Test Beam with Python

Luigi Vigani – University of Oxford

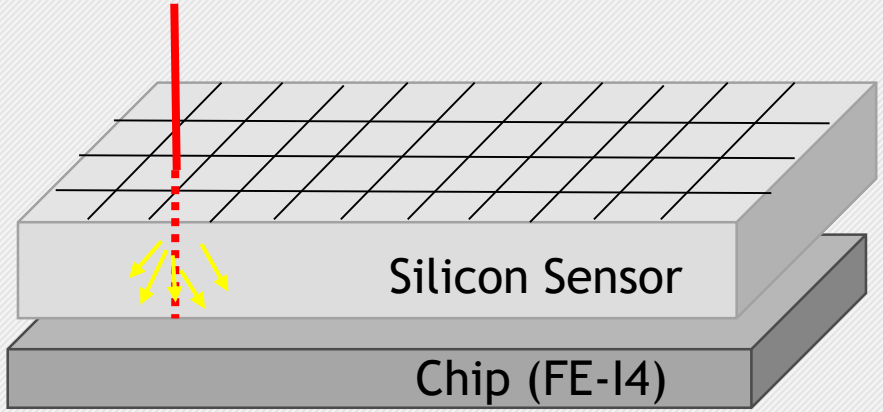
Tomasz Hemperek, Toko Hirono, Jens Janssen, David-Leon Pohl - Silab Bonn

