CSPAD2x2 modules in Python (deprecated)

Content

In 2014 new universal detector geometry software is implemented and documented in the Detector Geometry page. New interface completely supersedes features discussed in this page and is recommended for use.

This note describes alternative old-style python modules which access calibrated geometry parameters for CSPAD2x2.

Calibration types

Current total list of calibration types for CSPAD2x2 is shown in the table:

Type	Shape	Description
center	(3, 2)	x, y, z coordinates of the two 2x1 centers
tilt	(2,)	tilt angle of the two 2x1s assuming rotation angle = 180 degrees
beam_vector	(3,)	The same as for CSPAD
filter	(3,)	The same as for CSPAD
common_mode	(3,)	The same as for CSPAD
pedestals	(185, 388, 2)	The same as for CSPAD
pixel_status	(185, 388, 2)	The same as for CSPAD

Two of them, **center** and **tilt**, are used to describe precise pixel geometry in the detector. The **beam_vector** is intended (but not used yet and need to be elaborated) to describe the detector position w.r.t. IP. Other types, **filter**, **common_mode**, **pedestals**, and **pixel_status**, have the same meaning as for CSPAD detector and are used for intensity correction, that is beyond the scope of this note.

Code location

All modules discussed in this note belong to the package `PyCSPadImage` of **psana** release, beginning from ana-0.9.8. This package contain modules for CSPAD and CSPAD2x2 detectors. Module with name pattern `CSPAD2x2*.py` are intended for CSPAD2x2.
Their functionality is explained in the table:

Class name	Functionality
CSPAD2x2CalibPars.py	Provides access to the calibration parameters through the DB
CSPAD2x2CalibParsDefault.py	Sets default values for all calibration types
CSPAD2x2Examples.py	Example of how to get calibrated data, get image, and plot it
CSPAD2x2PixCoords.py	Provides access to the pixel coordinate arrays
PixCoords2x1.py	Provides access to the pixel coordinate arrays for 2x1 sensor

GlobalGraphics.py	A set of simple graphical methods based on `matplotlib`
HDF5Methods.py	A set of methods for interaction with hdf5 files
GlobalMethods.py	A set of useful global methods

Program interface

Import

The modules discussed in this note can be imported as:

```
import PyCSPadImage.CSPAD2x2PixCoords as pixcoor
import PyCSPadImage.CSPAD2x2CalibPars as calpars
```

In example we also import a couple of useful modules:

```
import PyCSPadImage.HDF5Methods       as hm
import PyCSPadImage.GlobalGraphics    as gg
```

Instantiate calibration object

The object of the class `CSPAD2x2CalibPars` may be instantiated by a few different ways, as explained below.

Regular constructor

For regular instantiation of the calibration object, the class `CSPAD2x2CalibPars` constructor expects at least two (named) parameters,

- `path` - the string path to the directory with calibration types, and
- `run` - integer run number.
  Third parameter `list_of_clib_types`is used for run-time optimization; if it is missing, all parameters will be loaded, that takes more time and memory.

```
    # Official path:
    path = '/reg/d/psdm/mec/mec73313/calib/CsPad2x2::CalibV1/MecTargetChamber.0:Cspad2x2.1/
    # or local:
    # path = '/reg/neh/home1/dubrovin/LCLS/CSPad2x2Alignment/calib-cspad2x2-01-2013-02-13/'
list_of_clib_types = ['center', 'tilt', 'pedestals']
    calib = calpars.CSPAD2x2CalibPars(path, run, list_of_clib_types)
```

Default constructor

Calibration object can be instantiated without parameters;

```
    calib = calpars.CSPAD2x2CalibPars()
```

Default values of all parameters will be used in this case, which allows to run code without any crash, but does not provide correct geometry. Necessary parameters may be provided later by the methods:

```
    calib.setCalibParsForPath (run, path)
```

or

```
    run      = 123,
    calibdir = '/reg/d/psdm/mec/mec73313/calib',
    group    = 'CsPad2x2::CalibV1',
    source   = 'MecTargetChamber.0:Cspad2x2.1'

    calib.setCalibPars (run, calibdir, group, source) :
```

Update run number

If the path to calibration types is already instantiated, the run number can be changed in the process of analysis by the call like

```
    run=234 # actual run number should be provided...
    calib.setRun (run)
```

Get pixel coordinates

In order to get the CSPAD2x2 geometry calibrated parameters, the `coord` object of the class `CSPAD2x2PixCoords` needs to be instatiated, using calibration parameters loaded in the `calib` object. Then the X and Y pixel coordinate (numpy) arrays of the data-like shape (185, 388, 2) can be obtained through the method `get_cspad2x2_pix_coordinate_arrays_pix()`,

```
    coord = pixcoor.CSPAD2x2PixCoords(calib)
    X,Y = coord.get_cspad2x2_pix_coordinate_arrays_pix()
```

Get array of calibration parameters

Method `calib.getCalibPars(type, run)` allows to get array of calibration parameters for specific type and run number, for example

```
    peds_arr = calib.getCalibPars('pedestals',123)
```

returns the pedestals (numpy) array with data-like shape (185, 388, 2).

