

Test Beam with Python

1

Luigi Vigani - University of Oxford

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

2nd Developers@CERN Forum

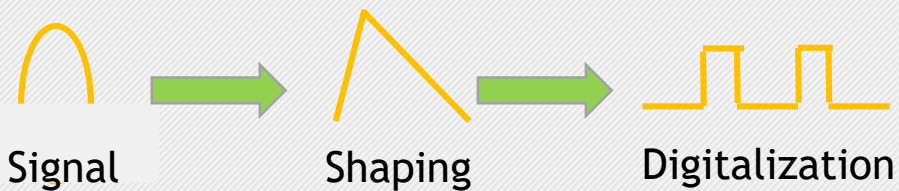
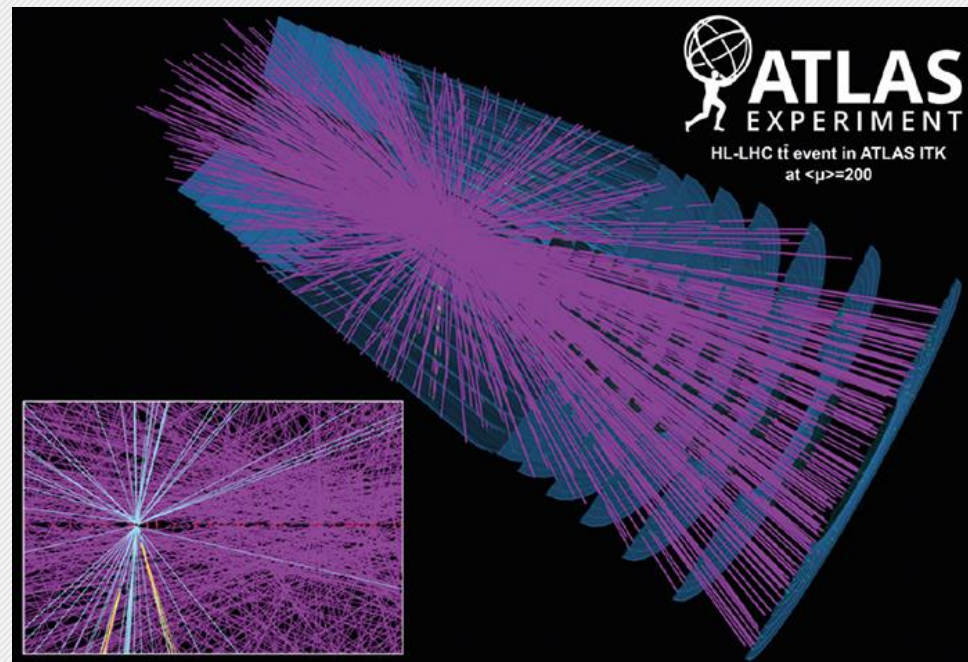
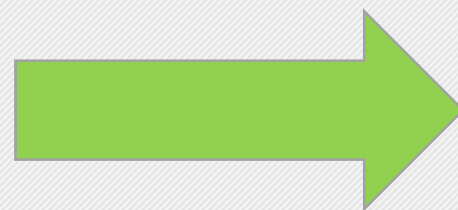
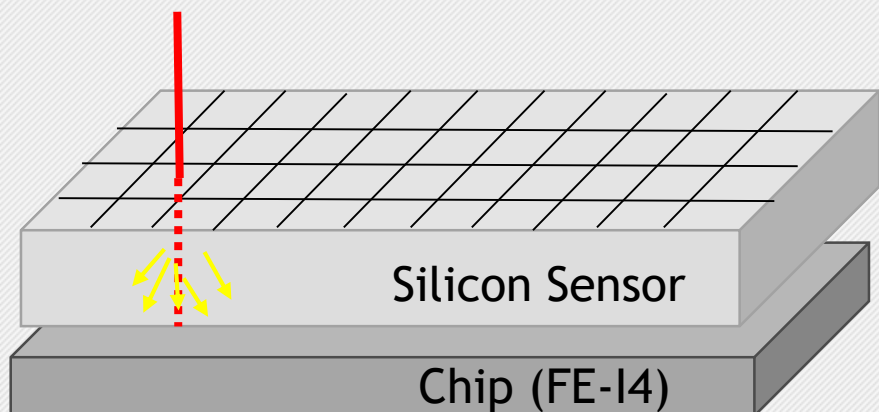
31st May 2016