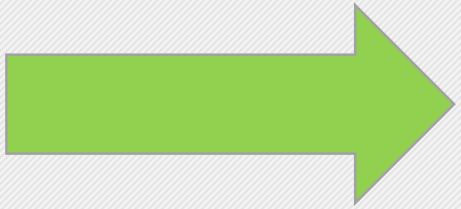
2nd Developers@CERN Forum

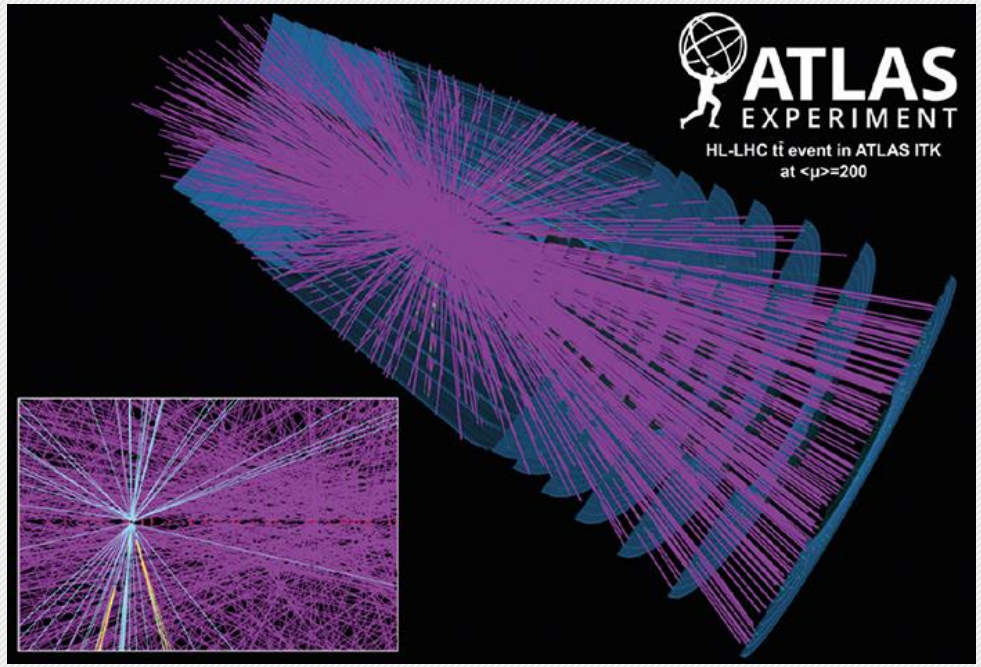
31st May 2016

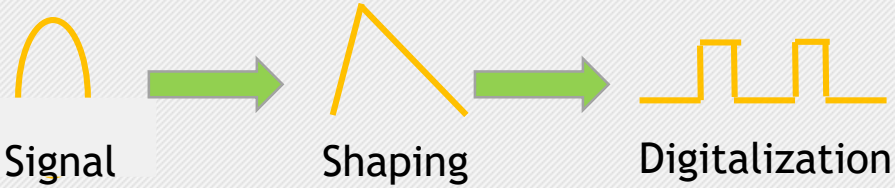
The Context

Silicon Pixel Detectors







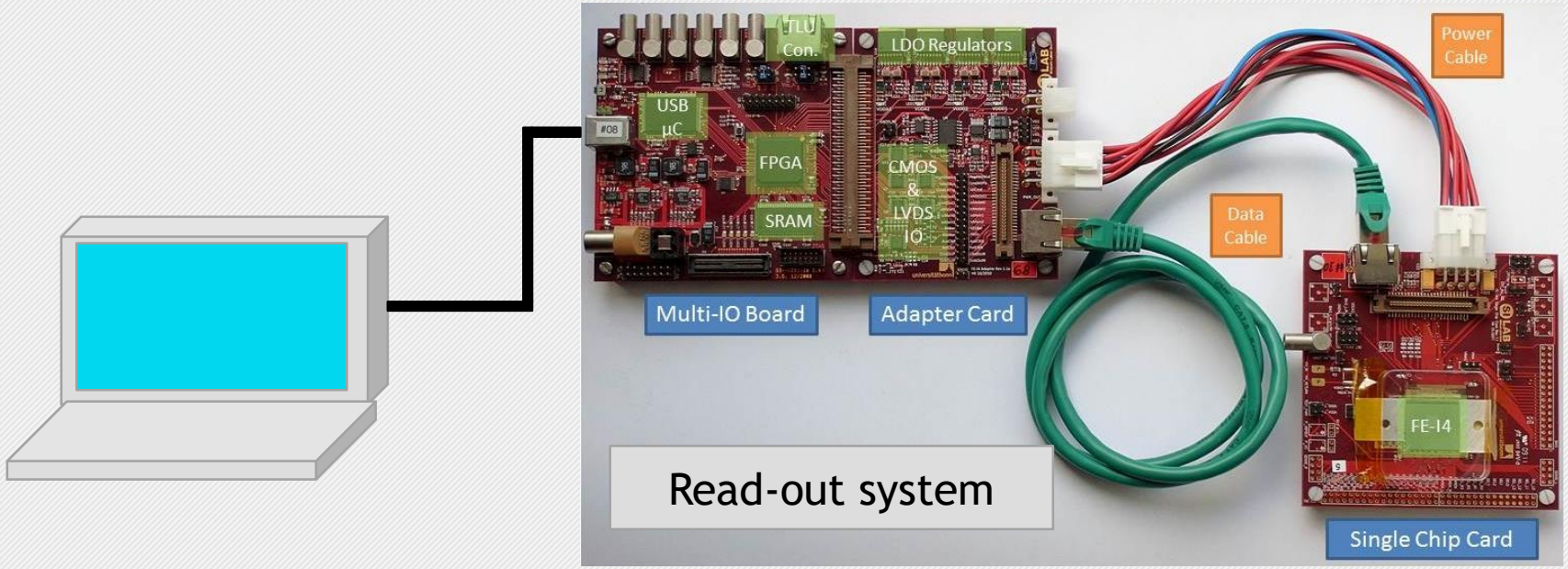


Atlas Tracker

Single Chip Read-out System

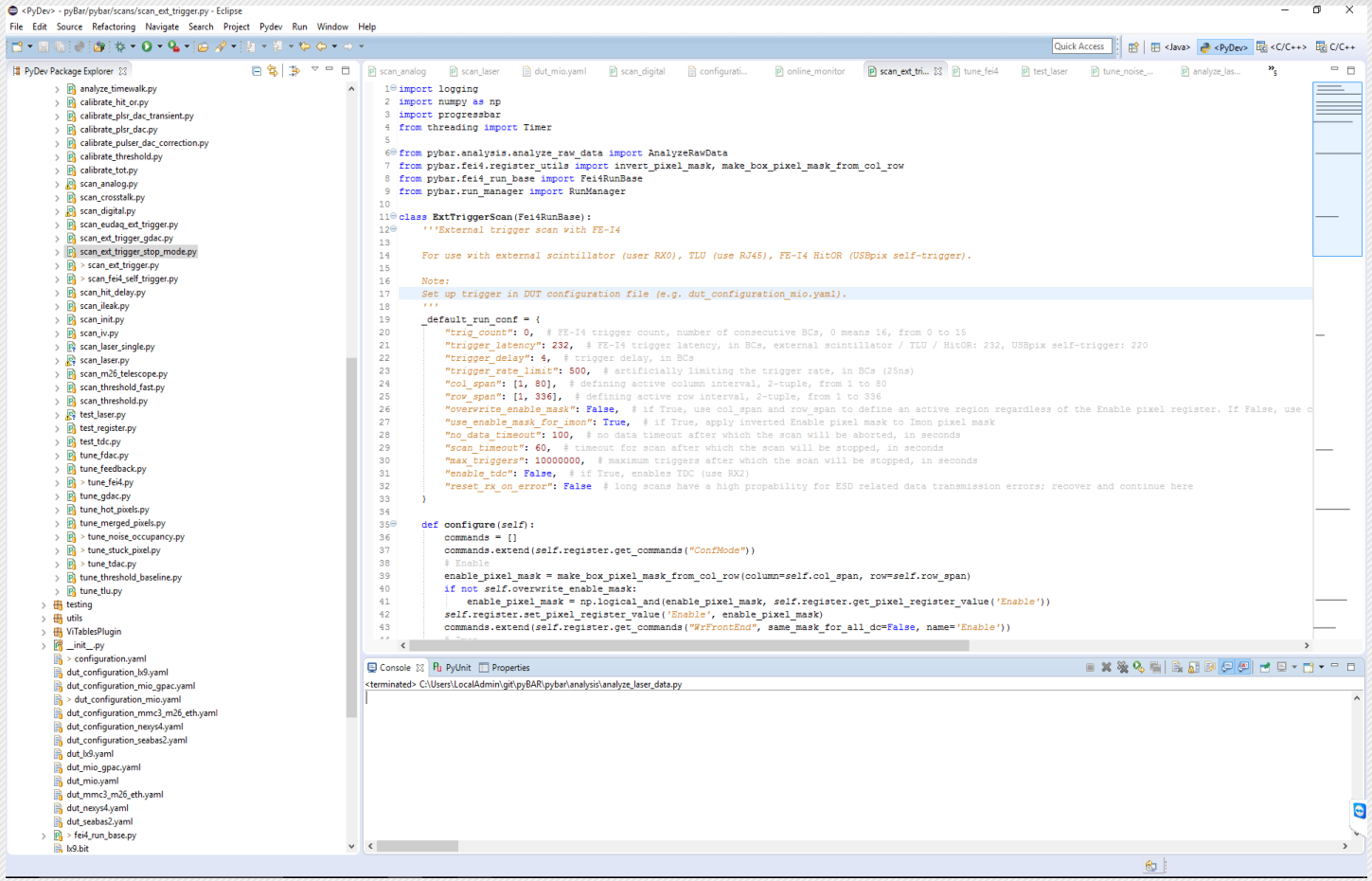
Data acquisition software: Basil + pyBAR (Bonn Atlas Readout)
(https://github.com/SiLab-Bonn/pyBAR)

- Python framework
- C++ compiled parts
- Firmware integration in Basil
- Different hardware supported



- Characterize and define chip's operative parameters.
- Record hits from the sensor.

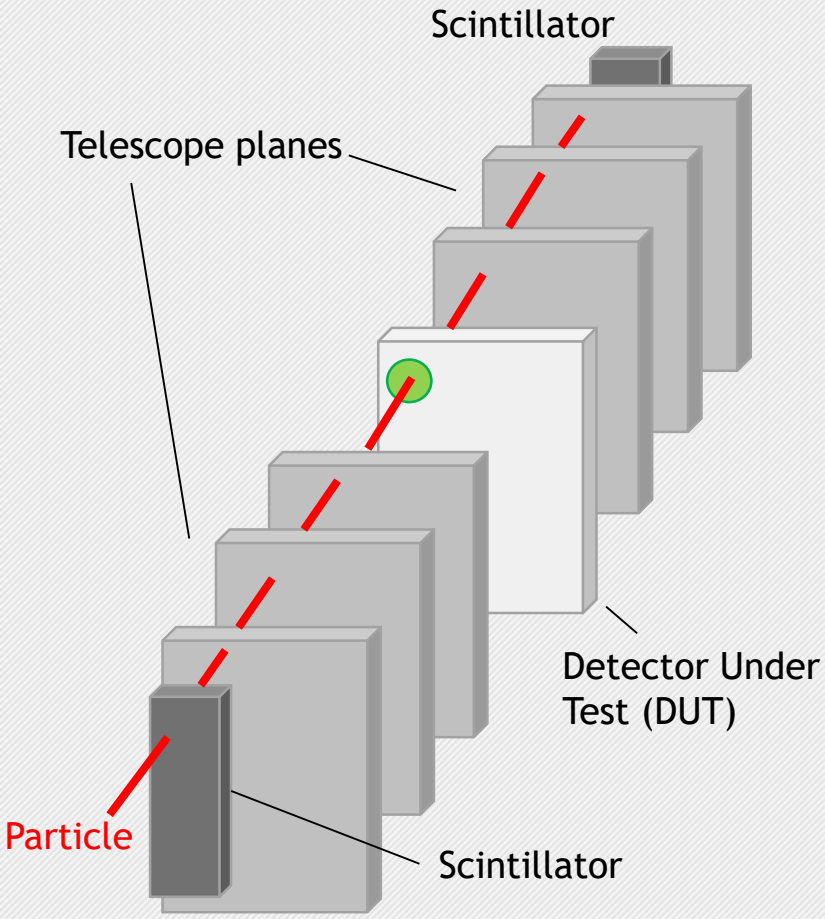
Overview on pyBAR



- Many Python packages needed. Suggestion:
  - Python packages installed with Anaconda (https://www.continuum.io/why-anaconda)
  - Integration with Eclipse (PyDev)
  - Compatible with any platform

- Each task (scan) performed by a script
- Classes based on Fei4RunBase
  - Parameters set in _default_run_conf
  - Three functions
    - configure
    - scan
    - analyze
- Data saved as HDF5 files, read with ViTables (http://vitables.org/) + BLOSC compression.

