# Automatic analysis of multi grid detector data

This is a report on the effort to produce usable data from the multigrid experiments at CNCS. The effort has focused on producing a number of tools to parse the raw data files, extract calibration data from the readout system, generate detailed statistics and detector pixel id’s. Data is produced in ascii format and some visualization tools for generating graphs, histograms and detector images are also presented. All of these tools can be scripted or used manually depending on the use case. The programs and scripts works on Linux and Mac and have very few dependencies.

## The data material

The currently available data is located in the following directories on the **login server** relative to the base directory **/nfs/groups/multigrid/data/raw/MG\_CNCS**. The directory structure for the data is shown below.

.

├── 07\_11

│   ├── beamOn\_resetOn

│   └── no\_reset

├── 07\_12\_background

├── 07\_13\_12A

├── 07\_13\_12A\_Vanadium\_powder

├── 07\_13\_4p96A

├── 07\_13\_7p2A

│   └── 1\_t0\_timing

├── 07\_14

├── 07\_15

├── 07\_15\_converted

├── 07\_25

├── 08\_16

├── 09\_29

├── 10\_13

├── 11\_29

└── MCA4\_data

The directories have the following characteristics

Total number of files 210711

Number of data files (.bin) 209761

The data files are grouped in approximately 29 runs with identical filenames and increasing ‘serial numbers’. These runs account for 99.6% of the total number of data files. Details of the runs are specified in a JSON file located in the **event-processing-sw-experiments** repository in the file **dataformats/cncs2016/allfiles.json**.

The number of files in the runs range from 2 to 60478. A small number of files have been ignored. Either because they seem to contain too little information for analysis, or because they look like they were produced during setup and configuration of the experiment.

Of the 29 runs, 19 produce valid calibration tables for grid and wire ids with the current zero-knowledge peak finder algorithm. This corresponds to 160420 files or ~76% of all data files. This subset of runs is also listed in the **allfiles.json** file in an object named **validruns**.

The initial batch analysis can be run by a single command

**> batchreader -b /data/MG\_CNCS/ -r allfiles.json -j validruns**

It takes about one hour to complete on a Xeon E5-2620 v3 CPU. Since each run can be analysed independently of other runs, this could be parallelized if necessary, for example using four shells and using the options to restrict parsing to a range of runs:

**shell1> batchreader -b /data/MG\_CNCS/ -r allfiles.json -j validruns –s 1 –e 4**

**shell2> batchreader -b /data/MG\_CNCS/ -r allfiles.json -j validruns –s 5 –e 8**

**shell3> batchreader -b /data/MG\_CNCS/ -r allfiles.json -j validruns –s 9 –e 12**

**shell4> batchreader -b /data/MG\_CNCS/ -r allfiles.json -j validruns –s 13 –e 16**

For each successful run, the analysis produce the following file types:

**.events** contains readout values for all valid events

**.csv** contains stats for each file of the run as well as global stats

**.hist** contains histogram data for heatmap generation

**.gcal** contains grid calibration data which can be read by the cspec pipeline and

the pixel id generator.

**.wcal** contains wire calibration data which can be read by the cspec pipeline and

the pixel id generator.

In addition, some results from the peakfinder algorithm is printed to standard output.

## Peak Positions

The peakfinder results are deemed valid when it identifies 128 wire positions and 96 grid positions for the same run as this corresponds to the physical detector configuration.

The output of the batchreader provides some details of the peak positions calculated from the readout data. For **wire0** and **grid0** positions the start and end positions, peak and gap width statistics are provided.

From Table 1 it is seen that the 19 runs falls in three different calibration regimes. Each regime has roughly the same calibration parameters with only small differences. In all cases the peaks are well defined with minimum gaps between 3 and 12 (most being above 8).



Table 1: Peak positions and number of events for “validruns”.



Table 2: Peak positions for “invalidruns”.