The Context

2

Silicon Pixel Detectors



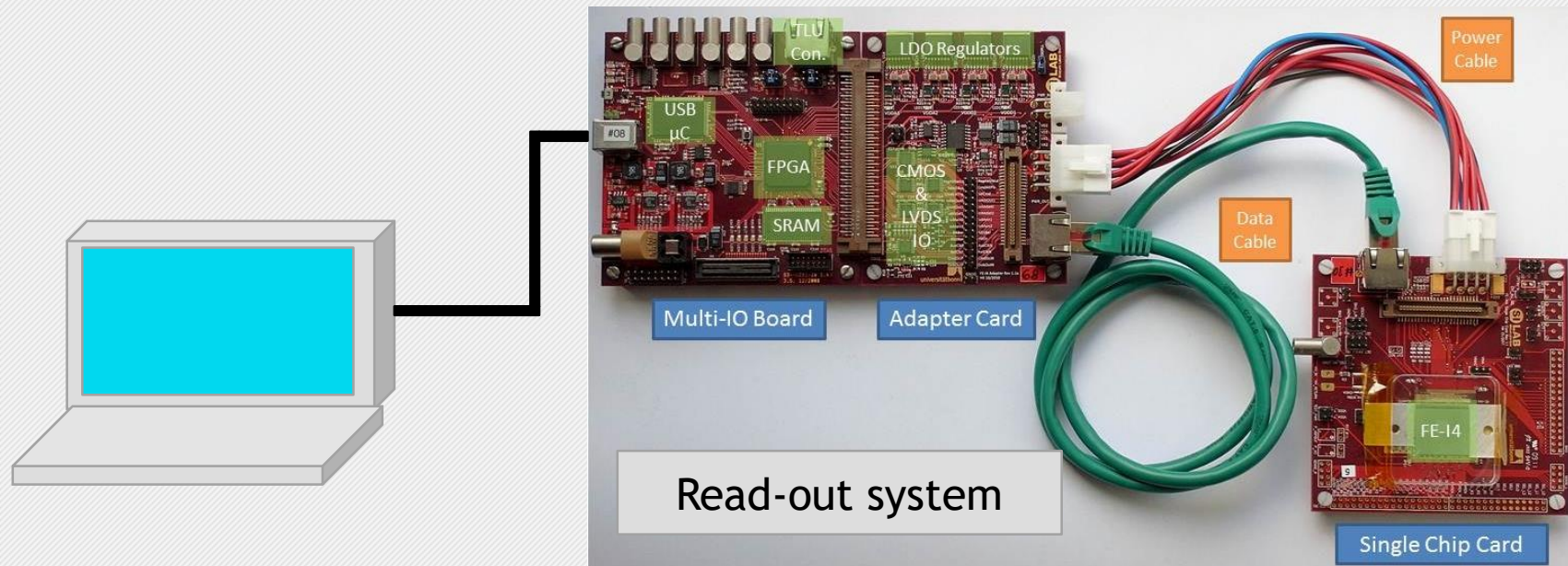
Atlas Tracker

Single Chip Read-out System

3

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

4

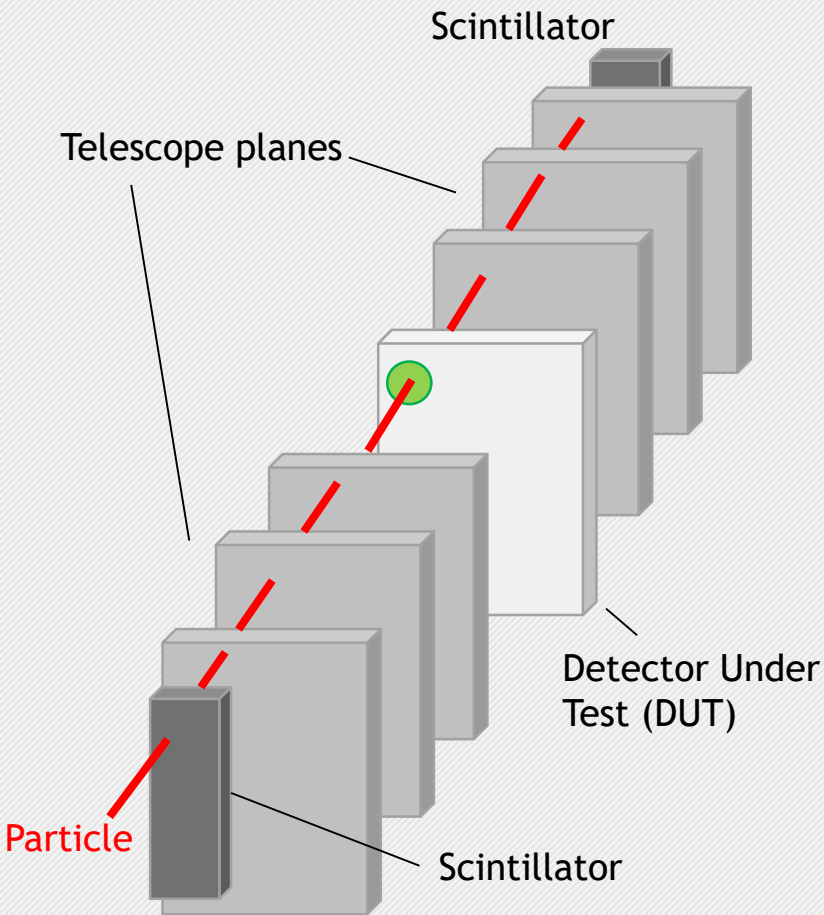
The screenshot shows the Eclipse IDE interface. On the left, the PyDev Package Explorer displays the project structure, including folders for analysis, calibration, scan, and test. The main editor window shows the code for `ExtTriggerScan` in `scan_ext_trigger.py`. The code includes imports for logging, numpy, and threading, and defines a class `ExtTriggerScan` that inherits from `Fei4RunBase`. The class has a `default_run_conf` dictionary with parameters like `trig_count`, `trig_latency`, and `trig_delay`. It also has a `configure` method that sets up the hardware configuration.

```
10 @logger(logging.INFO)
11 import logging
12 import numpy as np
13 import progressbar
14 from threading import Timer
15
16 @from pybar.analysis.analyze_raw_data import AnalyzeRawData
17 from pybar.fe14.register_utils import invert_pixel_mask, make_box_pixel_mask_from_col_row
18 from pybar.fe14_run_base import Fei4RunBase
19 from pybar.run_manager import RunManager
20
21 class ExtTriggerScan(Fei4RunBase):
22     """External trigger scan with FE-14
23     For use with external scintillator (user RX0), TLU (use R345), FE-14 HitOR (USBpix self-trigger).
24     Note:
25     Set up trigger in DUT configuration file (e.g. dut_configuration_mio.yaml).
26     """
27     default_run_conf = {
28         "trig_count": 0, # FE-14 trigger count, number of consecutive BCs, 0 means 16, from 0 to 15
29         "trig_latency": 232, # FE-14 trigger latency, in BCs, external scintillator / TLU / HitOR: 232, USBpix self-trigger: 220
30         "trig_delay": 4, # trigger delay, in BCs
31         "trig_rate_limit": 500, # artificially limiting the trigger rate, in BCs (25ns)
32         "col_span": (1, 80), # defining active column interval, 2-tuple, from 1 to 80
33         "row_span": (1, 336), # defining active row interval, 2-tuple, from 1 to 336
34         "overwrite_enable_mask": False, # if True, use col_span and row_span to define an active region regardless of the Enable pixel register. If False, use
35         "use_enable_mask_for_imon": True, # if True, apply inverted Enable pixel mask to Imon pixel mask
36         "no_data_timeout": 100, # no data timeout after which the scan will be aborted, in seconds
37         "scan_timeout": 60, # timeout for scan after which the scan will be stopped, in seconds
38         "max_triggers": 1000000, # maximum triggers after which the scan will be stopped, in seconds
39         "enable_sdc": False, # if True, enables SDC (use RX2)
40         "reset_rx_on_error": False # long scans have a high probability for ESD related data transmission errors: recover and continue here
41     }
42
43     def configure(self):
44         commands = []
45         commands.extend(self.register.get_commands("ConfMode"))
46         # Enable
47         enable_pixel_mask = make_box_pixel_mask_from_col_row(column=self.col_span, row=self.row_span)
48         if not self.overwrite_enable_mask:
49             enable_pixel_mask = np.logical_and(enable_pixel_mask, self.register.get_pixel_register_value("Enable"))
50         self.register.set_pixel_register_value("Enable", enable_pixel_mask)
51         commands.extend(self.register.get_commands("WriteFrontEnd", same_mask_for_all_do=False, name="Enable"))
```

