# XTCAV processing on psana

The script used to create the data files is scripts/xtcav_process.py

To execute connect to pslogin, then ssh psana, then e.g.

```
bsub -J XCAV.186 -q psanaq -o XCAV.186.log -n 10 mpirun python
xtcav_process.py 186
```

or you can launch multiple with this one-liner:

```
for r in {1..186}; do bsub -J XT.$r -q psanaq -o
/path/to/logfolder/XCAV.$r.log -n 2 mpirun python xtcav_process.py $r;
done
```

It is here for preservation:

```
import sys,os

from scipy.signal import argrelmax
from sklearn.cluster import KMeans
import numpy as np

import psana
from xtcav2.LasingOnCharacterization import LasingOnCharacterization
xx = LasingOnCharacterization()

##############
run = int( sys.argv[1] )
exp = "cxilu5617"
outdir = "/reg/d/psdm/cxi/cxilu5617/scratch/xtcav/batch.data.with.trace"
power_max = 10000 # sometimes power analysis diverges, suppress this
case
max_order = 3 # parameter describing local maxima neighborhood in power
spectrum, increase to lower the local max detected
##############

if not os.path.exists(outdir):
    os.makedirs( outdir)

outname = os.path.join( outdir, "run%d_xtcav.h5"%run)

def smooth(x, beta=10.0, window_size=11):
    """a nice recipe"""
    if window_size % 2 == 0:
        window_size += 1
    s = np.r_[x[window_size-1:0:-1], x, x[-1:-window_size:-1]]
    w = np.kaiser(window_size, beta)
    y = np.convolve( w/w.sum(), s, mode='valid' )
    b = (window_size-1) / 2
    smoothed = y[b:len(y)-b]

    return smoothed

ds = psana.MPIDataSource("exp=%s:run=%d:smd"%(exp,run))
smldata = ds.small_data(outname) # , gather_interval=100)

# for finding the  max position(s) in power spectrum
# we use Kmeans
k_for_max = KMeans(n_clusters=2)

events = ds.events()
####
# first iterate and figure out how long the traces are
for i_ev, ev in enumerate(events):
    xx.processEvent(ev)
    pulse_t, pulse_power = xx.xRayPower()
    if pulse_power is not None:
        Npts = pulse_power[0].shape[0]
        break
    else:
        print "skipping event %d"%i_ev

#events = ds.events()
dummie = np.zeros( Npts)

for i_ev, ev in enumerate( events):
    if ev is None:
        continue
    xx.processEvent(ev)

    pulse_t, pulse_power = xx.xRayPower()
    if pulse_t is None or pulse_power is None:
        print "Power is None %d"%i_ev
        first_pulse_power =  -1 #np.nan
        second_pulse_power =  -1 #np.nan
        pulse_separation =  -1 #np.nan
        first_pulse_time = -1 #np.nan
        second_pulse_time = -1 #np.nan
        first_pulse_max_position = -1 #np.nan
        second_pulse_max_position = -1 #np.nan
        is_separated = -1 #np.nan
        pulse_t = dummie
        pulse_power = dummie
        smooth_power = dummie
    else:
        pulse_t = pulse_t[0]

#       filter/cleanup the power trace
        pulse_power = np.nan_to_num( pulse_power[0] )
        pulse_power[ pulse_power > power_max] = 0

#       work with a smoothed power trace
        smooth_power = smooth( pulse_power)

#       finds all local max in power trace
        maxpos = argrelmax( smooth_power, order=max_order)[0]

#       handle use cases
        if len( maxpos) ==0 :
            first_pulse_power = -1# np.nan
            second_pulse_power = -1# np.nan
            pulse_separation =  -1#np.nan
            first_pulse_time = -1#np.nan
            second_pulse_time = -1#np.nan
            first_pulse_max_position = -1#np.nan
            second_pulse_max_position = -1#np.nan
            is_separated = -1#np.nan
            print "No max found baby %d"%i_ev
        elif len( maxpos) == 1:
            first_pulse_power =  smooth_power[maxpos[0]]
            second_pulse_power = -1# np.nan
            first_pulse_time =  pulse_t[ maxpos[0] ]
            second_pulse_time = -1#np.nan
            pulse_separation =  -1#np.nan

            first_pulse_max_position =  maxpos[0]
            second_pulse_max_position = -1#np.nan
            is_separated =  0 #False
            print "I see one peak in the power spectrum babe; %d"%i_ev
        else:
            k_for_max.fit( maxpos[:,None] )
            maxpos1 = maxpos[k_for_max.labels_==0]
            maxpos2 = maxpos[k_for_max.labels_==1]

            maxVals = smooth_power[maxpos]
            maxVals1 = maxVals[k_for_max.labels_==0]
            maxVals2 = maxVals[k_for_max.labels_==1]

#           the true peak is the max position within each cluster
            X  = maxpos1[ np.argmax( maxVals1  )]
            X2 = maxpos2[ np.argmax( maxVals2  )]
            X,X2 = np.sort( [X,X2])

            first_pulse_power =  smooth_power[X]
            second_pulse_power =  smooth_power[X2]
```