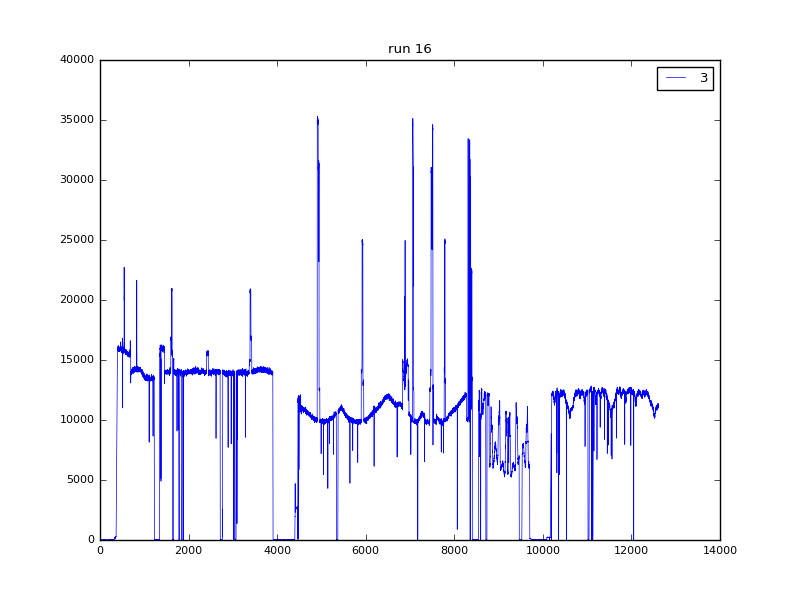
In Table 2 is listed the results from batch processing which fails to identify 128 wire ids and 96 grid ids. For some of these runs (1, 2, 3, 6) using a calibration file from one of the “validruns” might be able to produce some valid results. But this must be done with care. For other runs, (4, 5, 7, 8, 9, 10) this approach seems unlikely.

## Finding regions of interest

The 19 runs generate a total of about 469M events. But events are not distributed equally over all files in a run as seen in Table 1. For example run 7 has more than 60.000 files but generates 90M events, whereas run 16 has 12.600 files and generates 128M events. This naturally depends on the experimental setup.

To visualize regions of interest use the script **multiplot.py**

**> multiplot.py –t "run 16" -c "3" ../output2/2016\_12\_06\_1059\_sample\_.csv**



and zoom in on relevant peaks to find a specific file range.