- 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://vitable.org/>) + BLOSC compression.

Test Beam

5



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

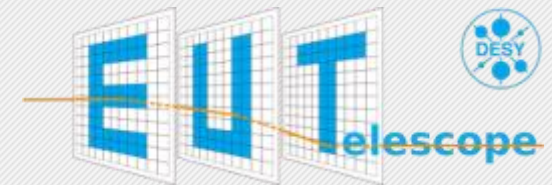
Particle tracks must be related to hits on DUT:

- Triggers from scintillators distributed to all planes

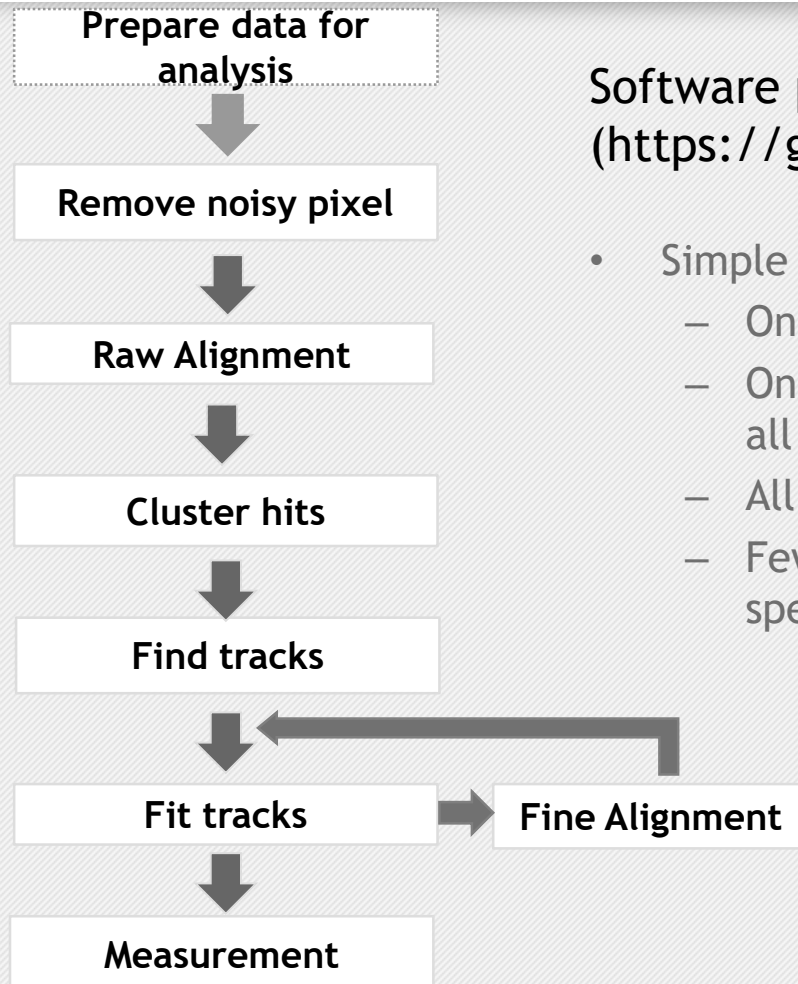
Two main options:

- 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 internal
                  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

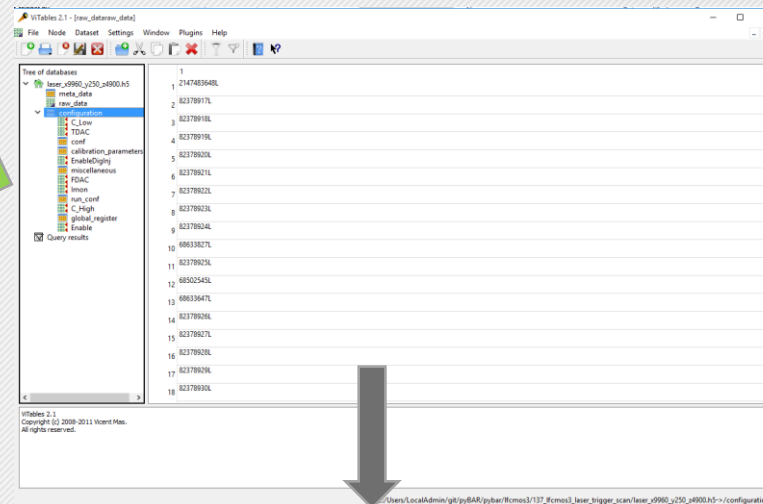
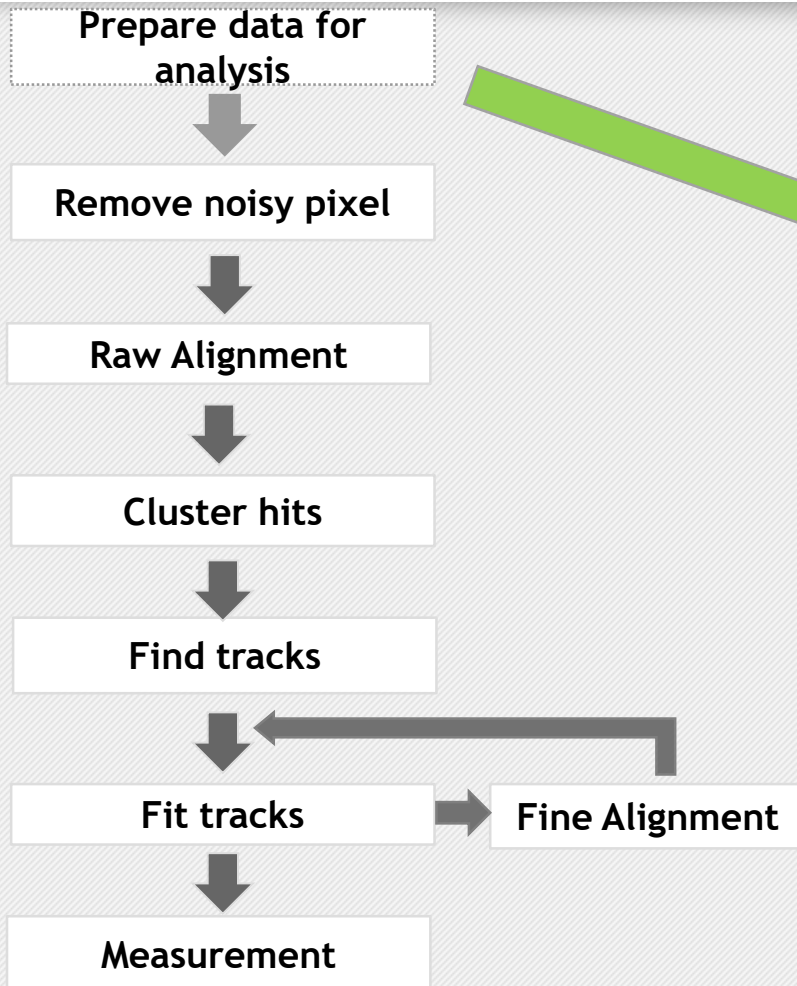
    # Correlate the row / column of each DUT
    tba.correlate_hits(data_files,
                      alignment_file=output_folder + r'/Correlation.h5',
                      fraction=1,
                      event_range=0)

    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

6



Raw data extracted from each chip...

