Science Research Center, ZPark II, Haidian, Beijing 100193, People's Republic of China, and [c]Department of Physics, Arizona State University, Tempe, AZ 85287, USA. *Correspondence e-mail: hgliu@csrc.ac.cn

Serial crystallography is a powerful technique in structure determination using many small crystals at X-ray free-electron laser or synchrotron radiation facilities. The large diffraction data volumes require high-throughput software to preprocess the raw images for subsequent analysis. *ClickX* is a program designated for serial crystallography data preprocessing, capable of rapid data sorting for online feedback and peak-finding refinement by parameter optimization. The graphical user interface (GUI) provides convenient access to various operations such as pattern visualization, statistics plotting and parameter tuning. A batch job module is implemented to facilitate large-data-volume processing. A two-step geometry calibration for single-panel detectors is also integrated into the GUI, where the beam center and detector tilting angles are optimized using an ellipse center shifting method first, then all six parameters, including the photon energy and detector distance, are refined together using a residual minimization method. Implemented in Python, *ClickX* has good portability and extensibility, so that it can be installed, configured and used on any computing platform that provides a Python interface or common data file format. *ClickX* has been tested in online analysis at the Pohang Accelerator Laboratory X-ray Free-Electron Laser, Korea, and the Linac Coherent Light Source, USA. It has also been applied in post-experimental data analysis. The source code is available via https://github.com/LiuLab-CSRC/ClickX under a GNU General Public License.

1. Introduction

In recent years, great success has been achieved in macromolecular structure determination by serial femtosecond crystallography (SFX) (Chapman *et al.*, 2011; Boutet *et al.*, 2012; Barends *et al.*, 2014). Using extremely bright X-ray free-electron laser (XFEL) pulses, a diffraction signal can be detected to atomic resolution with micrometre-sized crystals at room temperature under a 'diffract-before-destroy' scheme (Neutze *et al.*, 2000). This allows structure determination in ambient environments, thus significantly reducing structural alteration during the cryo-cooling process (Fraser *et al.*, 2011; Keedy *et al.*, 2014) adopted in diffraction experiments using synchrotron light sources. The femtosecond duration of XFEL pulses provides the unique advantage of probing fast dynamics in pump–probe experiments, particularly useful in understanding light-triggered processes (Kupitz *et al.*, 2014; Tenboer *et al.*, 2014; Pande *et al.*, 2016; Nogly *et al.*, 2018). If femtosecond temporal resolution is not desired, the idea of SFX can be extended to the serial crystallography (SX) at synchrotrons with microfocus crystallography beamlines (Nogly *et al.*, 2015).



In typical SX (including SFX) experiments, millions of raw images are collected to yield a complete data set that can be used for high-resolution structure determination (Liu & Spence, 2016). Furthermore, these data are often collected at high repetition rates, generating high throughput of raw data for the downstream analysis. For example, the Coherent X-ray Imaging instrument (CXI) (Liang *et al.*, 2015) at the Linac Coherent Light Source (LCLS), USA, allows diffraction data to be collected at a repetition rate of up to 120 Hz. However, only a small fraction of the raw data contains useful diffraction signals that can be processed to a merged 3D diffraction intensity map for final structure determination. Data reduction is the first step of the SX data analysis pipeline, where raw data frames with a number of diffraction peaks are identified as valid diffraction data (hits) for further analysis, such as indexing, scaling and merging. High-throughput programs with fast processing capability are required to provide real-time feedback to guide experiments. Furthermore, the indexing rate of SX diffraction data is sensitively dependent on the number of Bragg peaks. To reduce the inclusion of artifacts recognized as peaks (false positives) and exclusion of actual peaks (false negatives), several rounds of parameter fine tuning are necessary to yield optimal outcomes.

Currently there are several programs for SX data preprocessing. *CASS* (Foucar, 2016) is a modular program that can be used in both single-particle and SFX experiments. *Cheetah* (Barty *et al.*, 2014) provides rapid data reduction and a convenient graphical interface, *Cheetah-GUI*, for batch job management. Python-based *cctbx.xfel* (Sauter *et al.*, 2013) provides an alternative data analysis pipeline, including preprocessing (*e.g.* geometry optimization and spot finding) and post-processing (*e.g.* indexing and merging). *IOTA* (Lyubimov *et al.*, 2016) is a submodule of *cctbx.xfel* to optimize spot-finding parameters using a grid-search method. *OnDA* (Mariani *et al.*, 2016) is an online monitoring program that offers fast feedback on hit rates since it reads detector data directly from memory at LCLS. *NanoPeakCell* (Coquelle *et al.*, 2015) and *Psocake* (Thayer *et al.*, 2016; Damiani *et al.*, 2016) are Python-based graphical user interface (GUI) programs which provide data visualization and preprocessing support.

In this article, we introduce a software package for SX data analysis, with a focus on raw data preprocessing: *ClickX*. *ClickX* is implemented in Python, and therefore the program is easy to read and update and is highly portable for various platforms. *ClickX* is organized in modules, with the following major features:

(1) Cross platform, easy installation. As a Python program, *ClickX* can be downloaded and immediately run on any platform with Python environments.

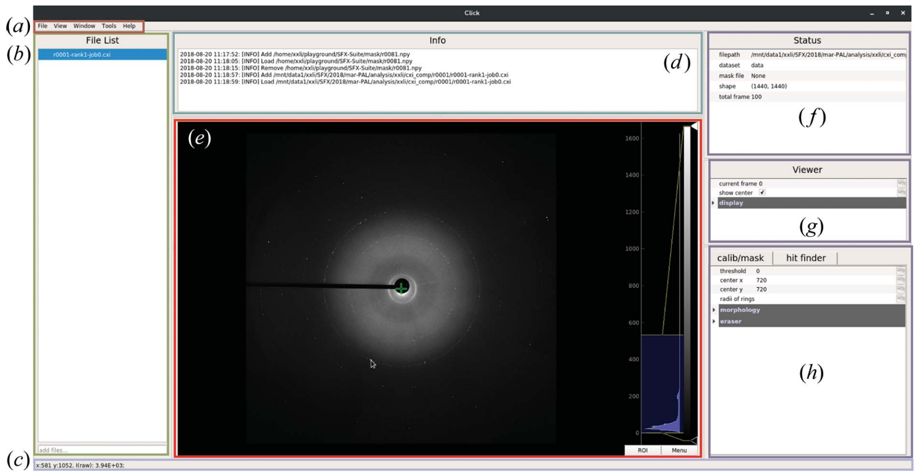
(2) Fast speed and high parallelization. *ClickX* extensively utilizes numerical packages including *NumPy* (https://www.numpy.org/) and *SciPy* (Jones *et al.*, 2001), so the processing speed is comparable to C/C++ software, such as *Cheetah*. Parallel computing is implemented with *mpi4py* (https://mpi4py.readthedocs.io/), gaining a linear speedup if input/output (I/O) is not a limiting factor.

(3) Real-time parameter tuning and feedback. *ClickX* provides two hit finders (SNR model and Poisson model, see Section 2.4 for details) and a GUI for real-time parameter tuning. Hit-finding results and statistics will be displayed immediately following the change of parameters.

(4) Highly automated data processing. A job management system is implemented for job submission and execution, and the program takes care of the data analysis in batches with a single click.

(5) Geometry calibration. *ClickX* integrates a calibration module to optimize geometry parameters such as the beam center, detector tilting angles, detector distance and photon energy.

(6) Data visualization. *ClickX* supports multiple data formats, and users can visualize a broad range of data types, including pixel values, individual diffraction patterns and statistical data from analyses.