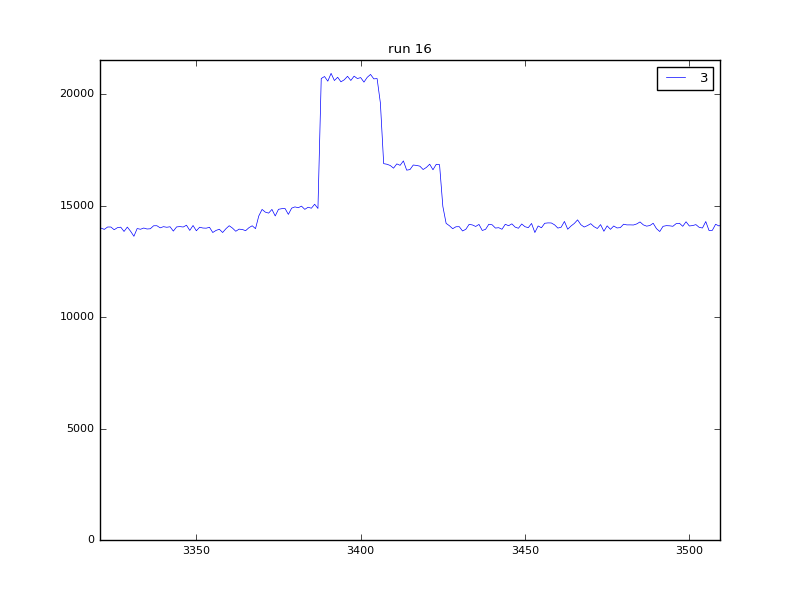


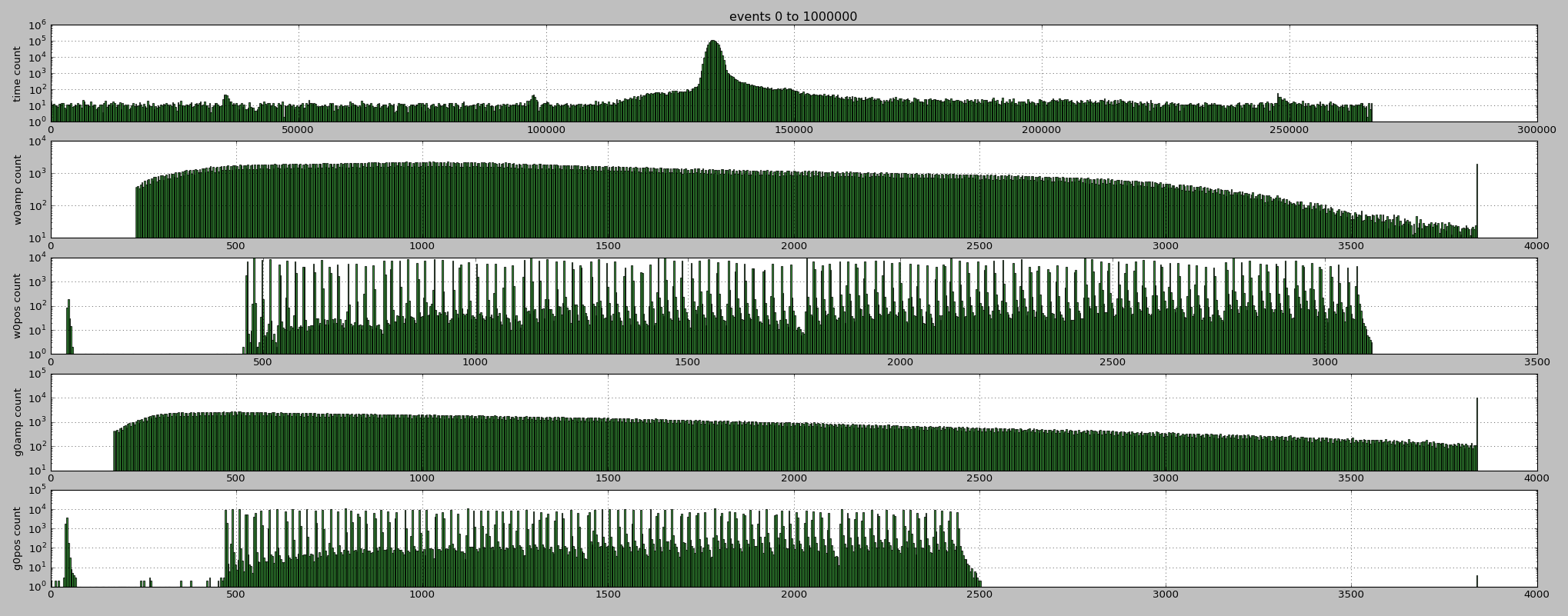
Figure 1: Range identification by visual inspection. The largest peak starts at file index 3388 and ends at 3404, a very small subset of the 12616 files in this run.

Based on allfiles.json as the template a json file should/could be created which contains the identified regions of interests. This file can then be used for further analysis.

## Histograms

To generate a histogram of time-of-flight values and wire and grid positions use the script **histogram.py**

**> histogram.py ../output2/2016\_12\_06\_1059\_sample\_.events 1000000**



## Pixel IDs and detector images

To generate the calculated detector pixel id’s from the automatic calibration files and the event data use the program **genpixids**

**> genpixids output2/2016\_12\_06\_1059\_sample\_**

Loading calibration data..

Creating MultiGrid parser objects..

