1.2.5 Azimuthal Averaging

Another option is to reduce the data by azimuthally averaging the signal. Here, we need to know the center for the integration as well as the detector distance and beam energy. The latter two are mostly important for the q-values of the bins that will also be stored in the smallData. As conditions will change during an experiment, it is convenient to have a function that return the correct integration setup for each (range of) run(s). In addition to a 1-d result as a q-vector, you can also define a number of phi-slices that should be used for integration, resulting in a 2-d average in q-phi space.
The averaging will automatically apply a corrections for the x-ray polarization and geometric acceptance. What is currently not corrected for are signal differences due to different path length of the photons when crossing the detector . Also, the calculations assume that the detector is mounted perpendicular to the beam direction.

```
def getAzIntParams(run):
    """ Parameters for azimuthal integration
    See azimuthalBinning.py for more info
    """
    if isinstance(run,str):
        run=int(run)
    ret_dict = {}
    if run>0:
        az_dict = {'eBeam': 6.5} # keV
        #az_dict['center'] = [224.367146, 870.269879] # um
        az_dict['center'] = [-549.943775, 261.418876] # um
        az_dict['dis_to_sam'] = 80. # mm
        az_dict['tx'] = 0 # deg
        az_dict['ty'] = 0 # deg
        az_dict['phiBins'] = 11 # number of phi bins
        ret_dict['epix10k2M'] = az_dict
    return ret_dict
```

Use of SmallDataAna_psana to set up the producer's parameters

See the following link to learn how to start an interactive iPython session and make an average image for a given detector:

1.2 Area Detector treatment with DetObject#1.2AreaDetectortreatmentwithDetObject-InteractiveSmallDataAnasession

Beam center and radius

```
SDAna In: anaps.FitCircle()
```

This will by draw the image and let you either select points by clicking or define a threshold above which points are selected and fitted to a single circle. The azimuthal integration code needs a center point which the circle fitting will return.

Below is an example of the entire prompt:

- First the image you created first will pop up. You can then either select points by hand or use a threshold (highest x% of pixels).
- The chosen pixel location will be shown and you are adjust your threshold until you are satisfied.
- In the last step, the fit is performed and overlaid on the image. The beam center and radius of the circle are printed. You will have to know about your sample to use the radius to extract the detector - sample distance. The latter will not affect she shape of the azimuthally integrated data (a wrong center will!), but the q-bin values will be wrong.

```
SDana In [2]: anaps.FitCircle()
plot AvImg_cspad using the 5/99.5 percentiles as plot min/max: (0.199616, 263.383)
Select Circle Points by Mouse?:
n
Select Circle Points with threshold (y/n):
y
min percentile % of selected points:
99.2
thresP 248.786973755
Happy with this threshold (y/n):
y
x,y:  99652.2541599 87977.9162216  R  18603.6250138
```

FitCircle has an optional argument (useMask=False/True) that defaults to False. If set to True, the mask stored in the calib directory will be applied.

Run the Azimuthal integration

This function will set up the azimuthal integration. It will also correct for the LCLS Xray polarization and the geometric acceptance. The center and distance to sample are needed. Then this code can be applied to any average image by:

```
SDana In [2]: anaps.AzInt?
Type:        instancemethod
String form: <bound method SmallDataAna_psana.AzInt of <SmallDataAna_psana.SmallDataAna_psana object at
0x7f7731f13f10>>
File:        /reg/data/ana13/xpp/xppo6616/results/littleData/xppmodules/scripts/SmallDataAna_psana.py
Definition:  anaps.AzInt(self, detname=None, use_mask=False)
```

It returns the data arrays. The q-bins are stored as well in anaps.<detname>.azav_q

This function is meant to test what your parameter will do to the average image you have created to check that they are reasonable before adding them to the SmallDataProducer file where the same code will be run on the data for each event and stored in the smallData hdf5 file. You can find an Example: Setting up the azimuthal integration of a powder pattern using anaps