Test Beam



Test prototypes in conditions as close as possible to the experimental:

- Particle beam (Minimum Ionizing Particles)
- Well-known pixel detectors for tracking: telescope
- Prototype placed as DUT

Particle tracks must be related to hits on DUT:

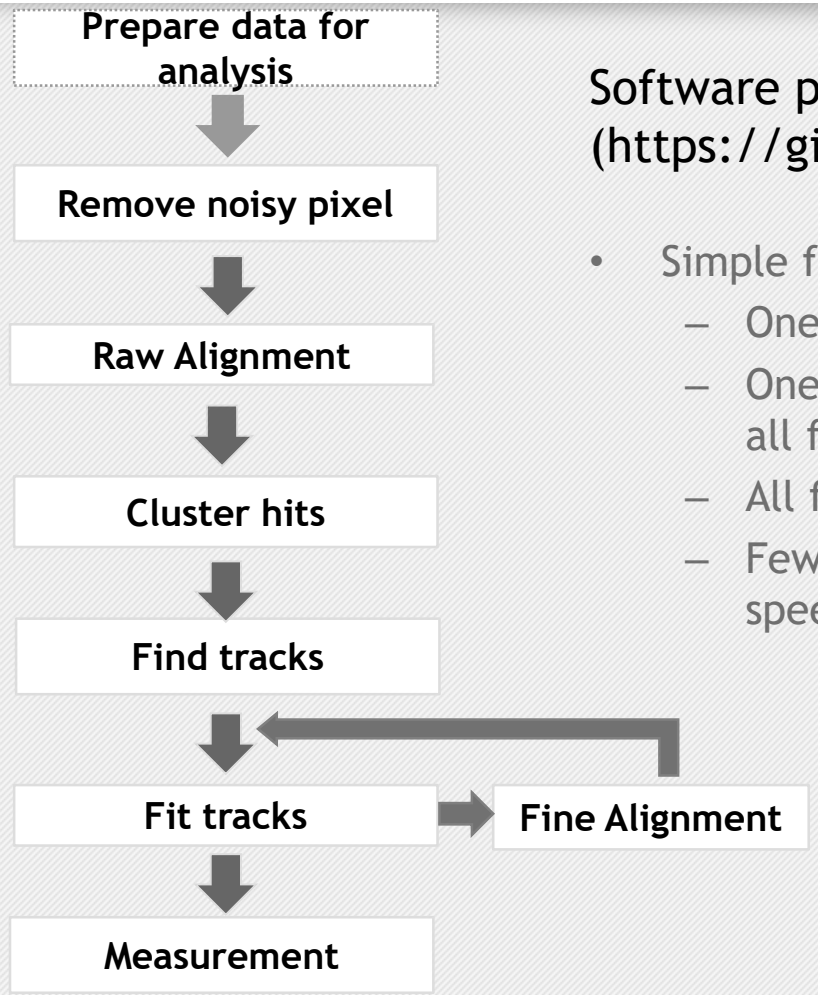
- Triggers from scintillators distributed to all planes

Two main options:

- EUTelescope with Mimosa26 planes
- Full FE-I4 telescope

(DUT is always run with FE-I4)

Analysis Workflow



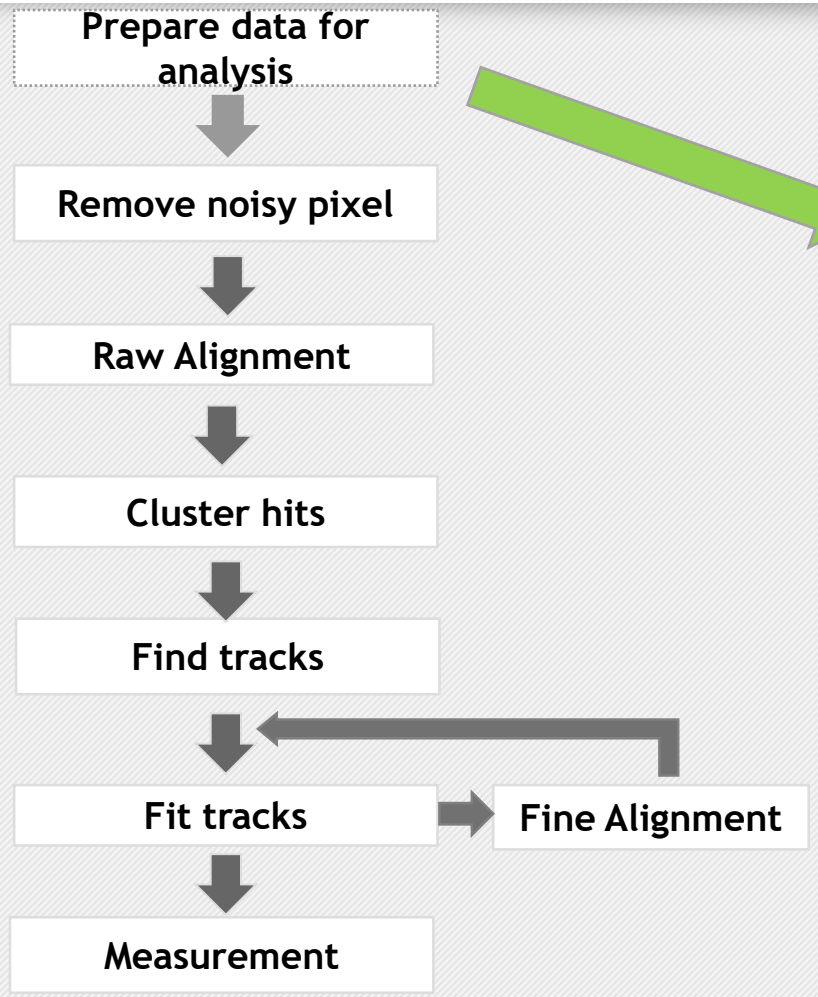
Software package: testbeam_analysis (https://github.com/SiLab-Bonn/testbeam_analysis)

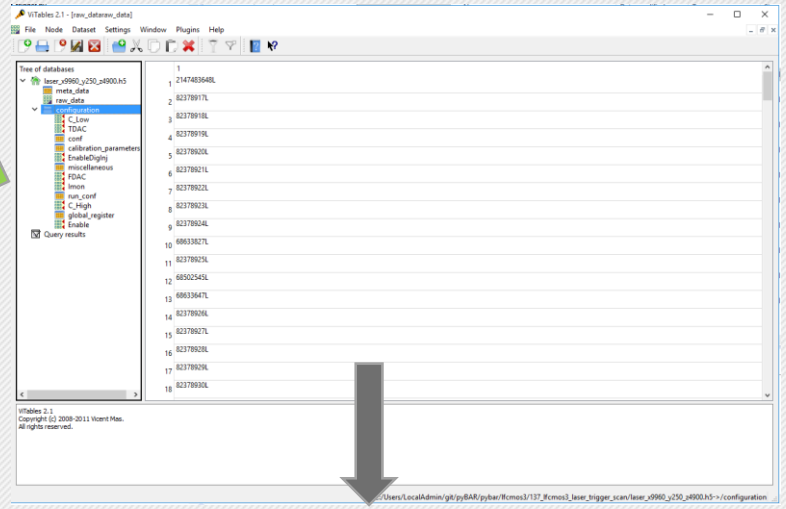
- Simple framework
  - One function per step
  - One meta script that execute all functions
  - All functions documented
  - Few C++ helper functions for speed up

Code example:

```
import testbeam_analysis.analysis as tba
from testbeam_analysis import plot_utils

if __name__ == '__main__':
    # The location of the data files, one file per DUT
    data_files = [r'data/TestBeamData_FEI4_DUT0.h5',  # the first DUT is the reference DUT defining the coordinate system, called internall
                  r'data/TestBeamData_FEI4_DUT1.h5',  # DUT1
                  r'data/TestBeamData_FEI4_DUT4.h5',  # DUT2
                  r'data/TestBeamData_FEI4_DUT5.h5'   # DUT3
                  ]

    # Dimensions
    pixel_size = (250, 50)  # in um
    z_positions = [0., 1.95, 10.88, 12.83]  # in cm; optional, can be also deduced from data, but usually not with high precision (~ mm)

    output_folder = os.path.split(data_files[0])[0]  # define a folder where all output data and plots are stored
    cluster_files = [data_file[:-3] + '_cluster.h5' for data_file in data_files]

    # The following shows a complete test beam analysis by calling the separate function in correct order

    # Correlate the row / column of each DUT
    tba.correlate_hits(data_files,
                       alignment_file=output_folder + r'/Correlation.h5',
                       fraction=1,
                       event_range=0)

    plot_utils.plot_correlations(alignment_file=output_folder + r'/Correlation.h5',
                                 output_pdf=output_folder + r'/Correlations.pdf')

    # Create alignment data for the DUT positions to the first DUT from the correlation data
    # When needed, set offset and error cut for each DUT as list of tuples
    tba.align_hits(correlation_file=output_folder + r'/Correlation.h5',
                   alignment_file=output_folder + r'/Alignment.h5',
                   output_pdf=output_folder + r'/Alignment.pdf',
                   fit_offset_cut=(2. / 10., 5.0),
```