Applying calibration data..

reading event data...

event: 1, wirepos 922, wire: 24, gridpos 2378, grid: 45, pixel: 823

event: 2, wirepos 557, wire: 6, gridpos 1883, grid: 22, pixel: 421

event: 3, wirepos 2513, wire: 101, gridpos 1217, grid: 86, pixel: 4774

...

**genpixids** generates the following files

.**vox** the intensity/event count for a given pixelid

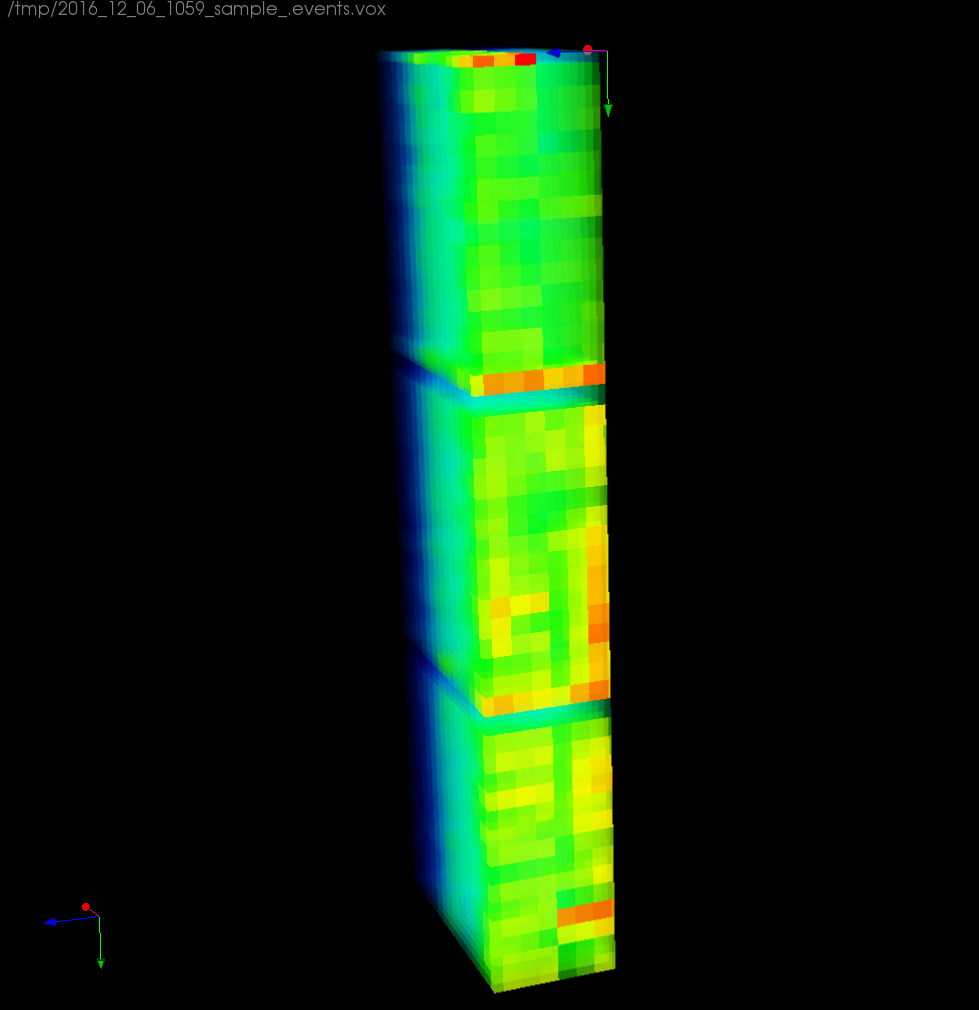
**.xyproj** projection of the intensities in the x-y plane