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

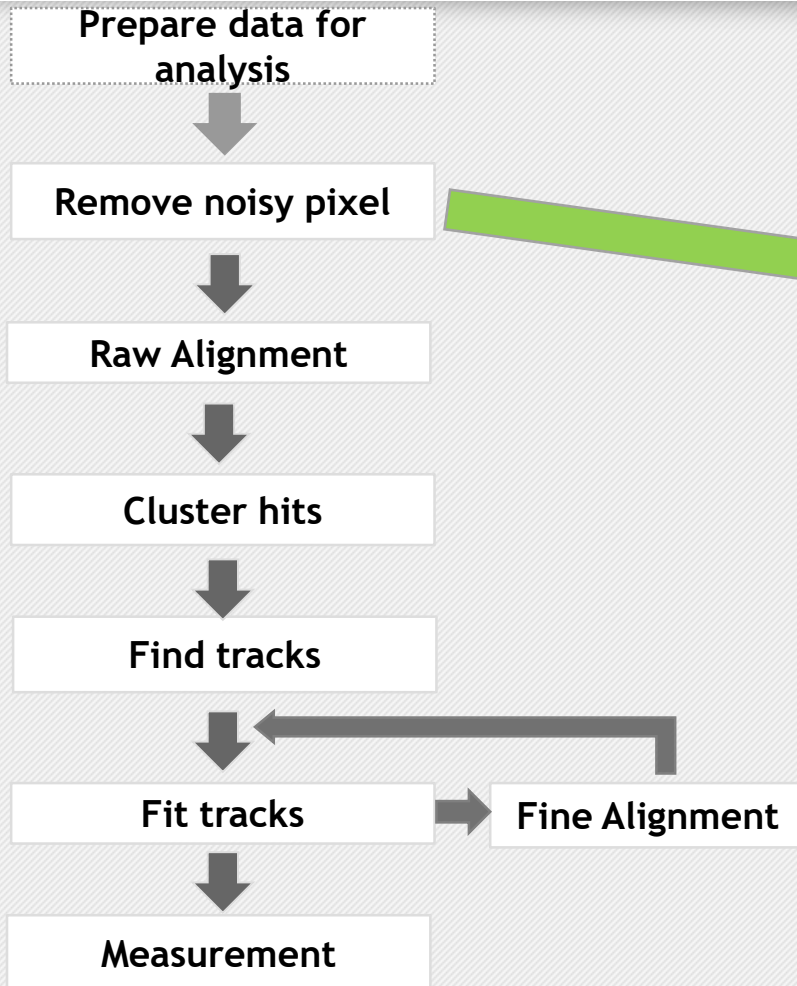
Then converted with pyTables

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

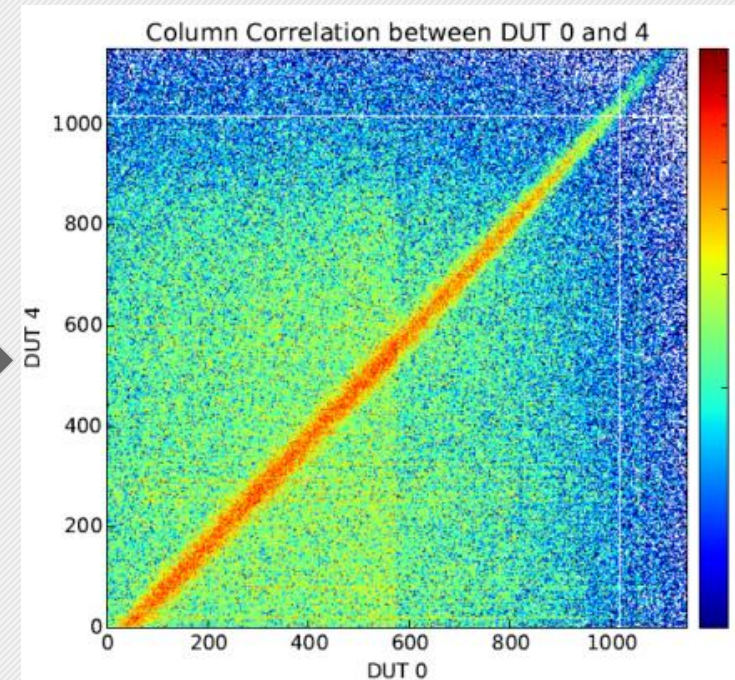
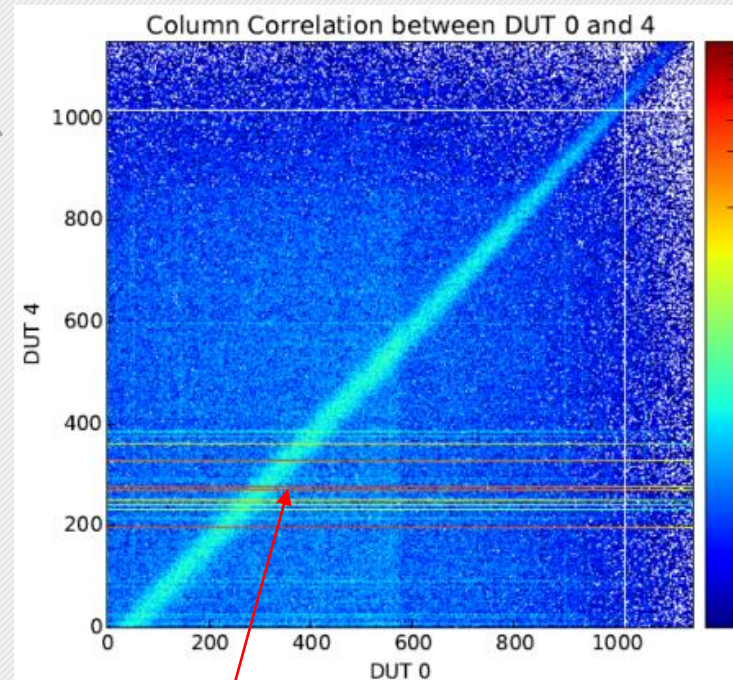
- Hits from all planes as pixel addresses (row and column)
- Noise

