The Fast Azimuthal Integration Python library

Giannis Ashiotis, Jérôme Kieffer
ESRF, 71 avenue des Martyrs, 38000 Grenoble, France

PyFAI is an open-source software package, written in Python, for performing azimuthal integration and 2D-regrouping on area detector frames for SAXS, WAXS and XRPD experiments. This is done by histogramming pixel positions weighted by the pixel intensity. Pixels spanning over more than one histogram bin contribute to the neighbouring bins accordingly. This results in greatly reduced noise in the diffraction pattern. To achieve the necessary execution speeds required by such experiments, the concept of a look-up table was introduced. Furthermore the code is parallelized using OpenCL, which achieves very good performances even on very cost-effective graphics cards.

Introducton

Azimuthal integration allows the use of area detectors for recording powder diffraction patterns

Data reduction step is very time-consuming → bottleneck

- 2D raster scans with pencil beam
- diffraction tomography

Recent developments in PyFAI address this issue by achieving the required execution speeds, without sacrificing any accuracy

Experiment Description

PyFAI offers 40 detector definitions (Pilatus, Maxipix...)

Support for optically coupled CCD detectors w/ geometric distortion correction

Custom definitions following NeXus convention

- sample-detector distance
- PONI
- 3D rotation of detector

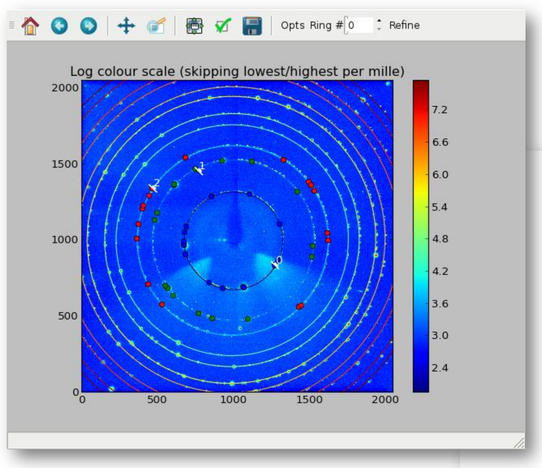
Binning of detectors → larger pixel size

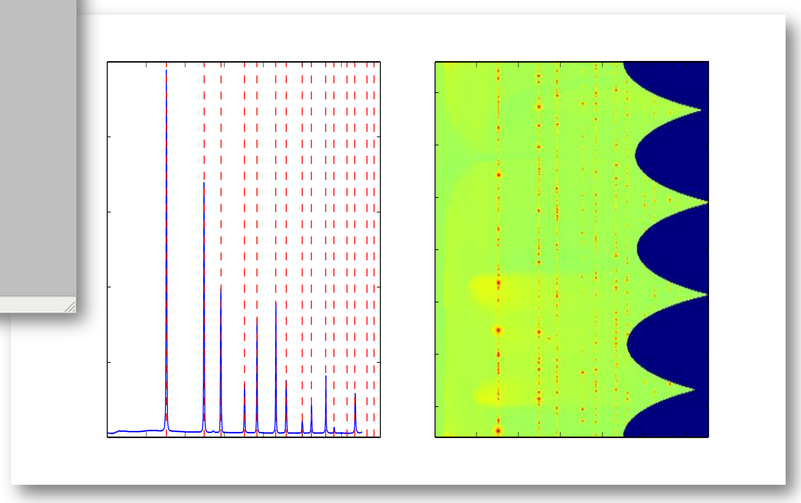
Calibration

Selection of 10 calibrants

- Powder diffraction (ceria, corundum, gold, silicon, …)
- Small angle scattering (silver behenate, Tetradecanol, …)

Pick-picking done using Massif extraction and Blob detection algorithms





Integration

User has three options when it comes to pixel-splitting:

- No splitting
- Bounding box splitting
- Tight pixel splitting

At this point, for correct error calculation avoid using pixel-splitting

Parallel implementation using OpenCL

Azimuthal integration is a scatter operation

Look Up Table (LUT) → sparse matrix – dense vector product

Easier to parallellise with OpenCL / PyOpenCL

Now code can run on :