Analysis Workflow





	event_number	frame	column	row	charge
1	2L	8	41	68	8
2	3L	8	61	133	10
3	4L	7	30	280	7

Raw data extracted from each chip…

Then converted with pyTables

Raw data coming from telescope and DUT with timestamps.
In each bunch:

- Hits from all planes as pixel addresses (row and column)
- Noise

Analysis Workflow

Prepare data for analysis

Remove noisy pixel

Raw Alignment

Cluster hits

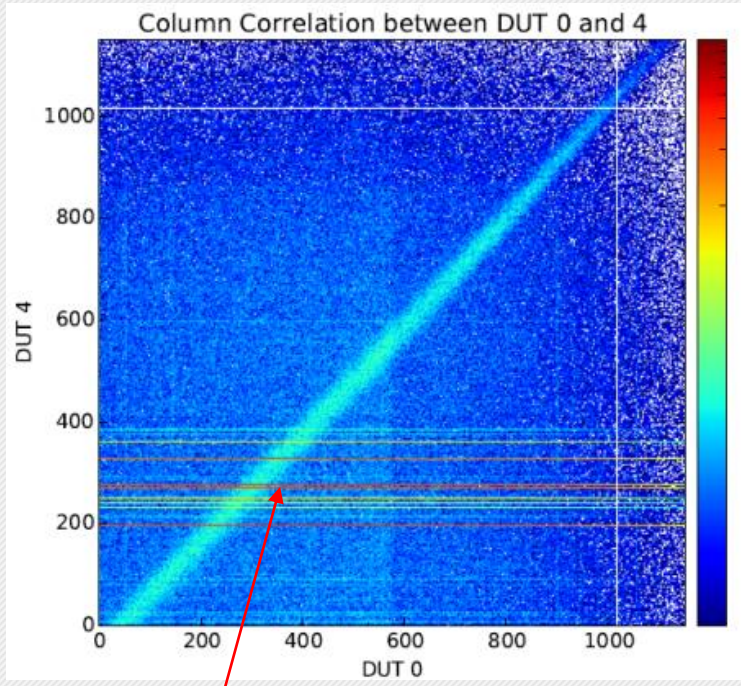
Find tracks

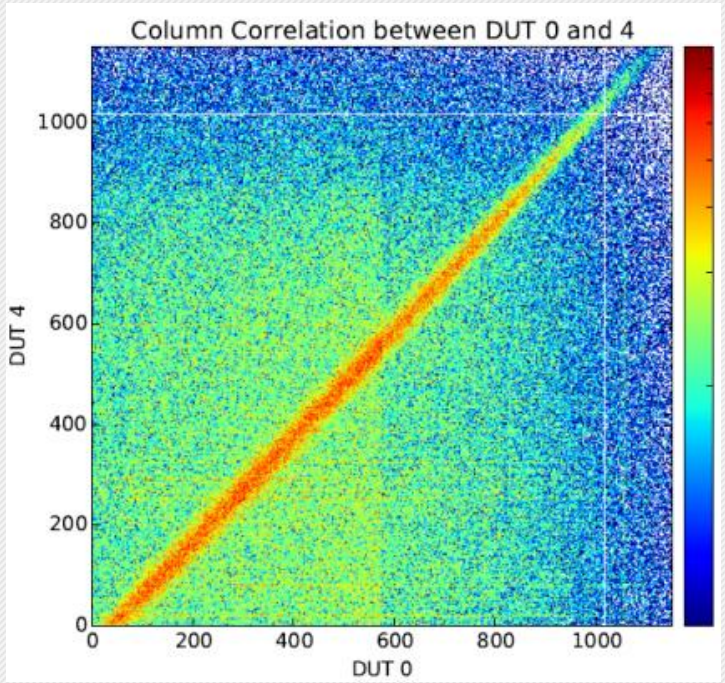
Fit tracks

Fine Alignment

Measurement

Plots produced with matPlotLib





This column on DUT4 has more hits than average wrt DUT0…
Noise! Remove noisy pixels from analysis.

Analysis Workflow

Prepare data for analysis

Remove noisy pixel

Raw Alignment

Cluster hits

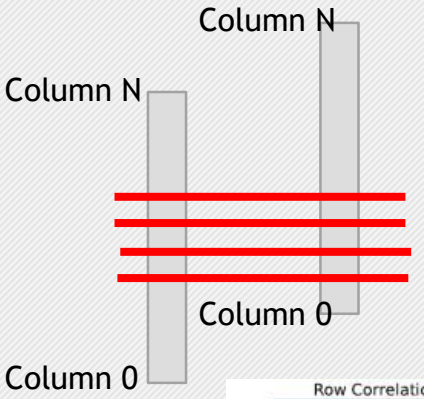
Find tracks

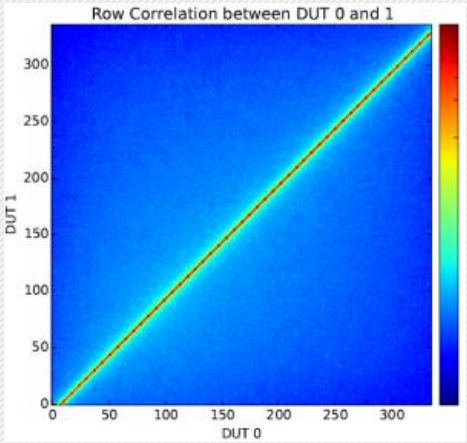
Fit tracks

Fine Alignment

Measurement

Looking at correlation between column (row) and column (row) between each plane and plane 0 (reference plane)