2018-08-20 11:17:52: [INFO] Add /home/xxli/playground/SFX-Suite/mask/r0081.npy
2018-08-20 11:18:05: [INFO] Load /home/xxli/playground/SFX-Suite/mask/r0081.npy
2018-08-20 11:18:15: [INFO] Remove /home/xxli/playground/SFX-Suite/mask/r0081.npy
2018-08-20 11:18:57: [INFO] Add /mnt/data1/xxli/SFX/2018/mar-PAL/analysis/xxli/cxi_comp/r0001/r0001-rank1-job0.cxi
2018-08-20 11:18:59: [INFO] Load /mnt/data1/xxli/SFX/2018/mar-PAL/analysis/xxli/cxi_comp/r0001/r0001-rank1-job0.cxi

filepath	/mnt/data1/xxli/SFX/2018/mar-PAL/analysis/xxli/cxi_com
dataset	data
mask file	None
shape	(1440, 1440)
total frame	100

current frame	0
show center	✓
display	

threshold	0
center x	720
center y	720
radii of rings	
morphology	
eraser	

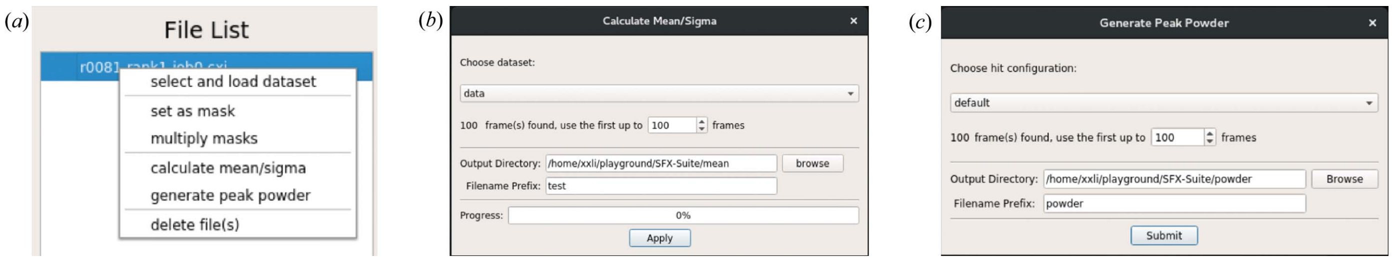
x:581 y:1052, I(raw): 3.94E+03

Figure 1
Layout overview of the main interface. (*a*) Menu bar. All other modules and widgets (*e.g.* batch module and geometry calibration module) can be accessed here. (*b*) File list panel. Data files can be added by dragging and dropping for further use. (*c*) Status bar. The pixel value and coordinates are displayed for inspection. (*d*) Info panel. Important events and messages are shown here. (*e*) Image viewer. The green '+' symbol shows the beam center. Users can zoom in/out or drag the displayed image for detailed checking. The intensity histogram is shown on the right-hand side, where the color bar can be altered via mouse dragging. (*f*) Status panel. Information on the current displayed file and mask file is shown here. (*g*) Viewer panel. Users can jump to a specified frame in the current data set. (*h*) Parameter panel. Users can create a mask and tune hit-finding parameters.

2. Functions and implementations

2.1. Data visualization

In data processing of SX experiments, visualization is helpful for users to get a comprehensive understanding of data, tune parameters and diagnose problems. *ClickX* provides a user-friendly interface (Fig. 1) to work with data stored in various formats, including



Choose dataset: data

100 frame(s) found, use the first up to 100 frames

Output Directory: /home/xxli/playground/SFX-Suite/mean

Filename Prefix: test

Progress: 0%

Choose hit configuration: default

100 frame(s) found, use the first up to 100 frames

Output Directory: /home/xxli/playground/SFX-Suite/powder

Filename Prefix: powder

Figure 2
Basic analysis. (*a*) Right-click file items in the file list panel. Users can perform some basic analyses. (*b*) Mean/sigma image generation dialog. (*c*) Peak powder generation dialog.

npy, npz, HDF5 and CXI (Maia, 2012). Most operations can be performed with a mouse (point–click–drag), including adding/removing files, selecting and loading data sets, zooming in/out on images, tuning the color bar, and inspecting pixel values.

2.2. Basic analysis of diffraction images

Some basic statistical analyses (Fig. 2) are implemented in *ClickX*, including the following:

(*a*) Generating the average intensity image and the associated variance image, *i.e.* mean/sigma value of each pixel. The average and standard deviation images can be generated from designated data sets, which can be used for mask preparation.

(*b*) Generating a powder diffraction image from hits. A powder diffraction pattern can be generated on the basis of the peaks identified with a user-specified hit-finding configuration. This can be used to calibrate experimental geometry using the calibration module to optimize the parameters associated with the detector.

(*c*) Composing masks. Users can combine multiple masks to generate a composed mask for more specific hit-finding tasks.

2.3. Mask building

During data collection of SX experiments, some pixels can be dysfunctional for various reasons, such as being shadowed by beam stops, intensity saturation, overheating *etc*. These pixels should be masked out for subsequent analysis. Fig. 3 shows mask-related widgets, where the mean/sigma images are used as primary references to build a mask. A threshold is applied to generate a binary mask, which can be further refined by morphology operations (dilation and erosion). *ClickX* also supports a flexible mask generation tool, *mask eraser*, which can be used to generate an arbitrary mask by dragging and erasing operations within the GUI. This feature facilitates the creation of masks composed of regions with arbitrary shapes.

2.4. Hit finding

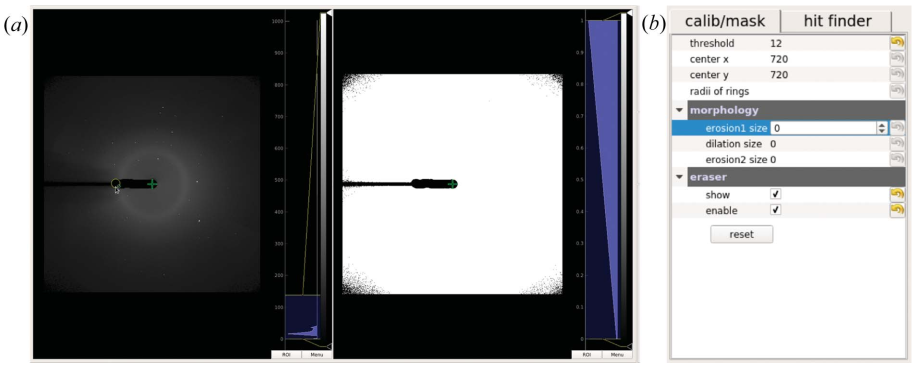
Hit finding is the most critical step in preprocessing of SX data, and it is also the core function in *ClickX*. We implemented two hit finders: the SNR (signal-to-noise ratio) model and the Poisson model (Lan *et al.*, 2018). The former approach evaluates peaks on the basis of SNR values, which is similar to the zaef method implemented in *CrystFEL* (White *et al.*, 2012), while the latter considers the statistical significance under the assumption of a Poisson distribution.

2.4.1. SNR model.

In the SNR model, the raw image is first smoothed by `gaussian_filter`, a Gaussian filter function implemented in *SciPy* (Jones *et al.*, 2001), to suppress background noise. The smoothed image is then used to calculate a gradient image for initial peak identification. Raw peaks in valid regions of gradient images are found by a local maximum searching method, which is implemented in the *scikit-image* (van der Walt *et al.*, 2014) package. All raw peaks are then evaluated on the basis of the SNR values. The number of signal pixels will be examined to screen spurious peaks resulting from sources other than crystals.

The SNR is estimated in a 7 × 7 cropped region centered on each peak candidate: SNR $= (\langle I_s \rangle - \langle I_b \rangle)/\sigma_b$, where $\langle I_s \rangle$ is the mean value of the signal region, and $\langle I_b \rangle$ and σ_b are the mean and standard derivation values of background region. The signal and background regions can be determined using three methods:

(1) Rings method. This method is similar to the ring integration method in



threshold 12
center x 720
center y 720
radii of rings
erosion1 size 0
dilation size 0
erosion2 size 0
show ✓
enable ✓

Figure 3
Mask building. (*a*) Raw image viewer (left) and calib/mask viewer (right). The yellow circle under the cursor is the mask eraser. Users can change the size and move around to create a custom mask. The generated mask is applied to the raw image viewer immediately and is also shown in the calib/mask viewer. (*b*) Calib/mask tab in the parameter panel.

CrystFEL. Each peak is described using three concentric circles, whose radius parameters are specified to define the signal and background regions. The smallest radius defines the signal region and other two radii define an annulus background region.

(2) Simple method. The signal and background regions are defined on the basis of the intensity values. By default, the 10 pixels (~20% of pixels in the cropped 7 × 7 image) with the highest intensities are selected as signal pixels, while the 35 pixels with the lowest intensities are treated as background pixels.

(3) Adaptive method. The lowest 70% of pixels (the percentage can be specified by the user) form the background region. The signal threshold is calculated from the background region: $I_t = \langle I_b \rangle + n\sigma_b$, where n is a user-specified parameter which is set to 5 by default.

