Note on XCS code from Marcin

Content

Get data from xtc files

Scripts for pre-processing in pyana

Pyana configuration file and module:

- `/reg/neh/home1/sikorski/xcs_pyana_current/pyana.cfg`
- `/reg/neh/home1/sikorski/xcs_pyana_current/xcs_timepix_pkg/src/import_data.py`

Main features of `import_data.py`:
splits image for 10x10 parts shaped as (130,134), and saves them in sub-directory like
`/reg/neh/home1/sikorski/xcs_pyana_current/e167-r0015-s00-c00/2013-04-03-10-39-22-734268/`

Content of `pyana.cfg`

on 2013-10-14

```
[pyana]
modules = xcs_timepix_pkg.import_data
[xcs_timepix_pkg.import_data]
detector = 1        <=== 1 = PI, 2 = TimePix
dark_fina =  /reg/neh/home1/sikorski/xcs_pyana_current/e167-r0020-s00-c00/2013-04-03-09-43-06-939033/e167-r0020-
s00-c00_dark_img.txt
flag_bin  = 1
bin_x = 1
bin_y = 1
LLD = 20.0
s_x = 10
s_y = 10
saxs_interval = 100
not_zero = 1.0E-9
r1 = 0
c1 = 0
r2 = 1300
c2 = 1340
saturation = 65535
```

`import_data.py`

Essential code

```
 def __init__ ( self, ...)
     self.frame_size = ... (1300,1340)
     self.step_x = ... 130
     self.step_y = ... 134
     self.LLD     = ... 20.
     self.not_zero = 1.0E-9
     self.saturation = 65535
     self.dark - array with pedestals

     self.avg_img = np.zeros(self.frame_size)
     self.saturated_pixels = np.ones((1300,1340))

 def beginjob( self, evt, env ) :
      ...
     for i in range(0,self.frame_size[0],self.step_x):
         for j in range(0,self.frame_size[1],self.step_y):
             fina = path + '/' +fina_gen + '_data_' + str(i) + '_' +str(j) + '.npy'
             self.f_id.append(open(fina, 'w'))

 def event( self, evt, env ) :
     ...
              PI = evt.get(xtc.TypeId.Type.Id_PrincetonFrame)
              frame = np.array(PI.data()[self.R1:self.R2,self.C1:self.C2], dtype = np.float64)
              a = np.where(frame >= self.saturation)
              self.saturated_pixels[a] = 0
              np.subtract(frame, self.dark, frame)
              frame[frame < self.LLD] = self.not_zero

     np.add(self.avg_img, frame , self.avg_img)


     counter = 0
     for i in range(0,self.frame_size[0],self.step_x):
         for j in range(0,self.frame_size[1],self.step_y):
            np.save(self.f_id[counter], frame[i:i+self.step_x,j:j+self.step_y])
            counter += 1


def endjob( self, evt, env ) :
    ...
    counter = 0
     self.avg_img = self.avg_img/self.calibcycle_counter
     for i in range(0,self.frame_size[0],self.step_x):
         for j in range(0,self.frame_size[1],self.step_y):
             self.f_id[counter].close()
             counter += 1

     np.savetxt(self.f_id_sat, self.saturated_pixels,fmt='%-15f')
     np.savetxt(self.f_id_avg,self.avg_img, fmt='%-15f')
```

Sequence of operations on pixels

1. Get data frame in np.float64: `frame = np.array(PI.data()[self.R1:self.R2,self.C1:self.C2], dtype = np.float64)`
2. Set saturated pixels with intensity >=65535 to 0 `a = np.where(frame >= self.saturation); self.saturated_pixels[a] = 0`
3. Subtract pedestals: `np.subtract(frame, self.dark, frame)`
4. Apply LLD: `frame[frame < self.LLD] = self.not_zero`
   ---
5. Accumulate frame in avg_img: `np.add(self.avg_img, frame , self.avg_img)`
6. Accumulate split frame in files:
   `np.save(self.f_id[counter], frame[i:i+self.step_x,j:j+self.step_y]) #, fmt='%-20.10f')`

Evaluate correlators

Scripts

/reg/neh/home1/sikorski/scripts/run_correlator.py
/reg/neh/home1/sikorski/scripts/correlator/xcs_correlator.py

Use split image files from

- in 2013-04-03 test: /reg/neh/home1/sikorski/xcs_pyana_current/e167-r0015-s00-c00/2013-04-03-10-39-22-734268/
- in 2013-08-23 test: /reg/neh/home1/sikorski/xcs_pyana_current/e167-r0011-s00-c00/2013-08-23-11-30-43-581071/

