Area Detector Interface

Request for Guidance

Sent to expert users by cpo on April 17, 2024:

As you perhaps know, most of our effort at this point is devoted to psana2 (for LCLS2) and the associated LCLS2 DAQ. psana1 (for LCLS1) is more-or-less frozen except for critical work. In my biased opinion we've done a better job with the LCLS2 daq/analysis, learning from the mistakes we made previously. One of the more complex issues we deal with is the area-detector interface (multi-panel, gain-ranging). The purpose of this message is to solicit some input on what features would be useful to you as users for the complex area detectors.

At the moment the Detector interface for these is simple, in part because this interface is exposed to the AMI2 realtime graphical analysis package:

- det.raw.raw(), det.raw.calib(), det.raw.image() which return raw-data, calibrated unassembled data, and calibrated assembled data respectively (similar to psana1)
- in cases where feature-extracted data is generated by the realtime data-reduction-pipeline it will show up as something like det.fex.peaks()
- det.calibconst returns a python dictionary of calibration constants, which can be complex for multi-panel gain ranging detectors. some of these may need helper functions to translate into user-friendly information like x/y/z coordinates of pixels
- python kwargs to control the calibration behavior can be provided to the Detector constructor (e.g. common-mode parameters). kwargs of unassembled arrays can also be provided to det.raw.image() to assemble, for example, a bad pixel mask into a 2D image.

My question: could you summarize what information you would like easy access to? I'm thinking of things like the get_psf() function that Mikhail implemented for Rick in psana1 that returns fast-index/slow-index geometry vectors, or arrays of pixel x/y/z coordinates.

For information that is of common interest we would like to try to create interfaces that you would find useful. Even though we would try to keep the main Detector interface simple (to keep AMI2 simple) we can create other helper objects that return the information you need. If email turns out to be too cumbersome for this discussion, we are happy to schedule a zoom brainstorming session for whoever might be interested/available. But we thought we would try email first, which may be more efficient use of people's time.

Thanks for any suggestions you might have for this,

Reply from Anton Barty

Perhaps it's already there, but I would also like to see a bad pixel map; or more precisely 'which pixels in this image can I trust and which can I not'. For complex detectors I accept this may be different on each frame depending on gain settings / memory cells / signal close to edge case of calibration, etc. Flattening this array often results in ridiculously many bad pixels, while frame-by-frame many more pixels are in fact useable.

It is definitely useful to be able to reverse index from a point in the assembled image to the point in the unassembled data (eg: person clicks on image, where is that pixel in the unassembled data?). This could be vectorised for efficiency...

I like to have access to a pixel map version of the geometry: 'where is this memory element in real space'. Despite it being very basic and sometimes limited, it's still super useful and covers many cases easily. One could add 'what is the solid angle of this pixel' - but I know that gets complicated due to for example 3D charge cloud created in the detector and spillover from one pixel to the next. Still, basic 'x,y,z and solid angle' arrays for the unassembled data are useful. A radius-from-bbeam-centre is useful for azimuthal averages. Sure one can create all these from more complex geometry descriptions, but why not one person do it properly once….

Come to think of it, 'polar plot of the data' could be a useful function (remap 2D data into radius-angle in 2D) - but depends on knowledge of the beam centre and detector tilt - which may be lacking - so would have to be used with extreme care.

IAnton adds these thoughts)

Is that bad pixel masks are not necessarily a constant. They can change from frame to frame.

For example with the AGIPD (admittedly an EuXFEL problem) we found that the bad pixel mask depended on memory cell and gain setting. And then possibly if signal was in one of the regimes near gain switching where calibration was unreliable.
In other words, bad pixel mask changed per shot - first due to which memory cell, then how much signal was on the pixel.
This makes it not a 'det.calibconst'.
My suggestion was to have space for a 'bad pixels in this event' rather than a constant mask applied for all events.

Although sometimes there may of course be a constant masks, for example for truly dead pixels.

Reply from Aaron Brewster

Hi Chris, in addition to Anton's comments which I echo (namely that the mask is dynamic per-image depending on gain switching), we also use the psana1 interfaces to read LCLS geometry files:

- PSCalib.SegGeometryStore.sgs
- Detector.pyda.geoaccess

(example usage in DIALS)

We use these interfaces to construct the set of vectors that describe a multi-panel detector, using the values recorded in the calib directory in the geometry subfolder. This gives us an initial model that we refine and store in the DIALS format.