[MS10-P18] DXTBX: the diffraction experiment toolbox

James Parkhurst[1], Aaron Brewster[2], Luis Fuentes-Montero[1], David Waterman[3], Johan Hattne[2], Nathaniel Echols[2], Gwyndaf Evans[1], Nicholas Sauter[2], Graeme Winter[1]

[1] *Diamond Light Source,*
[2] *Lawrence Berkeley National Laboratory,*
[3] *STFC Rutherford Appleton Laboratory*
E-mail: james.parkhurst@diamond.ac.uk

Despite a push towards the adoption of industry standards, the recording of X-ray diffraction data remains fragmented. Processing data from single crystal X-ray diffraction experiments, therefore, requires the ability to read, and interpret, image and metadata from a variety of data formats employing different experimental representations. Tools have previously been developed to address this problem. The CCP4 Diffraction Image library [1] was developed to support the DNA and xia2 projects; however, it was limited by a lack of extensibility. FabIO [2] provides generic access to 2D image data but does not give consistent access to experimental models derived from image metadata.

The dxtbx provides a consistent interface to both image data and experimental models whilst supporting a completely generic, user-extensible approach to reading the data and metadata from different sources. Both single image files, such as imgCIF [3], and multiple-image files, such as HDF5, are supported. It was originally developed for use in xia2 [4] to replace the Diffraction Image library; however, it has since been incorporated into the cctbx [5]. The library is written in a mixture of C++ and Python, is open source and is distributed under a BSD license.

Access to the experimental models and image data is provided through a high-level ‘sweep’ interface instantiated by a factory function from the input data representation. A sweep represents a series of images that have a well defined geometric relationship between adjacent pixels in 3D, e.g. a series of images taken using the rotation method. The sweep provides convenient access to image data through a python list-style interface and provides methods to extract arbitrary 3D volumes from the images sequence. Simple access to experimental geometry is provided through four container classes: beam, detector, goniometer and scan. The detector model, being necessarily the most complex, also provides, for example, methods to predict intersections by diffracted beam vectors; the concept of a virtual detector plane [6] is used in order to provide abstraction from the device dependent underlying implementation. Complex effects such as parallax correction and arbitrary pixel alignments are handled transparently through a configurable pixel to millimetre mapping function. In this way, general algorithms can be developed without reference to the specifics of the experimental setup and hardware used.

[1] Remacle, F. and Winter, G., (2007), CCP4 newsletter on protein crystallography.
[2] Knudsen, K., Sørensen, H., Wright, J., Goret, G. and Kieffer, J., (2013), *J. Appl. Cryst.*, **46** (2), 537–539.
[3] Bernstein, H. and Hammersley, A., (2005), International Tables for Crystallography, **G**, 37–43.
[4] Winter, G., (2009), *J. Appl. Cryst.*, **43** (1), 186–190.
[5] Grosse-Kunstleve, R., Sauter, N., Moriarty, N. and Adams, P., (2002), *J. Appl. Cryst.*, **35** (1), 126–136.
[6] Bricogne, G., (1987), Proceedings of the CCP4 Daresbury Study Weekend, 120–145.

Keywords: software; experimental geometry; diffraction image data