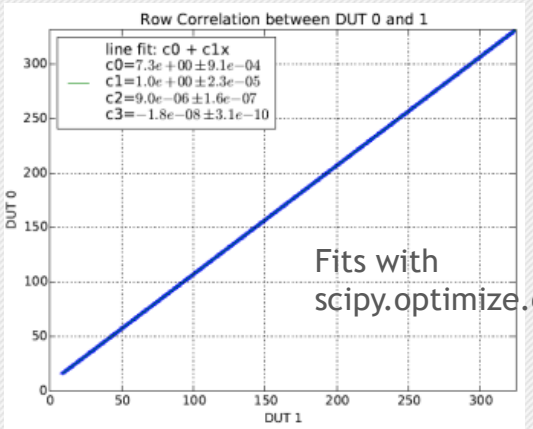




Displacement between planes

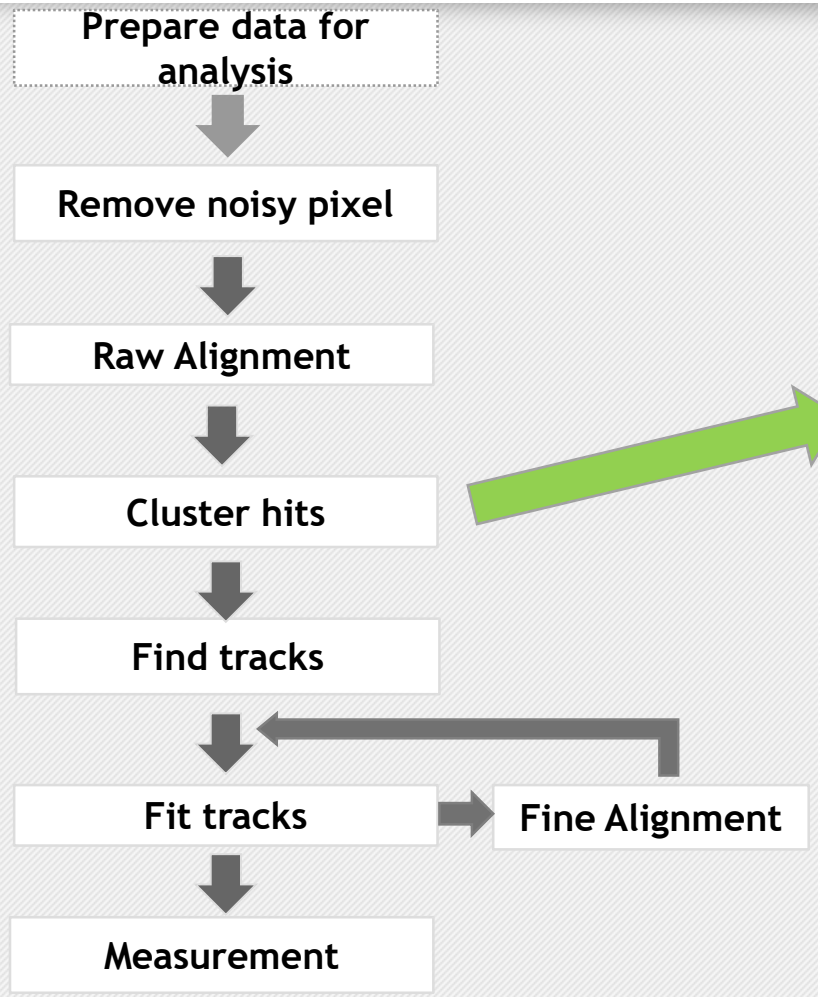
Offset between columns/rows

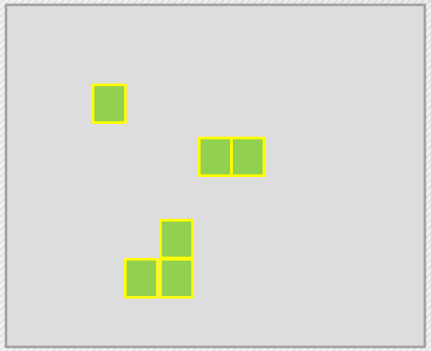
Average and fit



Fits with scipy.optimize.curve_fit

Analysis Workflow





A single particle may light up more than one adjacent pixel: cluster!

Clusterizer must:

- For each hit search for adjacent hits
- Merge them in a single unit (cluster) with number, charge position on the plane and its error (in μm)

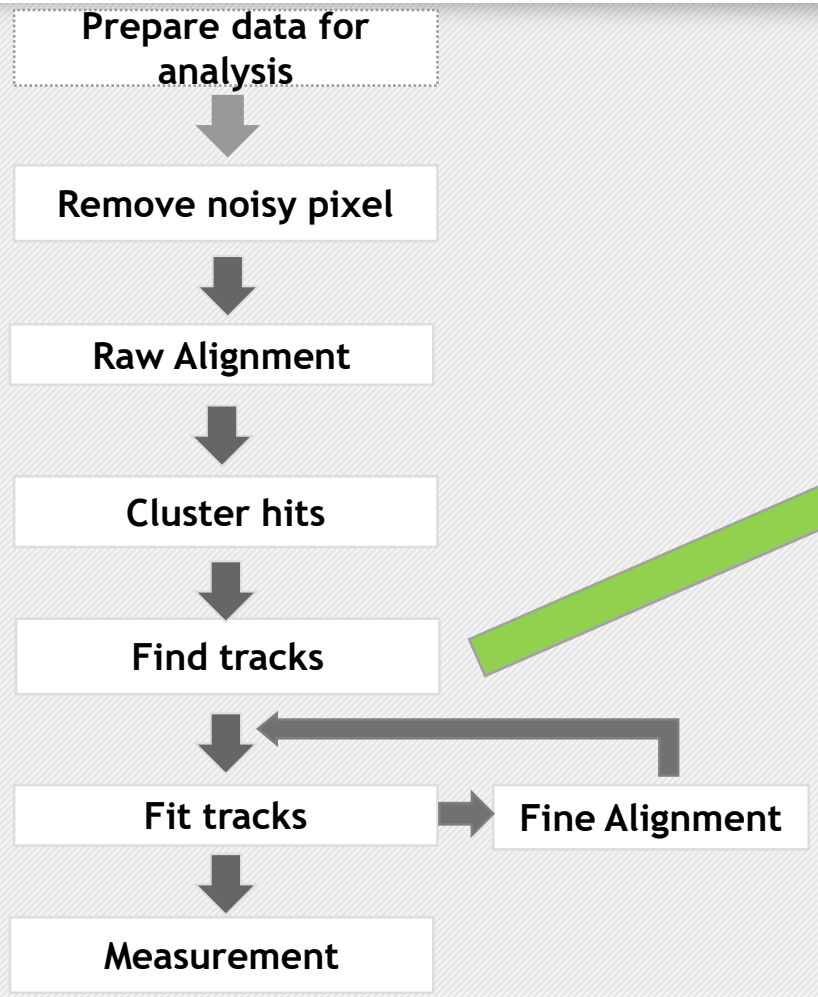
Very long process: using Numba JIT:

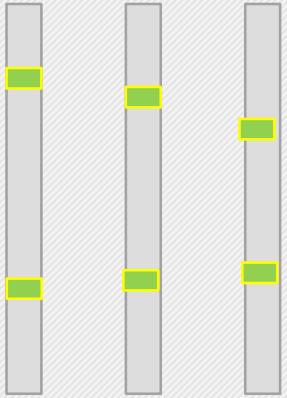
- Up till100 times faster
- 10 times less code than cython
- More flexible

Result:

	event_number	id	size	charge	seed_column	seed_row	mean_column	mean_row
6	8L	0	1	7	53	78	53.5	78.5
7	9L	0	1	8	23	219	23.5	219.5
8	9L	1	1	5	28	237	28.5	237.5
9	9L	2	1	8	53	321	53.5	321.5
10	10L	0	1	9	73	105	73.5	105.5