- CPUs
- GPUs
- Accelerators

Speeds of up to 700 Mpixel/s

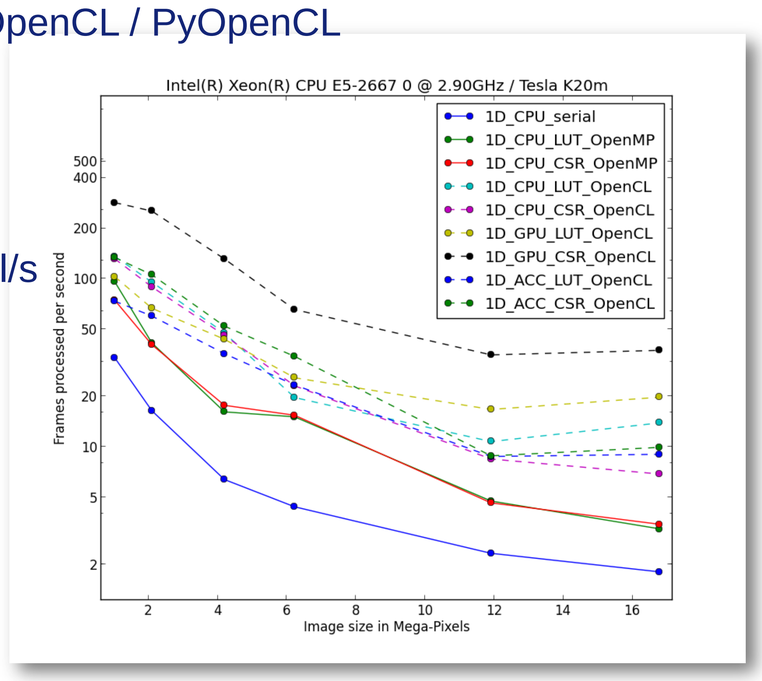


Image ?

Signal separation

PyFAI, through the *AzimuthalIntegrator* or the *Geometry* class, allows diffraction image generation for testing purposes

AzimuthalIntegrator.calcfrom1d

A script is also available to regenerate the diffraction image using the integrated powder pattern

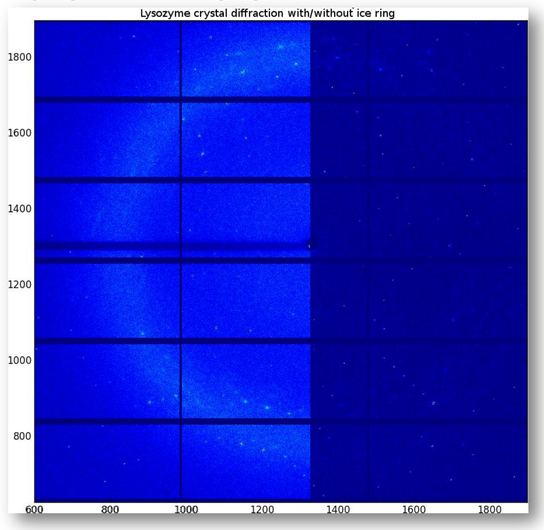
pyFAI-calib → validate

Another method of the *AzimuthalIntegrator* class is *separate*, which can separate the amorphous scattering from the Bragg peaks

AzimuthalIntegrator.separate

Applications in: High-pressure diffraction experiments
Serial crystallography (see image)

Image/code ?



Conclusions

PyFAI succeeds in addressing the challenges that arise from such experiments as SAXS, WAXS and XRPD. It even goes further by offering features that find applications in other type of experiments like serial crystallography. And with processing speed reaching 700 Mpixel/s, it is able to handle the huge data streams coming from modern detectors

Acknowledgements
Claudio Ferrero – head of the Data Analysis Unit
Andy Götz – head of the Software Group
LinkSCEEM-2 – fanatical support
Dimitris Karkoulis, Zubair Nawaz, Aurore Deschildre, Frédéric Picca – contributors

References