Conversion between (185, 388, 2) and (2, 185, 388) shapes

Module `CSPAD2x2CalibPars.py` contains a couple of global methods which convert the array shape forth and back between (185, 388, 2) and (2, 185, 388);

```
arrTwo2x1 = calpars.data2x2ToTwo2x1(arr2x2)    # converts (185, 388, 2) to (2, 185, 388)
arr2x2    = calpars.two2x1ToData2x2(arrTwo2x1) # converts (2, 185, 388) to (185, 388, 2)
```

Get CSPAD2x2 data array from hdf5 file

```
import HDF5Methods as hm
    fname  = '/reg/neh/home1/dubrovin/LCLS/HDF5Analysis-v01/PyCSPadImage/src/mec73313-r0180.h5
    dsname = '/Configure:0000/Run:0000/CalibCycle:0000/CsPad2x2::ElementV1/MecTargetChamber.0:Cspad2x2.1/data'
    data_arr = hm.getDataSetForOneEvent(fname, dsname, event=0)
```

Image from pixel coordinate and intensity arrays

Depending on experimental requirements, there may be different methods of how to generate image. For example we use a simple 2-d histogram technique.
The CSPAD2x2 image can be obtained from the pixel coordinate and intensity arrays (of the same shape) by call:
img2d = gg.getImageFromIndexArrays(X,Y,arr)

Plot image and its spectrum

```
import GlobalGraphics as gg
    my_range = None
    my_range = (-10,40)
    gg.plotImageLarge(img2d, amp_range=my_range)
    gg.plotSpectrum(img2d, amp_range=my_range)
    gg.show()
```

Code example

Essential code of the module `CSPAD2x2Examples.py`:

```
import sys
import numpy as np

import PyCSPadImage.CSPAD2x2PixCoords as pixcoor
import PyCSPadImage.CSPAD2x2CalibPars as calpars

import PyCSPadImage.HDF5Methods       as hm
import PyCSPadImage.GlobalGraphics    as gg
#------------------------------

def test_cspad2x2_calib_geometry() :
    """Test method, demonstrates how to work with CSPAD2x2CalibPars and CSPAD2x2PixCoords modules
    """
    #======= Define input parameters
    Ndet = 5
    run  = 180
    path = '/reg/d/psdm/mec/mec73313/calib/CsPad2x2::CalibV1/MecTargetChamber.0:Cspad2x2.%1d/' % Ndet
    #path = '/reg/neh/home1/dubrovin/LCLS/CSPad2x2Alignment/calib-cspad2x2-0%1d-2013-02-13/' % Ndet
    #fname  = '/reg/d/psdm/mec/mec73313/hdf5/mec73313-r%04d.h5' % run
    fname  = '/reg/neh/home1/dubrovin/LCLS/HDF5Analysis-v01/PyCSPadImage/src/mec73313-r%04d.h5' % run
    dsname = '/Configure:0000/Run:0000/CalibCycle:0000/CsPad2x2::ElementV1/MecTargetChamber.0:Cspad2x2.%1d
/data' % Ndet
    list_of_clib_types = ['center', 'tilt', 'pedestals']

    #======= Get calibration object
    calib = calpars.CSPAD2x2CalibPars(path, run, list_of_clib_types)

    #======= Get CSPAD2x2 pixel coordinate arrays, shaped as (2, 185, 388)
    coord = pixcoor.CSPAD2x2PixCoords(calib)
    X,Y = coord.get_cspad2x2_pix_coordinate_arrays_pix()

    #======= Get CSPAD2x2 pedestals array, shaped as (185, 388, 2)
    peds_arr = calib.getCalibPars('pedestals')

    #======= Get data array from hdf5 dataset, shaped as (185, 388, 2)
    data_arr = hm.getDataSetForOneEvent(fname, dsname, event=0) - peds_arr

    #======= Convert shape from (185, 388, 2) to (2, 185, 388)
    ord_arr  = calpars.data2x2ToTwo2x1(data_arr)

    #======= Compose and plot CSPAD2x2 image from coordinate and intensity arrays
    img2d = gg.getImageFromIndexArrays(X,Y,ord_arr)

    #======= Print for test purpose
    calib.printCalibParsStatus()
    #print 'pedestals:\n', calib.getCalibPars('pedestals')
    print 'center:\n',    calib.getCalibPars('center')
    print 'tilt:\n',      calib.getCalibPars('tilt')
    print 'peds_arr.shape:', peds_arr.shape  # = (185, 388, 2)
    print 'Get data array from file: ' + fname
    print 'data_arr.shape:', data_arr.shape
    print 'ord_arr.shape:', ord_arr.shape
    print 'img2d.shape:', img2d.shape

    #======= Plot image and spectrum
    my_range = (-10,40) # None
    gg.plotImageLarge(img2d, amp_range=my_range)
    gg.plotSpectrum(img2d, amp_range=my_range)
    gg.show()

#------------------------------
 if __name__ == "__main__" :
    test_cspad2x2_calib_geometry()
    sys.exit ( 'End of test.' )
#------------------------------
```

This example generates a couple of plots for CSPAD2x2 image and spectrum:

References

- Detector Geometry - new approach to detector geometry description
- CSPAD2x2 Alignment
- CSPAD Geometry and Alignment