2.4.2. Poisson model. The Poisson model assumes that the photon counts follow Poisson distributions. For a pixel with a background value b, the probability of obtaining a photon count K is $P_b(K) = \exp(-b)b^K/K!$. We can define a photon threshold K^* by the cumulative probability

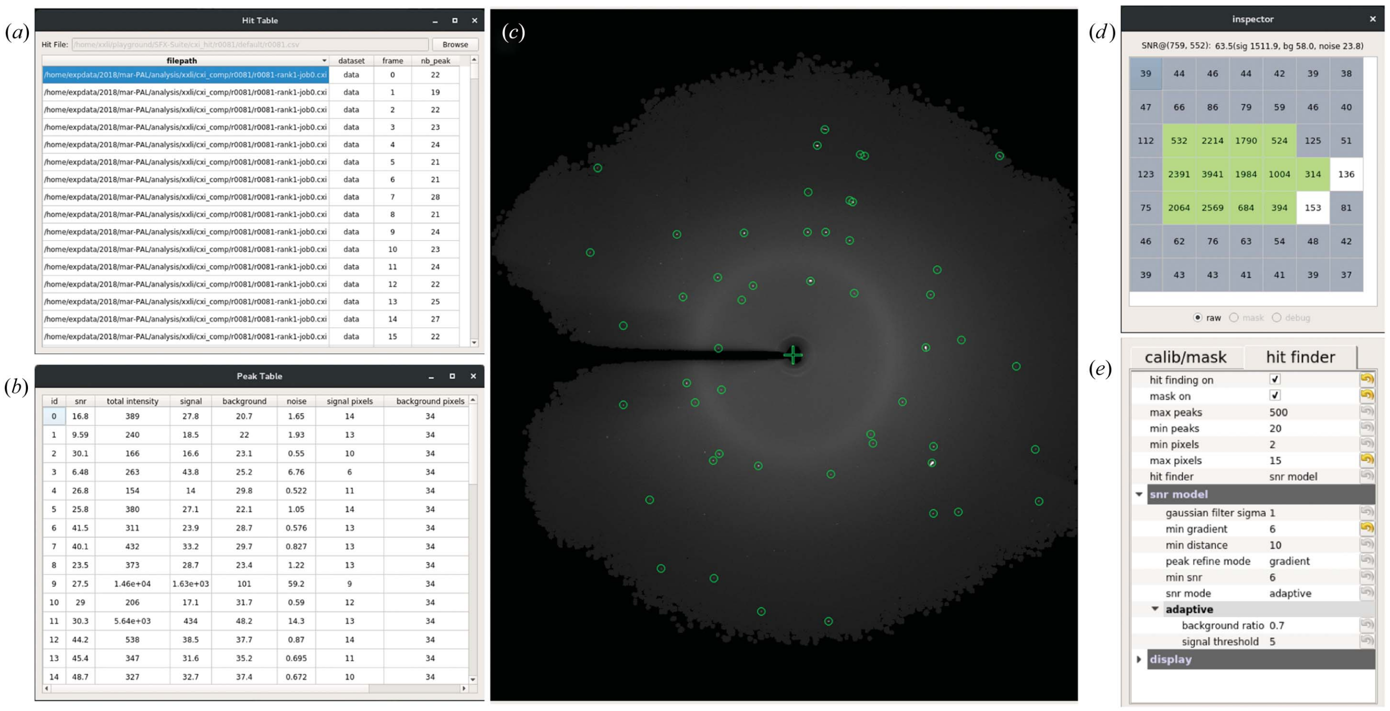
$$K^* = \operatorname{argmin} \sum_{K=0}^{K^*} P_b(K) > 1 - \epsilon, \qquad (1)$$

where ϵ is set to a very small number, *e.g.* 10^{-5}. A pixel is set as a signal pixel if its photon count is higher than the threshold K^*. Regions with connected signal pixels will be considered as peak candidates.

In the Poisson model, the raw image is first converted to a photon image by dividing ADU (analog–digital units) by the number of photons. A threshold image is then calculated on the basis of the Poisson distribution, where the background value is calculated by averaging photon counts in pixels located in the same ring. Initial peak candidates are identified from connected regions of the photon image with higher values than the threshold image. They are further filtered by the mask and the number of signal pixels, which can be adjusted by users.

In practice, the SNR-adaptive method is recommended in the first round of analysis. The detector layout (for multi-panel detectors such as the CSPAD at LCLS) and ADU per photon are required for hit finding using the Poisson model. The SNR-rings and SNR-simple methods have constraints on the sizes of the signal and background regions, determined by the user-specified parameters. The SNR-adaptive method only has one parameter controlling the size of the background region, and the signal region is determined by calculated the threshold from the background region and a user-specified parameter n. This will give more flexibility in signal pixel detection, which is advantageous in extracting information from peaks with large size variations.

2.4.3. Performance test. Speed tests of *ClickX* were carried out on a computer with a six-core Intel Xeon E5-2603 @



Hit File: /home/xxli/playground/SFX-Suite/cxi_hit/r0081/default/r0081.csv Browse

filepath	dataset	frame	nb_peak
/home/expdata/2018/mar-PAL/analysis/xxli/cxi_comp/r0081/r0081-rank1-job0.cxi	data	0	22
/home/expdata/2018/mar-PAL/analysis/xxli/cxi_comp/r0081/r0081-rank1-job0.cxi	data	1	19
/home/expdata/2018/mar-PAL/analysis/xxli/cxi_comp/r0081/r0081-rank1-job0.cxi	data	2	22
/home/expdata/2018/mar-PAL/analysis/xxli/cxi_comp/r0081/r0081-rank1-job0.cxi	data	3	23
/home/expdata/2018/mar-PAL/analysis/xxli/cxi_comp/r0081/r0081-rank1-job0.cxi	data	4	24
/home/expdata/2018/mar-PAL/analysis/xxli/cxi_comp/r0081/r0081-rank1-job0.cxi	data	5	21
/home/expdata/2018/mar-PAL/analysis/xxli/cxi_comp/r0081/r0081-rank1-job0.cxi	data	6	21
/home/expdata/2018/mar-PAL/analysis/xxli/cxi_comp/r0081/r0081-rank1-job0.cxi	data	7	28
/home/expdata/2018/mar-PAL/analysis/xxli/cxi_comp/r0081/r0081-rank1-job0.cxi	data	8	21
/home/expdata/2018/mar-PAL/analysis/xxli/cxi_comp/r0081/r0081-rank1-job0.cxi	data	9	24
/home/expdata/2018/mar-PAL/analysis/xxli/cxi_comp/r0081/r0081-rank1-job0.cxi	data	10	23
/home/expdata/2018/mar-PAL/analysis/xxli/cxi_comp/r0081/r0081-rank1-job0.cxi	data	11	24
/home/expdata/2018/mar-PAL/analysis/xxli/cxi_comp/r0081/r0081-rank1-job0.cxi	data	12	22
/home/expdata/2018/mar-PAL/analysis/xxli/cxi_comp/r0081/r0081-rank1-job0.cxi	data	13	25
/home/expdata/2018/mar-PAL/analysis/xxli/cxi_comp/r0081/r0081-rank1-job0.cxi	data	14	27
/home/expdata/2018/mar-PAL/analysis/xxli/cxi_comp/r0081/r0081-rank1-job0.cxi	data	15	22

id	snr	total intensity	signal	background	noise	signal pixels	background pixels
0	16.8	389	27.8	20.7	1.65	14	34
1	9.59	240	18.5	22	1.93	13	34
2	30.1	166	16.6	23.1	0.55	10	34
3	6.48	263	43.8	25.2	6.76	6	34
4	26.8	154	14	29.8	0.522	11	34
5	25.8	380	27.1	22.1	1.05	14	34
6	41.5	311	23.9	28.7	0.576	13	34
7	40.1	432	33.2	29.7	0.827	13	34
8	23.5	373	28.7	23.4	1.22	13	34
9	27.5	1.46e+04	1.63e+03	101	59.2	9	34
10	29	206	17.1	31.7	0.59	12	34
11	30.3	5.64e+03	434	48.2	14.3	13	34
12	44.2	538	38.5	37.7	0.87	14	34
13	45.4	347	31.6	35.2	0.695	11	34
14	48.7	327	32.7	37.4	0.672	10	34

SNR@(759, 552): 63.5(sig 1511.9, bg 58.0, noise 23.8)