Analysis Workflow

6



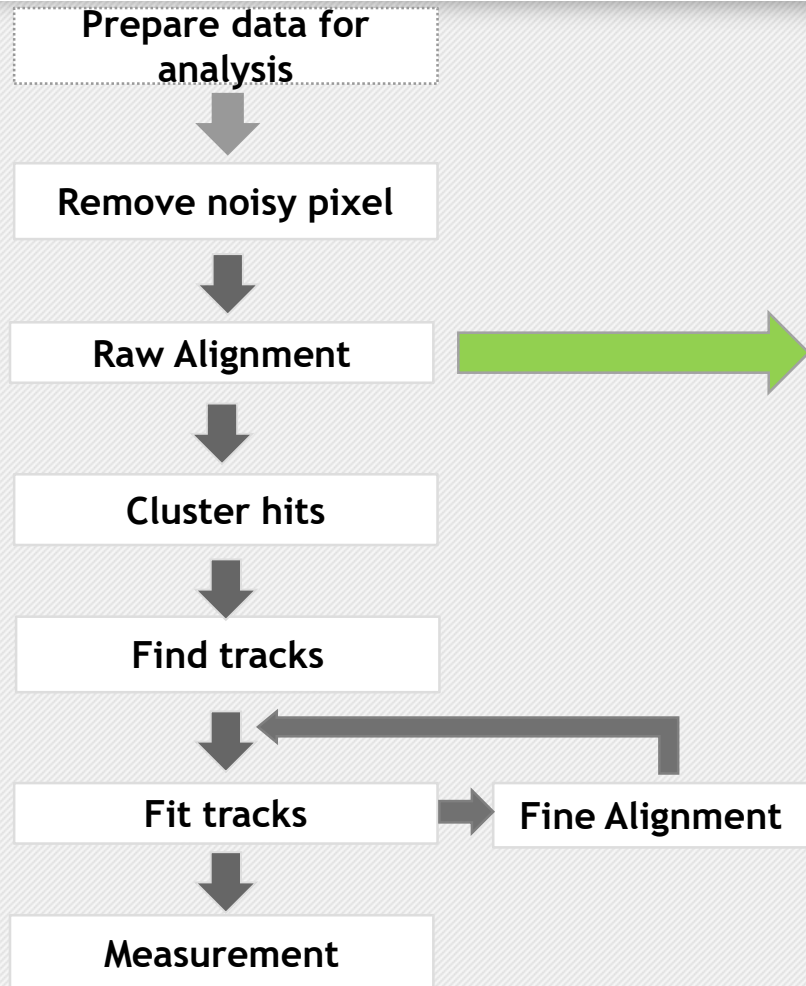
Plots produced with matplotlib



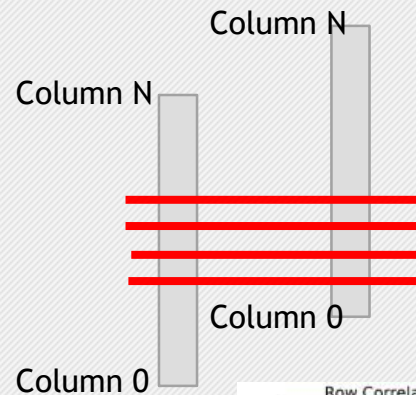
This column on DUT4 has more hits than average wrt DUT0...
Noise! Remove noisy pixels from analysis.

Analysis Workflow

6

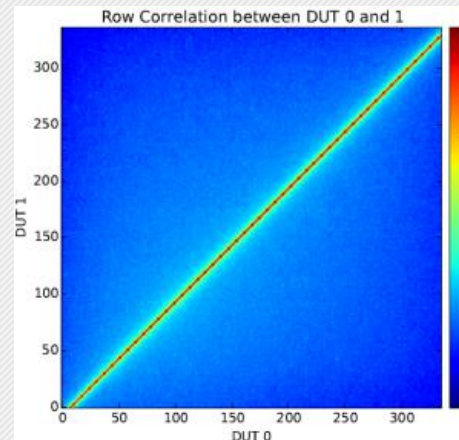


Looking at correlation between column (row) and column (row) between each plane and plane 0 (reference plane)