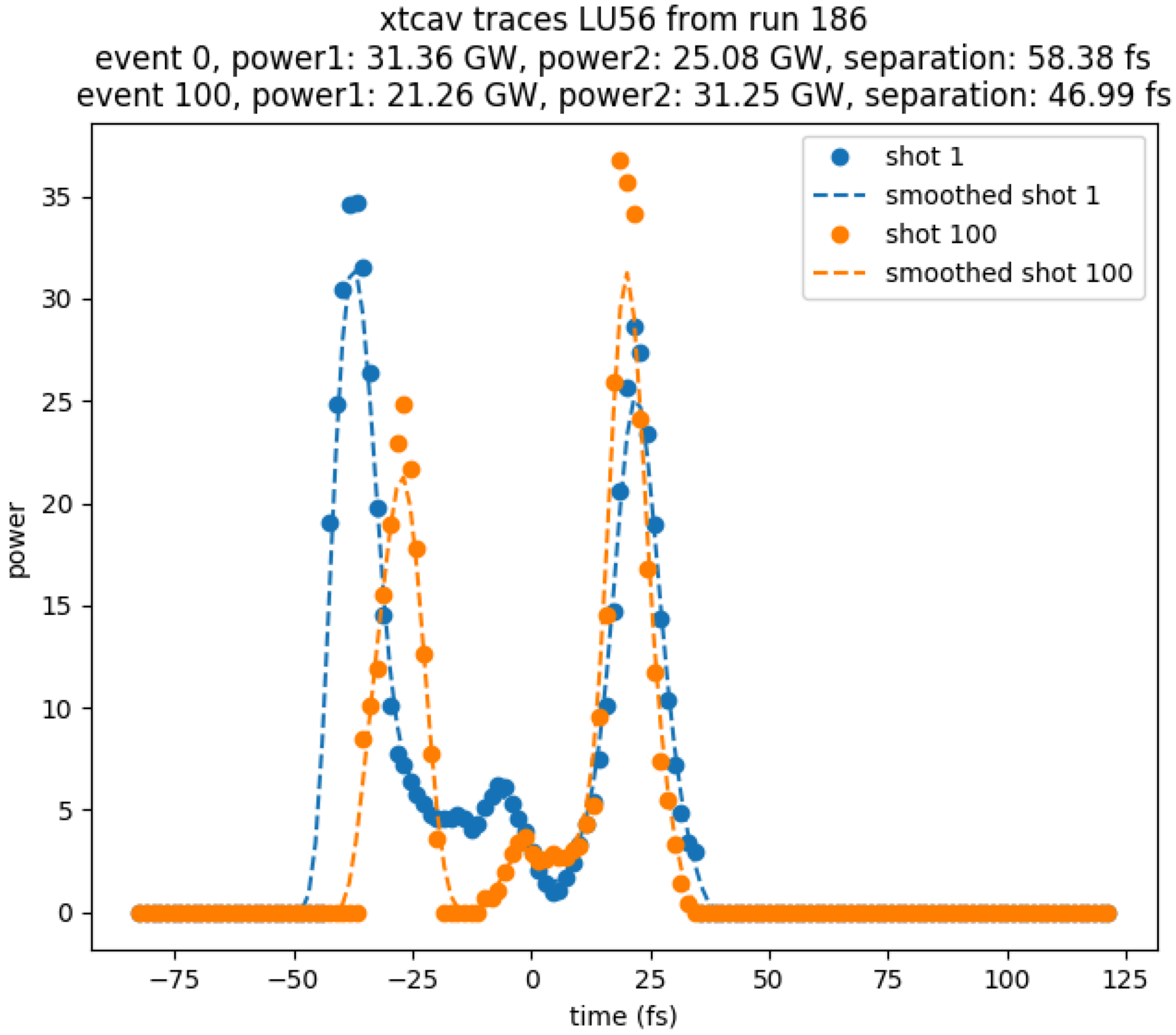
```
                first_pulse_time =  pulse_t[ X ]
                second_pulse_time = pulse_t[X2]
                pulse_separation =  pulse_t[X2] - pulse_t[X]
                first_pulse_max_position =  X
                second_pulse_max_position =  X2
                is_separated = 1 #True
                print "Looks like 2 peaks to me.. %d"%i_ev
        smldata.event( first_pulse_power = first_pulse_power,
                second_pulse_power = second_pulse_power,
                first_pulse_time = first_pulse_time,
                second_pulse_time = second_pulse_time,
                pulse_separation=pulse_separation,
                first_pulse_maxpos = first_pulse_max_position,
                second_pulse_maxpos = second_pulse_max_position,
                is_separated=is_separated,
                power_trace = pulse_power,
                smoothed_power_trace = smooth_power, pulse_t=pulse_t)
```

```
    #print "processed %d events"%i_ev
smldata.save()
```

The h5 file produced has a lot of useful info, here is a brief example of how to access the data inside:

```
import h5py
from pylab import *

f =
h5py.File('/reg/d/psdm/cxi/cxilu5617/scratch/xtcav/batch.data.with.trace
/run186_xtcav.h5','r')

# display some traces
plot(f["pulse_t"][0], f["power_trace"][0] , 'o', color='C0')
plot(f["pulse_t"][0], f["smoothed_power_trace"][0] , '--', color='C0')
plot(f["pulse_t"][100], f["power_trace"][100] , 'o', color='C1')
plot(f["pulse_t"][100], f["smoothed_power_trace"][100] , '--',
color='C1')
legend( ("shot 1", "smoothed shot 1", "shot 100 ", "smoothed shot 100"))
xlabel("time (fs)")
ylabel("power")

# get information on the shots..
info0 = 0, f['first_pulse_power'][0], f["second_pulse_power"][0],
f["pulse_separation"][0]
info100 =100, f['first_pulse_power'][100], f["second_pulse_power"][100],
f["pulse_separation"][100]
info_s = "event %d, power1: %.2f GW, power2: %.2f GW, separation: %.2f
fs"


title_s = "xtcav traces LU56 from run 186\n" +info_s%info0 + "\n" +
info_s%info100
title(title_s)


show()
```

If you run the above code on psana it should produce the following plot



xtcav traces LU56 from run 186
event 0, power1: 31.36 GW, power2: 25.08 GW, separation: 58.38 fs
event 100, power1: 21.26 GW, power2: 31.25 GW, separation: 46.99 fs