39	44	46	44	42	39	38
47	66	86	79	59	46	40
112	532	2214	1790	524	125	51
123	2391	3941	1984	1004	314	136
75	2064	2569	684	394	153	81
46	62	76	63	54	48	42
39	43	43	41	41	39	37

hit finding on	✓
mask on	✓
max peaks	500
min peaks	20
min pixels	2
max pixels	15
hit finder	snr model
snr model	
gaussian filter sigma	1
min gradient	6
min distance	10
peak refine mode	gradient
min snr	6
snr mode	adaptive
adaptive	
background ratio	0.7
signal threshold	5
display	

Figure 4
Parameter-tuning widgets. (*a*) Hit table. Hit-finding job results such as file path, frame and number of found peaks are shown in this table. Users can sort the table by clicking the corresponding header such as `nb_peak`. By double-clicking any row, the user can jump to the corresponding frame in the main interface for further inspection. (*b*) Peak table. Information on identified peaks of the current image is shown in this sortable table. If the user double-clicks any row, the image viewer of the main interface will zoom in on the corresponding peak. (*c*) Image viewer with identified peaks shown in green circles. (*d*) Pixel inspector. A crop under the cursor of the current image is shown, including signal pixels (in green boxes) and background ones (in gray boxes) with intensity values. Statistics such as SNR, signal, background and noise are shown at the top of the panel. (*e*) Hit finder tab in the parameter panel. All changes of parameters will take effect immediately and found peaks will be plotted on the central image viewer.

1.7 GHz. A data set composed of 1000 frames, each with 1440×1440 pixels, was processed using the SNR-adaptive method three times on between one and five workers (one CPU core per worker). The result shows that the processing speed of *ClickX* scales linearly with the number of workers. On average, 3.5 frames per second per core can be achieved, so $\sim$35 (34 workers + 1 master) cores are required to process raw images at 120 Hz if I/O is not the limiting factor.

A comparison between *ClickX* and *Cheetah* has also been conducted. According to the analysis of the data sets recently collected at the Pohang Accelerator Laboratory X-ray Free-Electron Laser (PAL-XFEL), Korea, the hit-finding results are improved compared with those from *Cheetah*. However, we cannot conclude that *ClickX* outperforms *Cheetah*, since the hit-finding results depend on the parameter setup and characteristics of experimental samples. The main advantage is that *ClickX* gives user controls immediately accessible from the GUI and fast feedback so that it is easier to obtain optimal hit-finding parameters.

2.5. Parameter tuning

In *ClickX*, multiple widgets are implemented for parameter tuning (Fig. 4). Users can change parameters within the GUI in real time without the need to edit and specify a new configuration file or interrupt the ongoing process. Identified peaks are highlighted on the current image. Statistics such as SNR, total intensity and number of signal pixels are available in a peak table. This fast feedback can help users with hit-finding-parameter fine tuning. Once the hit-finding parameters are satisfactory, users can submit batch jobs to process/reprocess all data sets.

2.6. Geometry calibration

An accurate geometry is important for indexing. Because XFEL experiments often require fast readout, multi-panel detectors, such as the CSPAD, are used to match the data collection rate. The relative placements of each panel should be optimized to achieve accurate indexing results. Such functions have been reported to be available in *cctbx.xfel* (Hattne *et al.*, 2014), *geoptimiser* (Yefanov *et al.*, 2015) in *CrystFEL*, *DIALS* (Waterman *et al.*, 2016; Brewster *et al.*, 2018) and *cppxfel* (Ginn & Stuart, 2017). In *ClickX*, we focus on refining parameters associated with single-panel detectors (or multi-panel detectors with well determined metrology) using powder diffraction signals, including the photon energy (λ), detector distance (D), beam center (c_x, c_y) and detector tilting angles (θ_t, ϕ_t). The multi-panel detectors require more information than powder diffraction data to optimize the geometry for each panel independently. It is noted that hit finding is a local operation, so it does not rely on accurate metrology or geometry information.

Fig. 5 shows the geometry setup in *ClickX*. O_sXYZ is the Cartesian coordinate system of SX experiments, where O_s is the scattering point and Z is the beam direction. O_D is the beam center on the detector. D is the distance between the detector and the crystal sample. $\mathbf{n}$ is the normal vector of the detector, which is parallel with the Z direction if the detector is not tilted. $\mathbf{n}$ can be represented by two tilting angles: polar angle θ_t and azimuthal angle ϕ_t. $\mathbf{n}_p$ is the projected vector of axis Z on the detector plane. $\mathbf{x}_D/\mathbf{y}_D$ is the projected vector of the X/Y axis on the detector. $C^{(1)}, C^{(2)}, \ldots, C^{(m)}$ are the centers of powder rings.

The calibration process consists of two steps. The first step optimizes the beam center and detector tilting angles using a small-tilting approximation method. It utilizes the fact that the powder diffraction rays form a series of concentric conics which intersect with the detector, resulting in circular rings that share the same center if the incident beam is perpendicular to the detector ($\theta_t = 0$). If tilting exists, the rings become elliptical and the centers are not overlapped. The centers of these ellipses are located on the same line, which is parallel with $\mathbf{n}_p$.

With some approximations (see detailed derivation in Appendix *A*), we can prove that

$$\Delta l^{(i)} \simeq D \sin\theta_t \tan^2 2\theta^{(i)}, \tag{2}$$

where $\Delta l^{(i)}$ is the distance between the beam center and the center of the powder ring i at scattering angle $\theta^{(i)}$ ($2\theta^{(i)}$ is the angle between the incident beam and scattered beam directions of ring i).

The first step is implemented as follows:

(1) Clustering. In calibration experiments, usually inorganic crystals with small unit cells are used to collect powder patterns. This avoids the complexity of powder ring overlapping. Given N powder peaks from m powder rings, the radius of each peak is calculated, and then the DBSCAN method (Ester *et al.*, 1996) is used to group peaks into m clusters. Each ring forms a cluster, so peaks in this cluster share the same d spacing of the calibrant. Each cluster is evaluated to obtain its number of peaks $N^{(i)}$, center coordinate $c_x^{(i)}$, $c_y^{(i)}$, and scattering angle $\theta^{(i)}$.

(2) ϕ_t estimation. A linear regression is performed on the cluster centers, and $\mathbf{n}_p$ can be calculated from the regression results:

$$k_1, b_1 = \operatorname{linregress}([c_x^{(1)}, c_x^{(2)}, \ldots, c_x^{(m)}], [c_y^{(1)}, c_y^{(2)}, \ldots, c_y^{(m)}]), \tag{3}$$

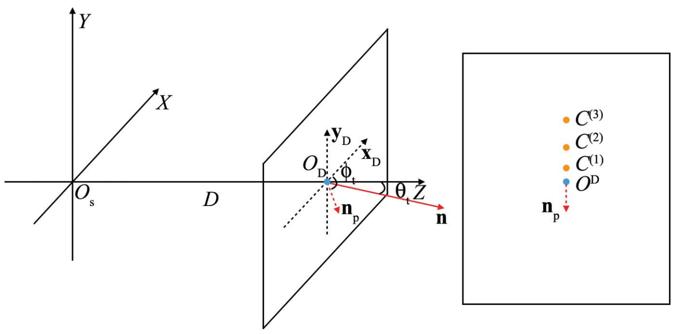
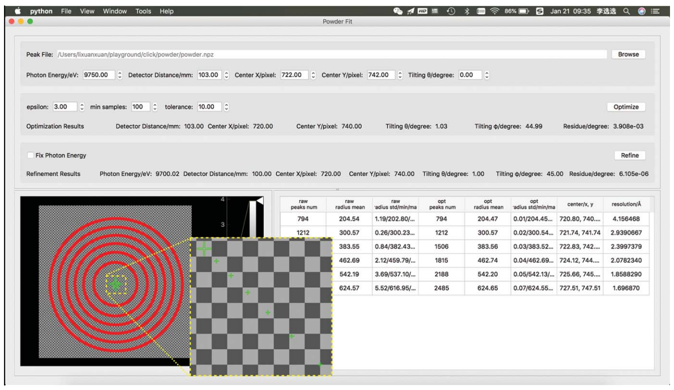


Figure 5
Geometry setup. The centers of powder rings $C^{(1)}, C^{(2)}, \ldots, C^{(m)}$ are used to optimize the beam center and detector tilting angles in the first step.