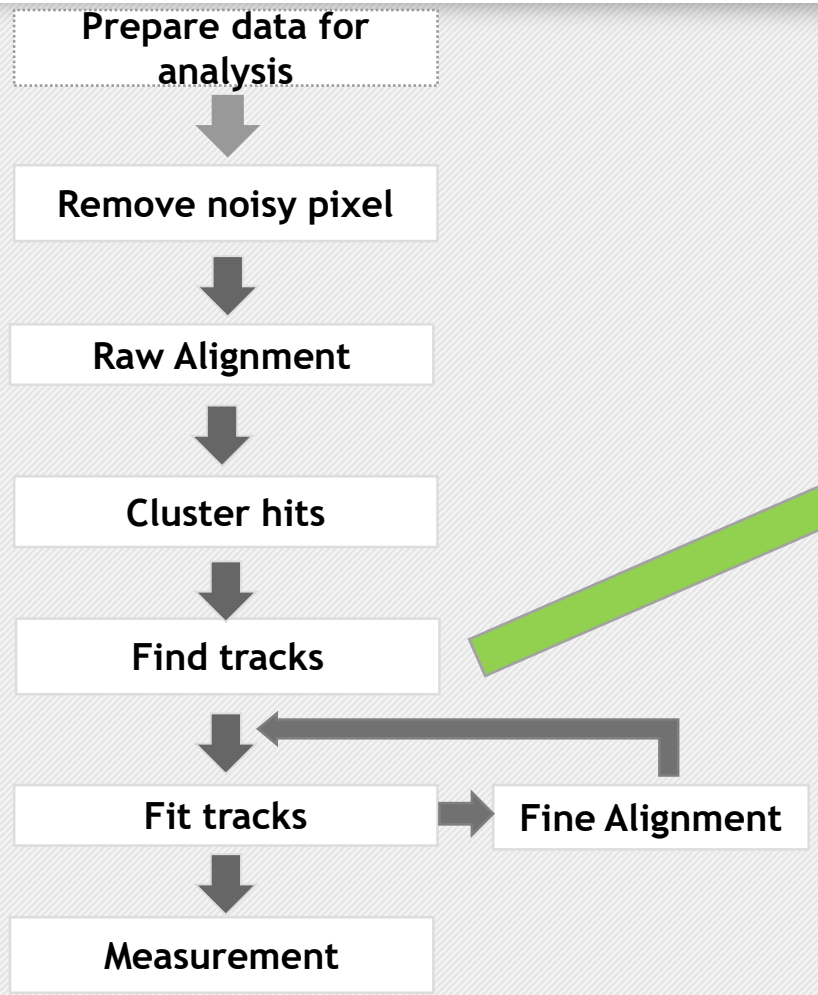
Analysis Workflow

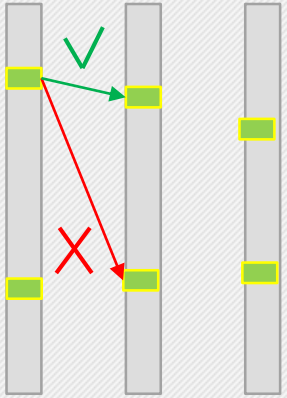




Multiple tracks can happen at the same time

Analysis Workflow

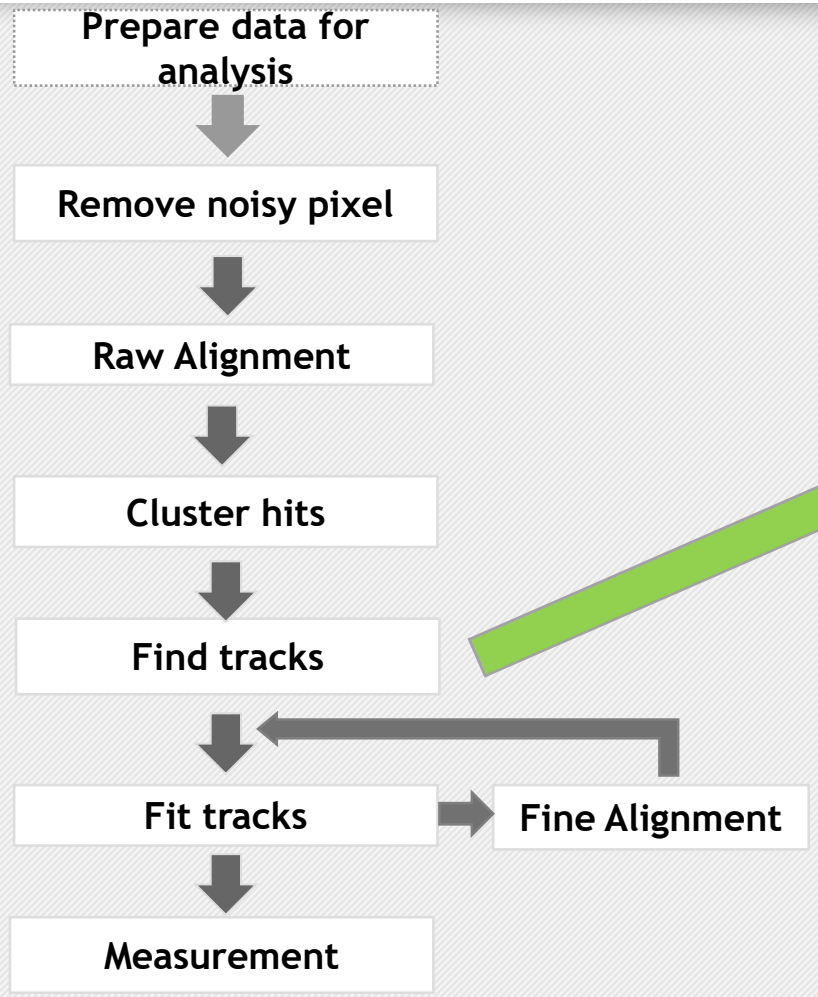


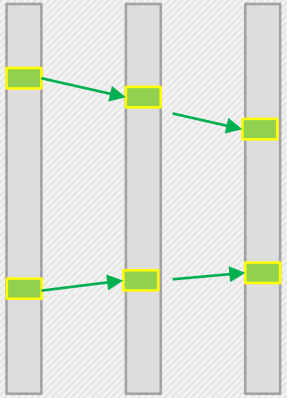


Multiple tracks can happen at the same time

Hits that belong to the same track are chosen depending on the distance between hits on aligned planes.

Analysis Workflow



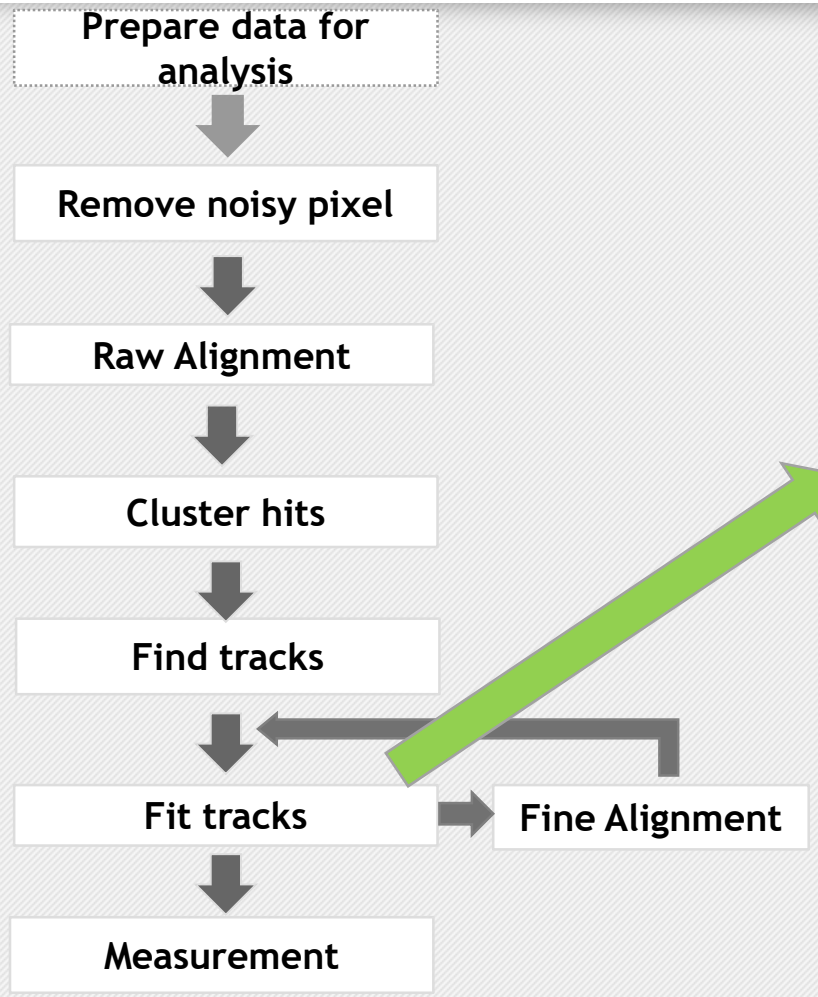


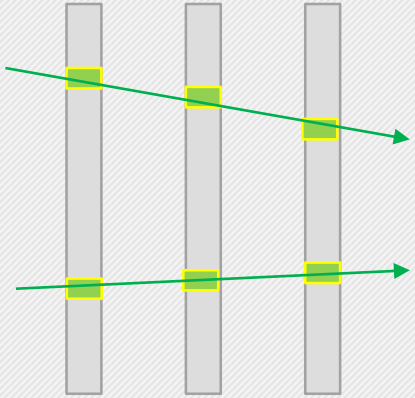
Multiple tracks can happen at the same time

Hits that belong to the same track are chosen depending on the distance between hits on aligned planes.