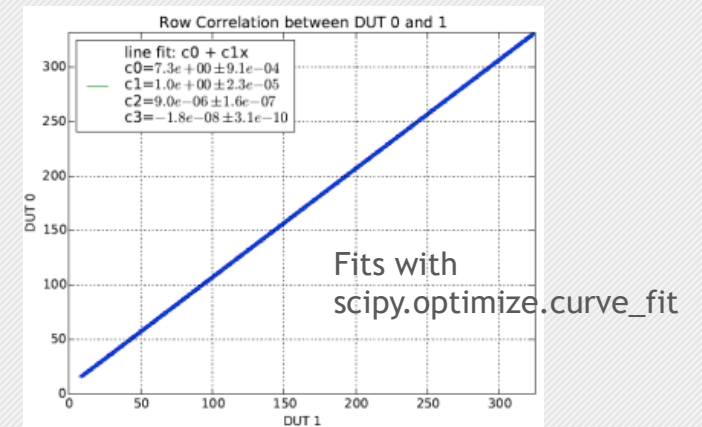


Displacement between planes

Offset between columns/rows

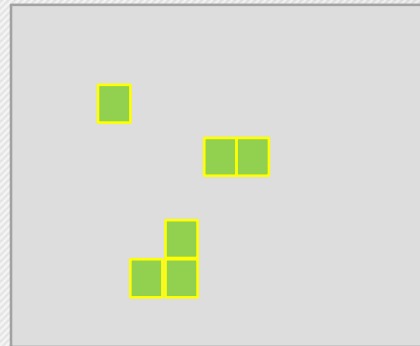
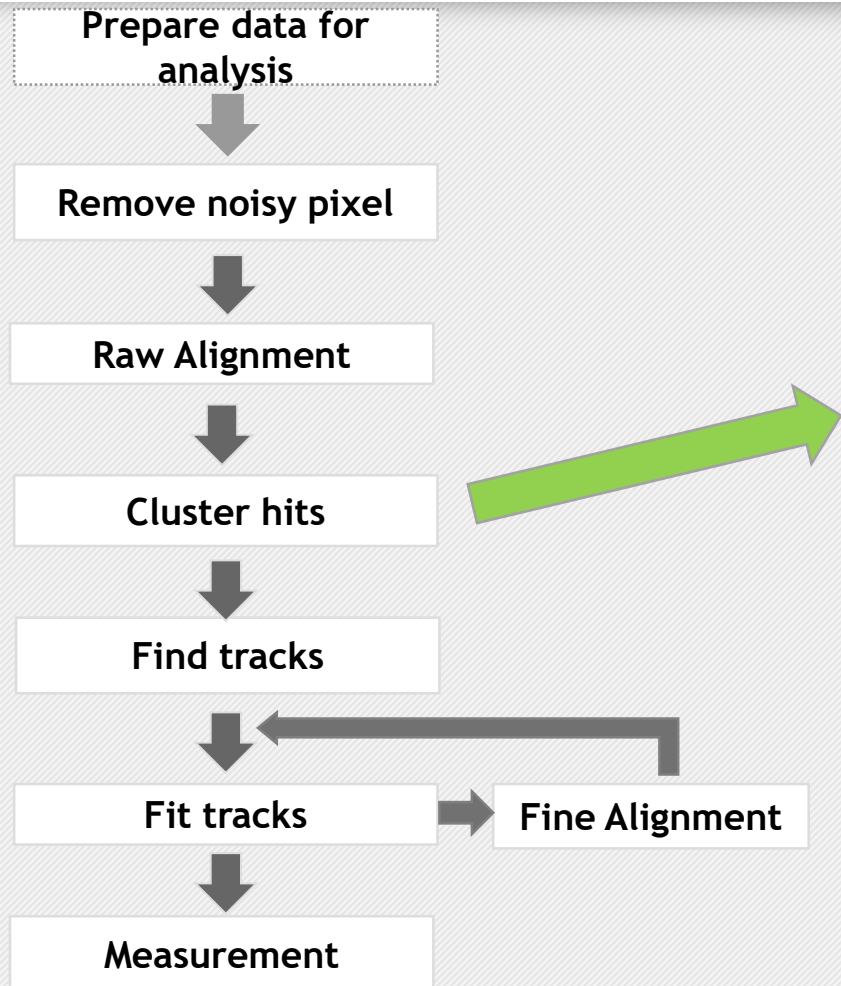


Average and fit



Analysis Workflow

6



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:

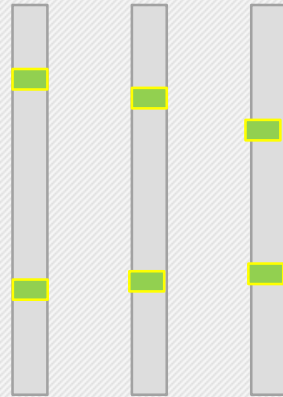
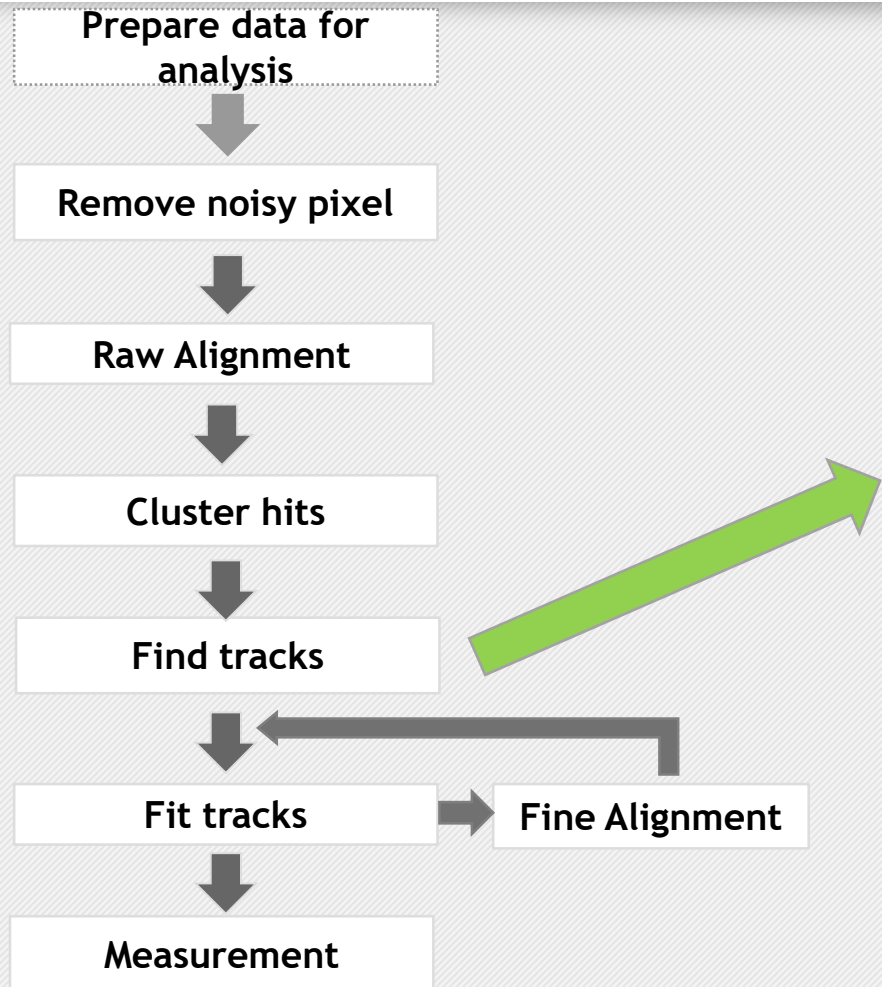
- Up till 100 times faster
- 10 times less code than cython
- More flexible