Peak File: /Users/lixuanxuan/playground/click/powder/powder.npz

Photon Energy/eV: 9750.00 Detector Distance/mm: 103.00 Center X/pixel: 722.00 Center Y/pixel: 742.00 Tilting θ/degree: 0.00

epsilon: 3.00 min samples: 100 tolerance: 10.00

Optimization Results Detector Distance/mm: 103.00 Center X/pixel: 720.00 Center Y/pixel: 740.00 Tilting θ/degree: 1.03 Tilting ϕ/degree: 44.99 Residue/degree: 3.908e-03

Fix Photon Energy

Refinement Results Photon Energy/eV: 9700.02 Detector Distance/mm: 100.00 Center X/pixel: 720.00 Center Y/pixel: 740.00 Tilting θ/degree: 1.00 Tilting ϕ/degree: 45.00 Residue/degree: 6.105e-06

raw peaks num	raw radius mean	raw 'adius std/min/ma	opt peaks num	opt radius mean	opt 'adius std/min/ma	center/x, y	resolution/Å
794	204.54	1.19/202.80/...	794	204.47	0.01/204.45...	720.80, 740....	4.156468
1212	300.57	0.26/300.23...	1212	300.57	0.02/300.54...	721.74, 741.74	2.9390667
	383.55	0.84/382.43...	1506	383.56	0.03/383.52...	722.83, 742....	2.3997379
	462.69	2.12/459.79/...	1815	462.74	0.04/462.69...	724.12, 744....	2.0782340
	542.19	3.69/537.10/...	2188	542.20	0.05/542.13/...	725.66, 745....	1.8588290
	624.57	5.52/616.95/...	2485	624.65	0.07/624.55...	727.51, 747.51	1.696870

Figure 6
Geometry calibration module. Peaks shown in red are clustered into multiple rings. The centers of rings are shown as small green '+' symbols, while the fitted beam center is shown as a large green '+' symbol. Detailed results of clustering and fitting are shown in the bottom-right table.

$$\mathbf{n}_{\rm p} = (1, k_1), \qquad (4)$$

where k_1 and b_1 represent the slope and intercept of this linear regression, respectively. The azimuthal angle $\phi_{\rm t}$ is the angle between $\mathbf{n}_{\rm p}$ and $\mathbf{x}_{\rm D}$.

(3) $\theta_{\rm t}$ estimation. For ring i, $\Delta l^{(i)} \simeq D \sin\theta_t \tan^2 2\theta^{(i)}$, and then the distance between the center of ring i and the first ring is $\Delta l_1^{(i)} = \Delta l^{(i)} - \Delta l^{(1)}$. A linear regression is performed on a series of $\Delta l_1^{(i)}$ and $\tan^2 2\theta^{(i)}$:

$$k_2, b_2 = \text{linregress}([\tan^2 2\theta^{(1)}, \tan^2 2\theta^{(2)}, \ldots, \tan^2 2\theta^{(m)}], \\ [\Delta l_1^{(1)}, \Delta l_1^{(2)}, \ldots, \Delta l_1^{(m)}]). \qquad (5)$$

The polar tilting angle $\theta_{\rm t}$ can be estimated using $k_2 = D \sin\theta_{\rm t}$.

(4) Beam-center estimation. In the previous step, we have $O_{\rm D}C^{(1)} = -b_2$, so the coordinate of the beam center $O_{\rm D}$ can be determined from $\mathbf{O}_{\rm D}\mathbf{C}^{(1)} = b_2\mathbf{n}_{\rm p}/|\mathbf{n}_{\rm p}|$.

(5) Optimization evaluation. To evaluate the goodness of fit after the optimization, the residual between the calculated 2θ and its model value of powder peaks can be calculated by

$$\text{Resi} = (1/N)\sum_{i,j} |2\theta^{ij}_{\rm obs} - 2\theta^{i}_{\rm model}|. \qquad (6)$$

$\theta^{ij}_{\rm obs}$ is the scattering angle of peak j on ring i based on current geometry parameters, while $\theta^{j}_{\rm model}$ is the model value of the scattering angle of ring i. In step (1), this model value is the mean value of all calculated scattering angles on this ring.

After the first step of optimization, a further refinement can be performed by minimizing the residual in the second step, which is similar to the approach taken in *FIT2D* (Hammersley *et al.*, 1996; Hammersley, 2016). The model value in this step is calculated from the photon energy and *d* spacings of the calibrant. The Nelder–Mead (Gao & Han, 2012) method is used to minimize the residual to obtain the final geometry parameters.

Fig. 6 shows a screenshot of a geometry calibration. For geometry calibration, a peak powder file is required as input. The powder image can be generated using the powder generator in the main interface of the *ClickX* program. Given appropriate initial geometry parameters, users can perform two-step calibration quickly.

2.7. Batch job system

In SX experiments, terabytes of data are collected and organized in tens, even hundreds, of data files to reduce the burden of I/O. To analyze such large data sets efficiently, a batch job system (Fig. 7) was implemented. Users can submit jobs and check the status and statistics of each job. Furthermore, an auto-processing mode can be enabled, so that jobs will be managed by the batch job system.

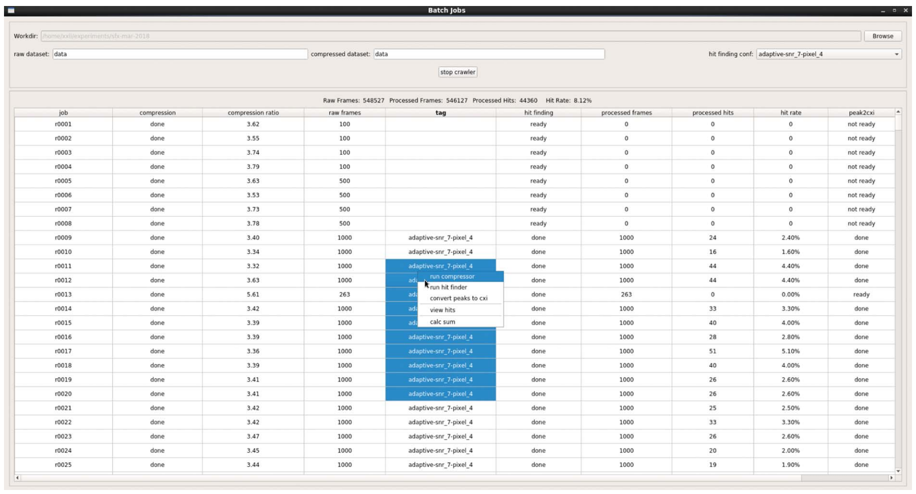
Currently *ClickX* supports three functions in batch mode:

(1) Data compression. Raw data can be compressed into CXI files with the HDF5 compression technique, which saves a significant amount of storage if users need to save or transport raw data sets.

(2) Hit finding. Users can execute hit finding with specified configurations obtained from the main interface.

(3) Peak2CXI conversion. If the hit-finding results are satisfactory, users can save image frames identified as hits and associated peak information in CXI files for further analysis; these files follow the format defined by the Coherent X-ray Imaging Data Bank (Maia, 2012).

Some useful statistics are summarized in the main panel of the batch job window, including job progress, number of raw frames, number of processed frames, number of hits, hit rate *etc*. After hit finding, users can visually inspect the identified hits in the main interface. On the basis of the outcomes, users can take appropriate actions, such as refining parameters for another round of hit-finding analysis.