- Track quality factor depending on number of hits
- Multiprocessing (Pool) used to speed up

Analysis Workflow



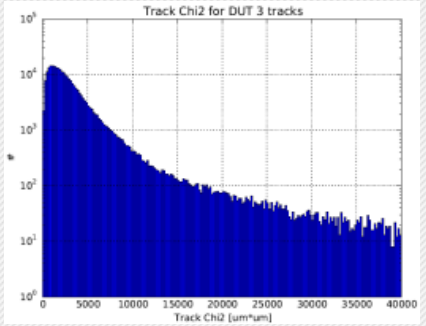


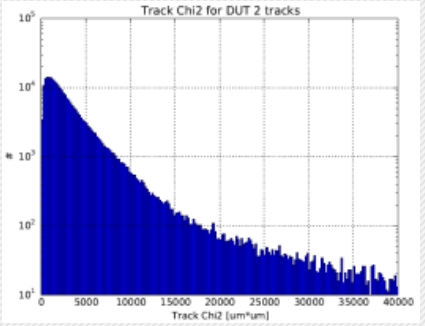
Straight line that fits best: X^2 minimization

Basically two lines of code:

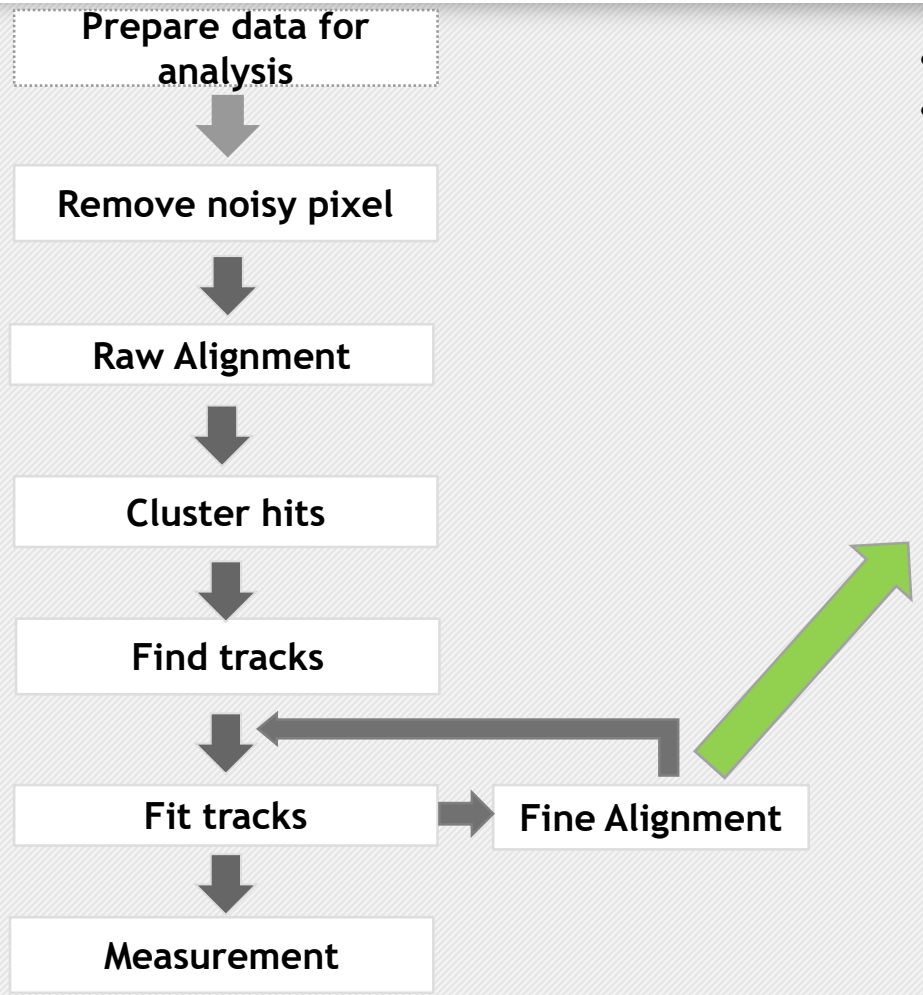
```
datamean = data.mean(axis=0)
offset, slope = datamean, np.linalg.svd(data - datamean)[2][0]
#http://stackoverflow.com/questions/2298390/fitting-a-line-in-3d
```

Track defined with x and y offset and slope (4 parameters).
X^2 calculated:



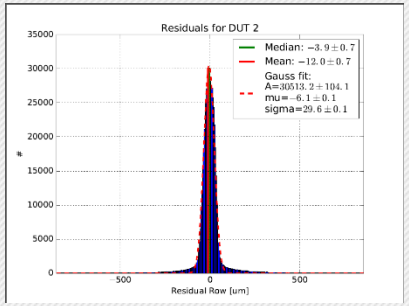


Analysis Workflow

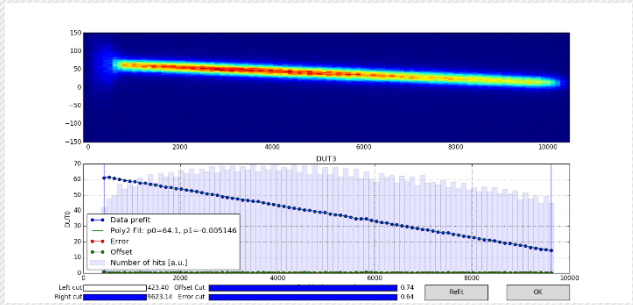


- Finds the relative positions and angles between planes
- Look at correlation between hit position and residual (difference between measured and predicted position)

Residual



Correlation



1D profile of the 2D distributions (4 in total), fit with straight line, extract geometrical parameters from fit.

Analysis Workflow

Prepare data for analysis

Remove noisy pixel

Raw Alignment

Cluster hits

Find tracks

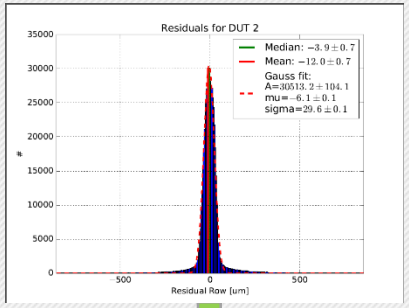
Fit tracks

Fine Alignment

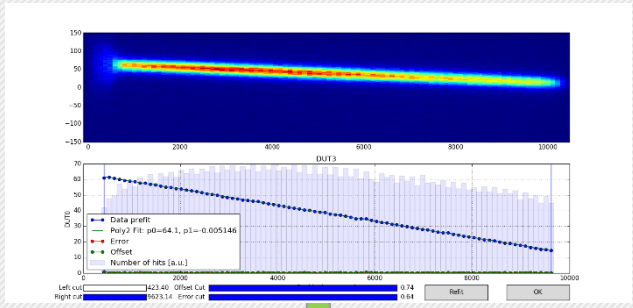
Measurement

- Finds the relative positions and angles between planes
- Look at correlation between hit position and residual (difference between measured and predicted position)

