3.1 Post-processing functions

Common

The additional analysis function are passed to the cube workflow by adding a `det_proc` keys to the detector definition dictionary. The content of this key must be another dictionary with the following sctructure:

```
{
        name: '<function_name>' # must be the exact DetObjectFunc name,
        'func_kwargs1': <...>,
         'func_kwargs2': <...>,
        ...
}
```

The function `<function_name>` will then be applied to the summed binned detector data and its output will be saved under `<detector name>_variable` in the binned hdf5 file, in addition to the full detector data.

Azimuthal integration

pyFAI

Detector definition section with azimuthal integration of the Rayonix:

```
# ##### DETECTORS ##############################################################################
# List detectors to be cubed. Area detector have additional options such as threshold
# Full images will always be saved.
# More area detector can be defined following the same syntax, and adding them to varList
detDict = {'source':'Rayonix',
           'full':1,
           'image':1,
           'thresADU':-1e5,
           'common_mode':0}

pix_size = 176e-6
func_kwargs = {
    'name': 'azav_pyfai', # must be the name of the smalldata_tools DetObjectFunc
    'ai_kwargs': {'dist':1, 'poni1':960*pix_size, 'poni2':960*pix_size},
    'npts': 512,
    'int_units' : '2th_deg',
    'return2d' : False
}
det_proc = [func_kwargs]
detDict['det_proc'] = det_proc

# make list of all variables to be added to the cube
varList = ['ipm_dg1/sum','ipm_dg2/sum', detDict]
```

The h5 resulting from this detector definition then contains:

- `Rayonix_azav (Array)`
- `Rayonix_data (Array)`
- `Rayonix_nEntries (Array)`
- `Rayonix_q (Array)`

The available keyword arguments to pass to the azav_pyfai function are

azav_pyfai documentation

```
        """
    Parameters
    ----------
    name: str
        Function name

    mask: array, optional
        User defined mask. 1 for valid pixels.

    return2d: bool, optional
        Return a 2d (q,phi). Default: False

    poni_file: str, Path object, optional
        Path to a pyFAI calibration file

    ai_kwargs: dict, optional
        Arguments to pyFAI.AzimuthalIntegrator. Either this parameter or a calib file is necessary
        For arguments see: https://pyfai.readthedocs.io/en/master/api/pyFAI.html#module-pyFAI.
azimuthalIntegrator

    pol_factor: float, optional
        Polarization factor. Default 1. Passed to integrate1d or integrate2d.

    npts_radial: int, optional
        Number of points for the radial binning. Default 256.

    npts_az: int, optional
        Number of points for the azimuthal binning. Default 360. Only used for the 2d integration.

    azav_kwargs: dict, optional
        Additonal arguments to pass to integrate1d or integrate2d.
        See https://pyfai.readthedocs.io/en/master/api/pyFAI.html#pyFAI.azimuthalIntegrator.AzimuthalIntegrator.
integrate1d
    """
```

Fourier transform

TBD

Autocorrelation

TBD