Workdir: /home/xxli/experiments/sfx-mar-2018

raw dataset: data compressed dataset: data hit finding conf: adaptive-snr_7-pixel_4

Raw Frames: 548527 Processed Frames: 546127 Processed Hits: 44360 Hit Rate: 8.12%

job	compression	compression ratio	raw frames	tag	hit finding	processed frames	processed hits	hit rate	peak2cxi
r0001	done	3.62	100		ready	0	0	0	not ready
r0002	done	3.55	100		ready	0	0	0	not ready
r0003	done	3.74	100		ready	0	0	0	not ready
r0004	done	3.79	100		ready	0	0	0	not ready
r0005	done	3.63	500		ready	0	0	0	not ready
r0006	done	3.53	500		ready	0	0	0	not ready
r0007	done	3.73	500		ready	0	0	0	not ready
r0008	done	3.78	500		ready	0	0	0	not ready
r0009	done	3.40	1000	adaptive-snr_7-pixel_4	done	1000	24	2.40%	done
r0010	done	3.34	1000	adaptive-snr_7-pixel_4	done	1000	16	1.60%	done
r0011	done	3.32	1000	adaptive-snr_7-pixel_4	done	1000	44	4.40%	done
r0012	done	3.63	1000	ada	done	1000	44	4.40%	done
r0013	done	5.61	263	ada	done	263	0	0.00%	ready
r0014	done	3.42	1000	ada	done	1000	33	3.30%	done
r0015	done	3.39	1000	ada	done	1000	40	4.00%	done
r0016	done	3.39	1000	adaptive-snr_7-pixel_4	done	1000	28	2.80%	done
r0017	done	3.36	1000	adaptive-snr_7-pixel_4	done	1000	51	5.10%	done
r0018	done	3.39	1000	adaptive-snr_7-pixel_4	done	1000	40	4.00%	done
r0019	done	3.41	1000	adaptive-snr_7-pixel_4	done	1000	26	2.60%	done
r0020	done	3.41	1000	adaptive-snr_7-pixel_4	done	1000	26	2.60%	done
r0021	done	3.42	1000	adaptive-snr_7-pixel_4	done	1000	25	2.50%	done
r0022	done	3.42	1000	adaptive-snr_7-pixel_4	done	1000	33	3.30%	done
r0023	done	3.47	1000	adaptive-snr_7-pixel_4	done	1000	26	2.60%	done
r0024	done	3.45	1000	adaptive-snr_7-pixel_4	done	1000	20	2.00%	done
r0025	done	3.44	1000	adaptive-snr_7-pixel_4	done	1000	19	1.90%	done

Figure 7
Batch job system. Detailed results of jobs are shown in the job table. Users can right-click the table to submit corresponding jobs.

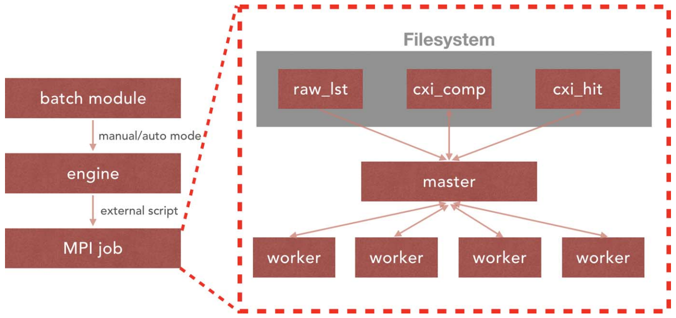


Figure 8
Parallel implementation. The engine accepts jobs from the batch module, and they are executed in a master/worker architecture. The master node collects job information from specified folders and dispatches jobs to workers dynamically. The results are sent to the master and saved in corresponding folders.

2.7.1. Parallel implementation. The batch job system is implemented in an efficient and extensible manner (Fig. 8). To ensure that the batch system can be used on any platform, an interface layer is implemented as the parallel engine. The available engines are for either local machines or PBS-configured clusters. The engines can be expanded to include other types of parallel systems, such as IBM's LSF or SLURM. Jobs can be submitted to the engine manually or automatically when the data are ready. The engine will run corresponding MPI jobs with external scripts.

The GUI of *ClickX* communicates with jobs via buffering files. All analysis projects using *ClickX* follow the same format for data and outputs. The raw data files to be processed are compiled to a list file and saved to the `raw_lst` folder. Compressed data are saved in the `cxi_comp` folder. Hit-finding results are saved in the `cxi_hit` folder.

All jobs are executed in the master/worker paradigm. When a new job is sent to the engine, the master will first collect job information from the corresponding folders or files, then split the multi-frame job into many single-frame job batches, and dispatch them to workers dynamically to balance the loads at the worker. The master is responsible for collecting analysis results from workers and saving them to corresponding folders.

3. Workflow with *ClickX*

As a modular program, *ClickX* is designed to provide users with full control of SX data preprocessing via three independent but also interconnected modules. Fig. 9 shows a basic workflow to process SX data with *ClickX*. In the main interface, preparation work such as mask building and hit-finding-parameter tuning can be done via mouse operations. In the batch job module, different types of jobs (compressing, hit finding, peak2cxi conversion) can be submitted to the engine in manual or auto modes. The results can be used to refine parameters in the main GUI. The peak powder data generated from the main interface can be used in the geometry calibration module to optimize experimental geometry. All these modules work together to support preprocessing of SX data.

4. Availability

The source code of *ClickX* is publicly available under a GPL license at GitHub (https://github.com/LiuLab-CSRC/ClickX). *ClickX* is compatible with both Python2 and Python3 environments. Dependencies include *NumPy*, *SciPy*, *mpi4py*, *scikit-image*, *pyqt5* (https://pypi.org/project/PyQt5/)and *pyqtgraph* (http://www.pyqtgraph.org/). A detailed introduction and usage instruction can be found at https://lixx11.github.io/ClickX.

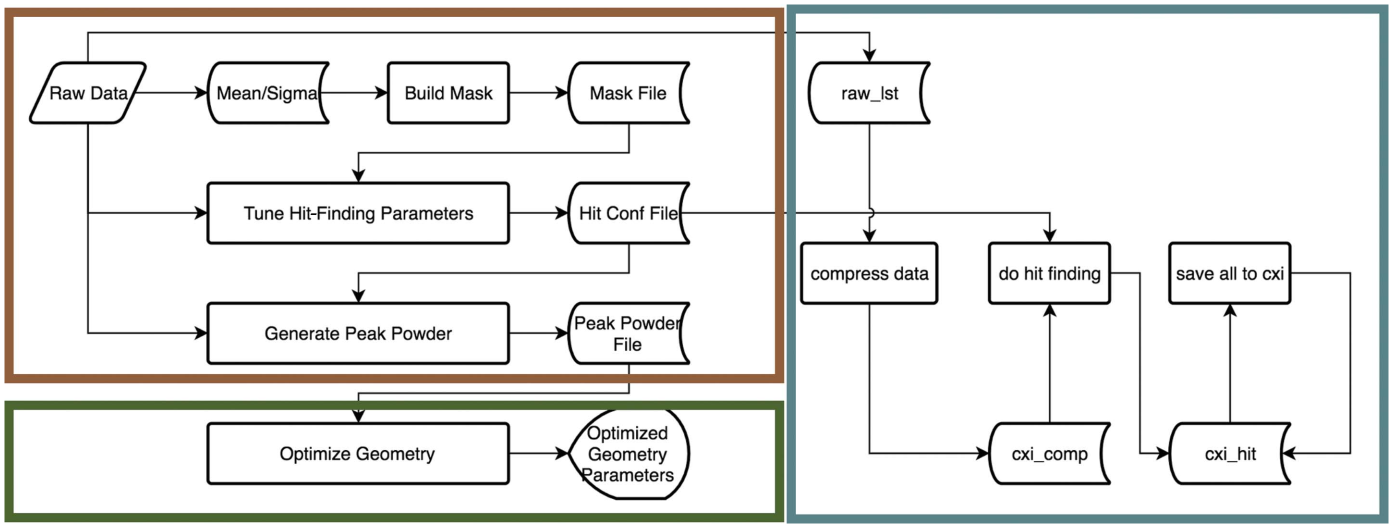


Figure 9
Workflow with *ClickX*. *ClickX* is organized in three modules: the main interface (top left), the geometry calibration module (bottom left) and the batch job system (right). Data compression is optional in the batch system. Hit finding can be performed directly on the raw data in `raw_lst` if compression is not required.

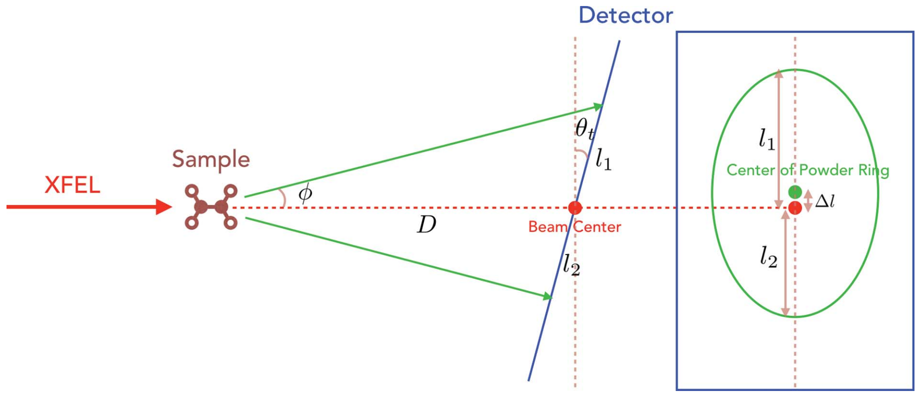


Figure 10
Schematic diagram of geometry calibration. The detector (blue) with small tilting angle is located downstream at a distance D from the powder sample. The resulting powder rings (green) exhibit an ellipse shape on the detector.