**.zyproj** projection of the intensities in the z-y plane

**.xzproj** projection of the intensities in the x-z plane

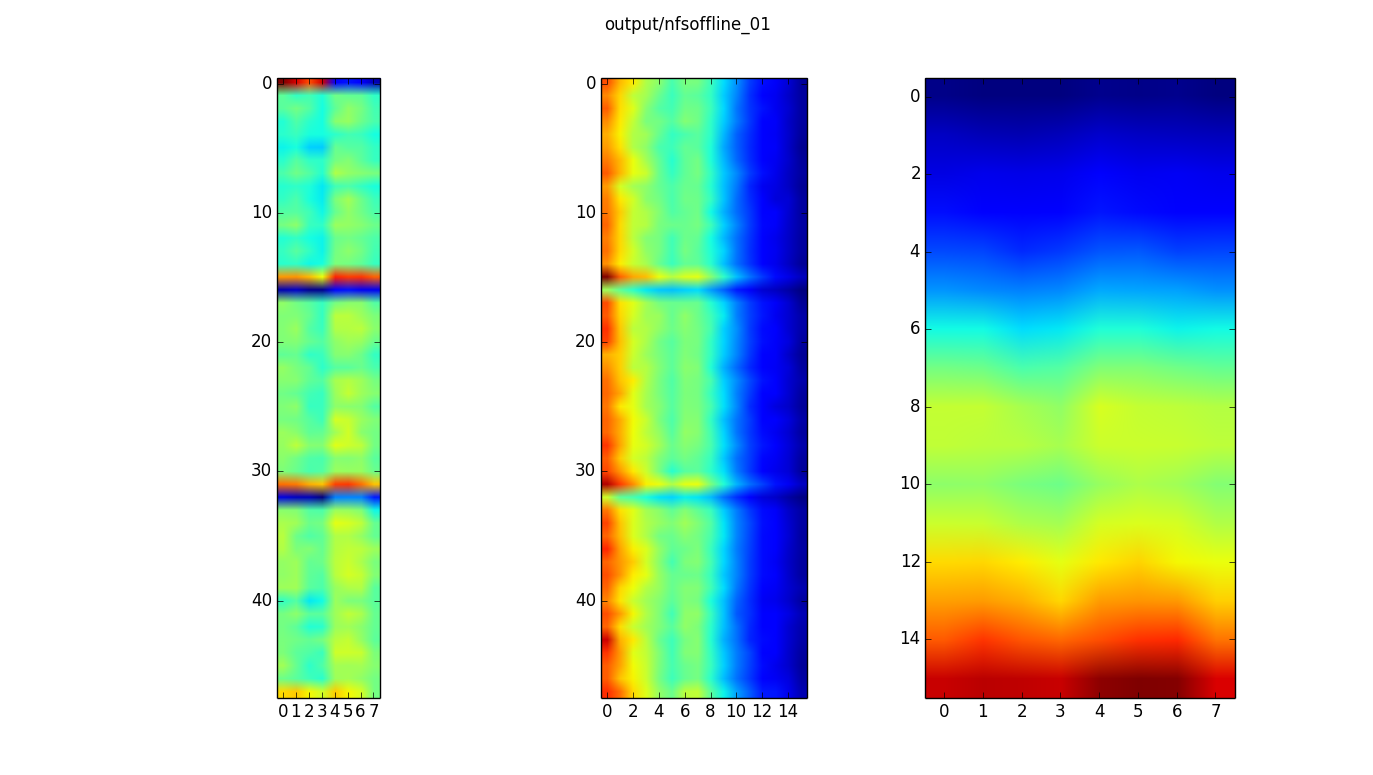
The **.vox** can be used to visualize the voxel intensity for the multigrid detector use the script **voxelrender.py**

**> voxelrender.py /tmp/2016\_12\_06\_1059\_sample\_.events.vox**



To see the intensity projections run the command

**> projections.py output/validruns\_1**



## Heatmaps

To visualize the wire0 positions as a heatmap use the script **heatmap.py**

**> heatmap.py output/validruns\_1.hist**

|  |
| --- |
|  |