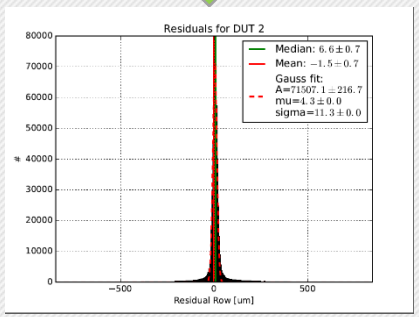
Residual

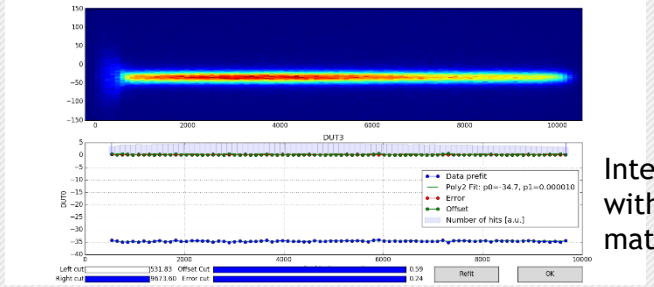


Correlation



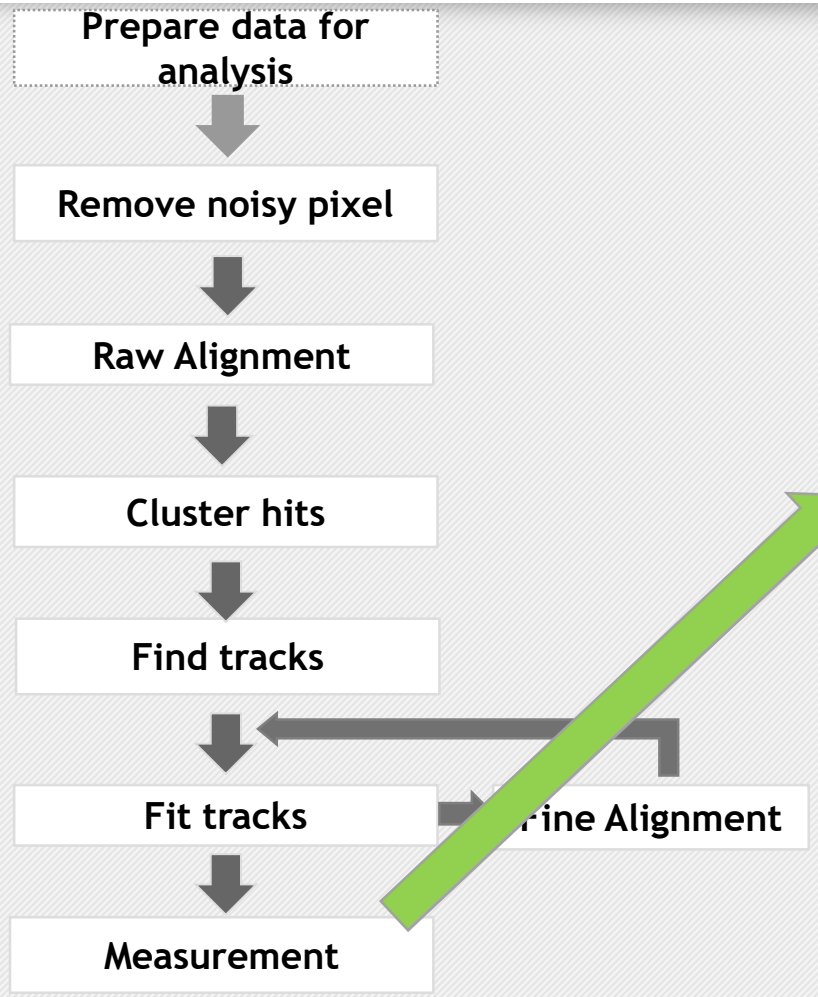
1D profile of the 2D distributions (4 in total), fit with straight line, extract geometrical parameters from fit.





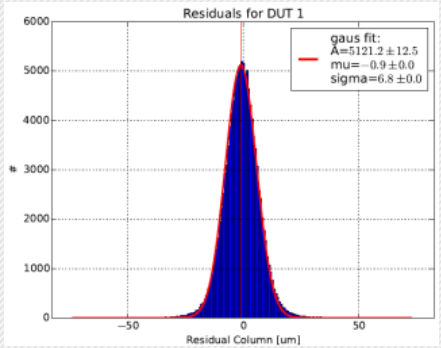
Interactively done with matplotlib.widgets

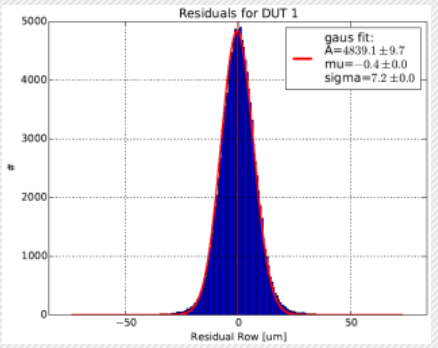
Analysis Workflow



Several ones:

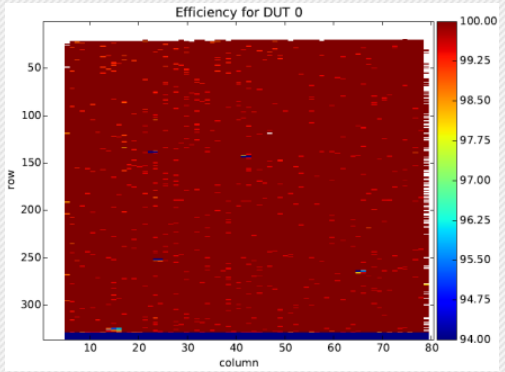
Residuals:





Telescope resolution:
7 µm for inner planes with EUTelescope (6 planes, 18x18 µm^2 pixels)

Efficiency:



Typical measurement for DUTs:
Number of recorded hits over number of reconstructed tracks.

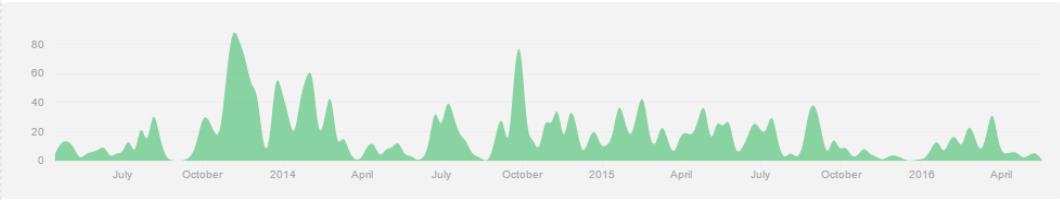
And others…

Applications

- Framework used to analyze data with brand new devices
  - Three main test beam facilities
    - CERN
    - DESY
    - ELSA (Bonn)
  - Different technologies investigated
    - 3D Silicon sensors
    - Diamond sensors
    - HV/HR-CMOS

All possible alternatives to the current Atlas pixel sensors

- Development and improvements ongoing…



Conclusions

- Analysis framework written with Python to analyze Physics data
  - Chip characterization
  - Test beam
- Easy to displace and use (Anaconda + Eclipse)
- Makes use of many Python packages
  - Numpy, SciPy, multiprocessing, cython,…
- Important applications for Silicon sensor R&D
  - Testing of novel technologies to replace/support the current detector

Conclusions

- Analysis framework written with Python to analyze Physics data
  - Chip characterization
  - Test beam
- Easy to displace and use (Anaconda + Eclipse)
- Makes use of many Python packages
  - Numpy, SciPy, multiprocessing, cython,…
- Important applications for Silicon sensor R&D
  - Testing of novel technologies to replace/support the current detector

Thanks!

Backup: the Zen of Python

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!

Tim Peters