5. Conclusion

An SX preprocessing program, *ClickX*, has been developed with a user-friendly graphical interface for parameter tuning, batch job management and geometry calibration. *ClickX* is intended to support online/offline analysis for XFEL and synchrotron facilities that have the capability to conduct SX experiments. It has been thoroughly tested at PAL-XFEL and LCLS. *ClickX* is under active development and new features, such as support of time-resolved (Kupitz *et al.*, 2014; Tenboer *et al.*, 2014; Pande *et al.*, 2016; Nogly *et al.*, 2018) and two-color (Gorel *et al.*, 2017) serial crystallography experiments, will be integrated in future releases.

APPENDIX *A*
Geometry calibration

Fig. 10 shows a schematic diagram of the geometry calibration. Crystal samples are exposed to XFEL pulses, resulting in powder diffraction patterns with multiple rings. If the detector is tilted, the powder rings are not perfectly circular but elliptical with different centers.

For the powder ring at ϕ ($\phi = 2\theta$, where θ is the scattering angle) on a detector with tilting angle θ_t, we can calculate the distances between the beam center and two endpoints of the long axis of the elliptical ring:

$$l_1 = D\frac{\sin\phi}{\sin[90^\circ - (\phi + \theta_t)]} = D\frac{\sin\phi}{\cos(\phi+\theta_t)}, \tag{7}$$

$$l_2 = D\frac{\sin\phi}{\sin[90^\circ - (\phi - \theta_t)]} = D\frac{\sin\phi}{\cos(\phi-\theta_t)}. \tag{8}$$

Then the distance between the beam center and the center of this elliptical ring is

$$\begin{aligned}\Delta l &= \frac{l_1 - l_2}{2}\\ &= \frac{D\sin\phi}{2}\left[\frac{1}{\cos(\phi+\theta_t)} - \frac{1}{\cos(\phi-\theta_t)}\right]\\ &= \frac{D\sin\phi}{2}\,\frac{2\sin\phi\sin\theta_t}{\cos(\phi+\theta_t)\cos(\phi-\theta_t)}\\ &\simeq \frac{D\sin\theta_t\sin^2\phi}{\cos^2\phi} \quad (\text{when} \quad \theta_t \ll \phi)\\ &= D\sin\theta_t\tan^2\phi.\end{aligned} \tag{9}$$

Funding information

This work is partially supported by the National Natural Science Foundation of China (grant Nos. 11575021, U1530401 and U1430237 to Haiguang Liu). Funding from the Chinese Academy of Engineering Physics is acknowledged. This work was also supported by the US National Science Foundation BIO ABI grant No. 1565180.

References

Barends, T. R., Foucar, L., Botha, S., Doak, R. B., Shoeman, R. L., Nass, K., Koglin, J. E., Williams, G. J., Boutet, S., Messerschmidt, M. & Schlichting, I. (2014). *Nature*, **505**, 244–247.
Barty, A., Kirian, R. A., Maia, F. R. N. C., Hantke, M., Yoon, C. H., White, T. A. & Chapman, H. (2014). *J. Appl. Cryst.* **47**, 1118–1131.
Boutet, S., Lomb, L., Williams, G. J., Barends, T. R., Aquila, A., Doak, R. B., Weierstall, U., DePonte, D. P., Steinbrener, J., Shoeman, R. L., Messerschmidt, M., Barty, A., White, T. A., Kassemeyer, S., Kirian, R. A., Seibert, M. M., Montanez, P. A., Kenney, C., Herbst, R., Hart, P., Pines, J., Haller, G., Gruner, S. M., Philipp, H. T., Tate, M. W., Hromalik, M., Koerner, L. J., Bakel, N. van, Morse, J., Ghonsalves, W., Arnlund, D., Bogan, M. J., Caleman, C., Fromme, R., Hampton, C. Y., Hunter, M. S., Johansson, L. C., Katona, G., Kupitz, C., Liang, M., Martin, A. V., Nass, K., Redecke, L., Stellato, F., Timneanu, N., Wang, D., Zatsepin, N. A., Schafer, D., Defever, J., Neutze, R., Fromme, P., Spence, J. C. H., Chapman, H. N. & Schlichting, I. (2012). *Science*, **337**, 362–364.
Brewster, A. S., Waterman, D. G., Parkhurst, J. M., Gildea, R. J., Young, I. D., O'Riordan, L. J., Yano, J., Winter, G., Evans, G. & Sauter, N. K. (2018). *Acta Cryst.* D**74**, 877–894.

Chapman, H. N., Fromme, P., Barty, A., White, T. A., Kirian, R. A., Aquila, A., Hunter, M. S., Schulz, J., DePonte, D. P., Weierstall, U., Doak, R. B., Maia, F. R. N. C., Martin, A. V., Schlichting, I., Lomb, L., Coppola, N., Shoeman, R. L., Epp, S. W., Hartmann, R., Rolles, D., Rudenko, A., Foucar, L., Kimmel, N., Weidenspointner, G., Holl, P., Liang, M., Barthelmess, M., Caleman, C., Boutet, S., Bogan, M. J., Krzywinski, J., Bostedt, C., Bajt, S., Gumprecht, L., Rudek, B., Erk, B., Schmidt, C., Hömke, A., Reich, C., Pietschner, D., Strüder, L., Hauser, G., Gorke, H., Ullrich, J., Herrmann, S., Schaller, G., Schopper, F., Soltau, H., Kühnel, K., Messerschmidt, M., Bozek, J. D., Hau-Riege, S. P., Frank, M., Hampton, C. Y., Sierra, R. G., Starodub, D., Williams, G. J., Hajdu, J., Timneanu, N., Seibert, M. M., Andreasson, J., Rocker, A., Jönsson, O., Svenda, M., Stern, S., Nass, K., Andritschke, R., Schröter, C., Krasniqi, F., Bott, M., Schmidt, K. E., Wang, X., Grotjohann, I., Holton, J. M., Barends, T. R. M., Neutze, R., Marchesini, S., Fromme, R., Schorb, S., Rupp, D., Adolph, M., Gorkhover, T., Andersson, I., Hirsemann, H., Potdevin, G., Graafsma, H., Nilsson, B. & Spence, J. C. H. (2011). *Nature*, **470**, 73–77.

Coquelle, N., Brewster, A. S., Kapp, U., Shilova, A., Weinhausen, B., Burghammer, M. & Colletier, J.-P. (2015). *Acta Cryst.* D**71**, 1184–1196.

Damiani, D., Dubrovin, M., Gaponenko, I., Kroeger, W., Lane, T. J., Mitra, A., O'Grady, C. P., Salnikov, A., Sanchez-Gonzalez, A., Schneider, D. & Yoon, C. H. (2016). *J. Appl. Cryst.* **49**, 672–679.

Ester, M., Kriegel, H.-P., Sander, J. & Xu, X. (1996). *KDD-96 Proceedings*, pp. 226–231. AAAI Press.

Foucar, L. (2016). *J. Appl. Cryst.* **49**, 1336–1346.

Fraser, J. S., van den Bedem, H., Samelson, A. J., Lang, P. T., Holton, J. M., Echols, N. & Alber, T. (2011). *Proc. Natl Acad. Sci. USA*, **108**, 16247–16252.

Gao, F. & Han, L. (2012). *Comput. Optim. Appl.* **51**, 259–277.

Ginn, H. M. & Stuart, D. I. (2017). *J. Synchrotron Rad.* **24**, 1152–1162.

Gorel, A., Motomura, K., Fukuzawa, H., Doak, R. B., Grünbein, M. L., Hilpert, M., Inoue, I., Kloos, M., Kovácsová, G., Nango, E., Nass, K., Roome, C. M., Shoeman, R. L., Tanaka, R., Tono, K., Joti, Y., Yabashi, M., Iwata, S., Foucar, L., Ueda, K., Barends, T. R. M. & Schlichting, I. (2017). *Nat. Commun.* **8**, 1170.