Content of `run_correlator.py`

on 2013-10-14

```
import sys
sys.path.append("/reg/neh/home/sikorski/scripts/correlator")
import xcs_correlator as xcs_corr
######################
path = r"/reg/neh/home1/sikorski/xcs_pyana_current/e167-r0011-s00-c00/2013-08-23-11-30-43-581071/
path_mask = r"/reg/neh/home/sikorski/scripts"
fina_mask = r"test.mat.npz"
ts_fina = r"e167-r0011-s00-c00_time_stamps.txt"
monitor_number = -1
treshold_low = -0.002
treshold_high = 0.1E9
LLD = 20
tau = [1,2,3,4,5,6,7,8,9,10,12,14,16,20]
######################             | Define parameters in xcs_correlator.py
c = xcs_corr.img_correlator(path)  | self.path = path; self.notzero = 1; self.results = {};
c.set_time_stamps_fina(ts_fina)    | self.time_stamps_fina = "e167-r0011-s00-c00_time_stamps.txt"
c.initialize_correlator()          | defines many self.*
c.set_monitor(monitor_number)      | self.monitor_name = 'no normaloization'; self.monitor = np.ones_like
(monitor[:,0]) = all 1
c.set_time_stamps_fina(ts_fina)    | !!! repetition...
c.set_LLD(LLD)                     | self.LLD = 20
c.set_tau(tau)                     | self.tau = [1,2,3,4,5,6,7,8,9,10,12,14,16,20]
c.set_mask_folder(path_mask)       | self.mask_folder = "/reg/neh/home/sikorski/scripts"
c.set_mask_file(fina_mask)         | self.mask_fina = "test.mat.npz"
c.set_mask()                       | loads a few files in dictionary self.mask = {}
c.set_treshold(treshold_low)       | self.treshold =  -0.002 (def=0)
c.set_treshold_up(treshold_high)   | self.treshold_up =0.1E9 (def=1E4)
c.submit_correlator_jobs()         | calculates correlators
c.avg_g2()
c.harvest_SAXS_data()
c.plot_g2()
c.plot_fit_results(777)
c.plot_SAXS()
plt.show()
```

Essential code of `xcs_correlator.py`

on 2013-10-14

```
class img_correlator:
    def __init__ (self,path):
        self.path = path
        self.notzero = 1
        self.results = {}
        #self.LLD = 20 # it was 200 in the past


    def submit_correlator_jobs(self):
        filenames = os.listdir(self.path)
        for filename in filenames:

                  self.correlator_2D(filename)
```

```
    def correlator_2D(self, fina):

        f_id = open(os.path.join(self.path, fina), 'r')
        data = np.zeros((self.ncols, self.nrows, self.nframes)) # (130,134,500)

        for i in range(self.nframes):
            data[:,:,i] = np.load(f_id)
            data[:,:,i] = data[:,:,i]/ self.monitor[i]
            self.log('Frame ' + str(i) + ' was normalized by ' + str(self.monitor[i]))
            data[:,:,i][data[:,:,i]<(self.LLD)] = self.notzero

        f_id.close()
        s = data.shape
        self.tau = ... tau = [1,2,3,4,5,6,7,8,9,10,12,14,16,20]
       ...

        IP = np.zeros(s,np.float64)
        IF = np.zeros_like(IP)
        G2 = np.zeros_like(IP)
        counter = np.zeros_like(self.tau)

        for t in range(len(self.tau)):
            delta = self.tau[t]
            for i in range(data.shape[2]-delta):

                if (self.monitor[i] >= self.treshold) and (self.monitor[i + delta] >= self.treshold) and (self.
monitor[i] <= self.treshold_up) and (self.monitor[i + delta] <= self.
treshold_up):

                    np.add(G2[:,:,t],np.multiply(data[:,:,i], data[:,:,i+delta]),G2[:,:,t])
                    np.add(IP[:,:,t],data[:,:,i],IP[:,:,t])
                    np.add(IF[:,:,t],data[:,:,i+delta],IF[:,:,t])
                    counter[t] += 1
       ...

        to_be_deleted = []
        for i in range(0,len(self.tau)):
            if counter[i] == 0:
                to_be_deleted.append(i)
        self.tau = np.delete(self.tau,to_be_deleted)
        print "tbd = ", to_be_deleted
        counter = np.delete(counter, to_be_deleted)
        G2 = np.delete(G2,to_be_deleted, axis = 2)
        IP = np.delete(IP,to_be_deleted, axis = 2)
        IF = np.delete(IF,to_be_deleted, axis = 2)
        for i in range(0,len(self.tau)):
            G2[:,:,i] = G2[:,:,i]/counter[i]
            IP[:,:,i] = IP[:,:,i]/counter[i]
            IF[:,:,i] = IF[:,:,i]/counter[i]
            IP[:,:,i][IP[:,:,i] < self.notzero]  = self.notzero
            IF[:,:,i][IF[:,:,i] < self.notzero]  = self.notzero
       ...

        f_id = open(path_G2, 'w')
        for i in range(G2.shape[2]):
            np.save(f_id, G2[:,:,i])
        f_id.close()

        f_id = open(path_IP, 'w')
        for i in range(IP.shape[2]):
            np.save(f_id, IP[:,:,i])
        f_id.close()

        f_id = open(path_IF, 'w')
        for i in range(IF.shape[2]):
            np.save(f_id, IF[:,:,i])
        f_id.close()
```

References

- Algorithms for Time Correlation Experiments
- Weekly progress of the IDPE project for TCE