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

Hammersley, A., Svensson, S., Hanfland, M., Fitch, A. & Hausermann, D. (1996). *Int. J. High. Press. Res.* **14**, 235–248.

Hattne, J., Echols, N., Tran, R., Kern, J., Gildea, R. J., Brewster, A. S., Alonso-Mori, R., Glöckner, C., Hellmich, J., Laksmono, H., Sierra, R. G., Lassalle-Kaiser, B., Lampe, A., Han, G., Gul, S., DiFiore, D., Milathianaki, D., Fry, A. R., Miahnahri, A., White, W. E., Schafer, D. W., Seibert, M. M., Koglin, J. E., Sokaras, D., Weng, T. C., Sellberg, J., Latimer, M. J., Glatzel, P., Zwart, P. H., Grosse-Kunstleve, R. W., Bogan, M. J., Messerschmidt, M., Williams, G. J., Boutet, S., Messinger, J., Zouni, A., Yano, J., Bergmann, U., Yachandra, V. K., Adams, P. D. & Sauter, N. K. (2014). *Nat. Methods*, **11**, 545–548.

Jones, E., Oliphant, T., Peterson, P., *et al.* (2001). *SciPy: Open Source Scientific Tools for Python*, http://www.scipy.org/.

Keedy, D. A., van den Bedem, H., Sivak, D. A., Petsko, G. A., Ringe, D., Wilson, M. A. & Fraser, J. S. (2014). *Structure*, **22**, 899–910.

Kupitz, C., Basu, S., Grotjohann, I., Fromme, R., Zatsepin, N. A., Rendek, K. N., Hunter, M. S., Shoeman, R. L., White, T. A., Wang, D., James, D., Yang, J. H., Cobb, D. E., Reeder, B., Sierra, R. G., Liu, H., Barty, A., Aquila, A. L., Deponte, D., Kirian, R. A., Bari, S., Bergkamp, J. J., Beyerlein, K. R., Bogan, M. J., Caleman, C., Chao, T. C., Conrad, C. E., Davis, K. M., Fleckenstein, H., Galli, L., Hau-Riege, S. P., Kassemeyer, S., Laksmono, H., Liang, M., Lomb, L., Marchesini, S., Martin, A. V., Messerschmidt, M., Milathianaki, D., Nass, K., Ros, A., Roy-Chowdhury, S., Schmidt, K., Seibert, M., Steinbrener, J., Stellato, F., Yan, L., Yoon, C., Moore, T. A., Moore, A. L., Pushkar, Y., Williams, G. J., Boutet, S., Doak, R. B., Weierstall, U., Frank, M., Chapman, H. N., Spence, J. C. & Fromme, P. (2014). *Nature*, **513**, 261–265.

Lan, T.-Y., Wierman, J. L., Tate, M. W., Philipp, H. T., Martin-Garcia, J. M., Zhu, L., Kissick, D., Fromme, P., Fischetti, R. F., Liu, W., Elser, V. & Gruner, S. M. (2018). *IUCrJ*, **5**, 548–558.

Liang, M., Williams, G. J., Messerschmidt, M., Seibert, M. M., Montanez, P. A., Hayes, M., Milathianaki, D., Aquila, A., Hunter, M. S., Koglin, J. E., Schafer, D. W., Guillet, S., Busse, A., Bergan, R., Olson, W., Fox, K., Stewart, N., Curtis, R., Miahnahri, A. A. & Boutet, S. (2015). *J. Synchrotron Rad.* **22**, 514–519.

Liu, H. & Spence, J. C. (2016). *Quant. Biol.* **4**, 159–176.

Lyubimov, A. Y., Uervirojnangkoorn, M., Zeldin, O. B., Brewster, A. S., Murray, T. D., Sauter, N. K., Berger, J. M., Weis, W. I. & Brunger, A. T. (2016). *J. Appl. Cryst.* **49**, 1057–1064.

Maia, F. R. (2012). *Nat. Methods*, **9**, 854–855.

Mariani, V., Morgan, A., Yoon, C. H., Lane, T. J., White, T. A., O'Grady, C., Kuhn, M., Aplin, S., Koglin, J., Barty, A. & Chapman, H. N. (2016). *J. Appl. Cryst.* **49**, 1073–1080.

Neutze, R., Wouts, R., van der Spoel, D., Weckert, E. & Hajdu, J. (2000). *Nature*, **406**, 752–757.

Nogly, P., James, D., Wang, D., White, T. A., Zatsepin, N., Shilova, A., Nelson, G., Liu, H., Johansson, L., Heymann, M., Jaeger, K., Metz, M., Wickstrand, C., Wu, W., Båth, P., Berntsen, P., Oberthuer, D., Panneels, V., Cherezov, V., Chapman, H., Schertler, G., Neutze, R., Spence, J., Moraes, I., Burghammer, M., Standfuss, J. & Weierstall, U. (2015). *IUCrJ*, **2**, 168–176.

Nogly, P., Weinert, T., James, D., Carbajo, S., Ozerov, D., Furrer, A., Gashi, D., Borin, V., Skopintsev, P., Jaeger, K., Nass, K., Båth, P., Bosman, R., Koglin, J., Seaberg, M., Lane, T., Kekilli, D., Brünle, S., Tanaka, T., Wu, W., Milne, C., White, T., Barty, A., Weierstall, U., Panneels, V., Nango, E., Iwata, S., Hunter, M., Schapiro, I., Schertler, G., Neutze, R. & Standfuss, J. (2018). *Science*, **361**, eaat0094.

Pande, K., Hutchison, C. D., Groenhof, G., Aquila, A., Robinson, J. S., Tenboer, J., Basu, S., Boutet, S., DePonte, D. P., Liang, M., White, T. A., Zatsepin, N. A., Yefanov, O., Morozov, D., Oberthuer, D., Gati, C., Subramanian, G., James, D., Zhao, Y., Koralek, J., Brayshaw, J., Kupitz, C., Conrad, C., Roy-Chowdhury, S., Coe, J. D., Metz, M., Xavier, P. L., Grant, T. D., Koglin, J. E., Ketawala, G., Fromme, R., rajer, V., Henning, R., Spence, J. C. H., Ourmazd, A., Schwander, P., Weierstall, U., Frank, M., Fromme, P., Barty, A., Chapman, H. N., Moffat, K., van Thor, J. J. & Schmidt, M. (2016). *Science*, **352**, 725–729.

Sauter, N. K., Hattne, J., Grosse-Kunstleve, R. W. & Echols, N. (2013). *Acta Cryst.* D**69**, 1274–1282.

Tenboer, J., Basu, S., Zatsepin, N., Pande, K., Milathianaki, D., Frank, M., Hunter, M., Boutet, S., Williams, G. J., Koglin, J. E., Oberthuer, D., Heymann, M., Kupitz, C., Conrad, C., Coe, J., Roy-Chowdhury, S., Weierstall, U., James, D., Wang, D., Grant, T., Barty, A., Yefanov, O., Scales, J., Gati, C., Seuring, C., Srajer, V., Henning, R., Schwander, P., Fromme, R., Ourmazd, A., Moffat, K., Van Thor, J. J., Spence, J. C. H., Fromme, P., Chapman, H. N. & Schmidt, M. (2014). *Science*, **346**, 1242–1246.

Thayer, J., Damiani, D., Ford, C., Gaponenko, I., Kroeger, W., O'Grady, C., Pines, J., Tookey, T., Weaver, M. & Perazzo, A. (2016). *J. Appl. Cryst.* **49**, 1363–1369.

Walt, S. van der, Schonberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., Gouillart, E., Yu, T. & the scikit-image contributors (2014). *PeerJ*, **2**, e453.

Waterman, D. G., Winter, G., Gildea, R. J., Parkhurst, J. M., Brewster, A. S., Sauter, N. K. & Evans, G. (2016). *Acta Cryst.* D**72**, 558–575.

White, T. A., Kirian, R. A., Martin, A. V., Aquila, A., Nass, K., Barty, A. & Chapman, H. N. (2012). *J. Appl. Cryst.* **45**, 335–341.

Yefanov, O., Mariani, V., Gati, C., White, T. A., Chapman, H. N. & Barty, A. (2015). *Opt. Express*, **23**, 28459–28470.