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

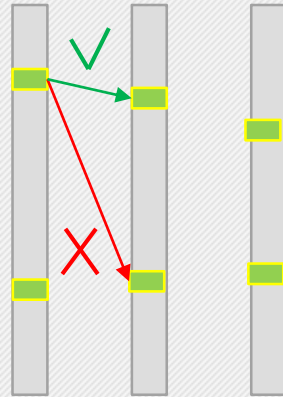
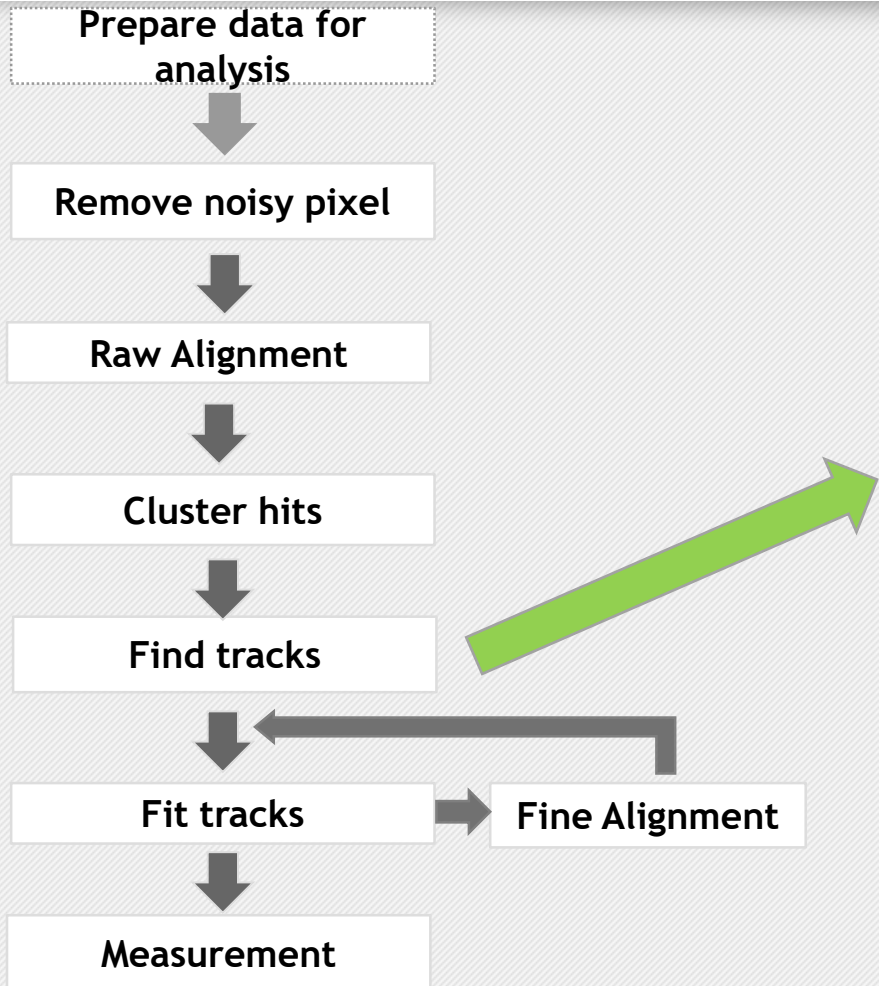
6



Multiple tracks can happen at the same time

Analysis Workflow

6

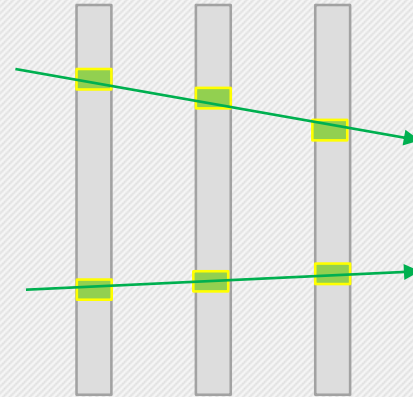
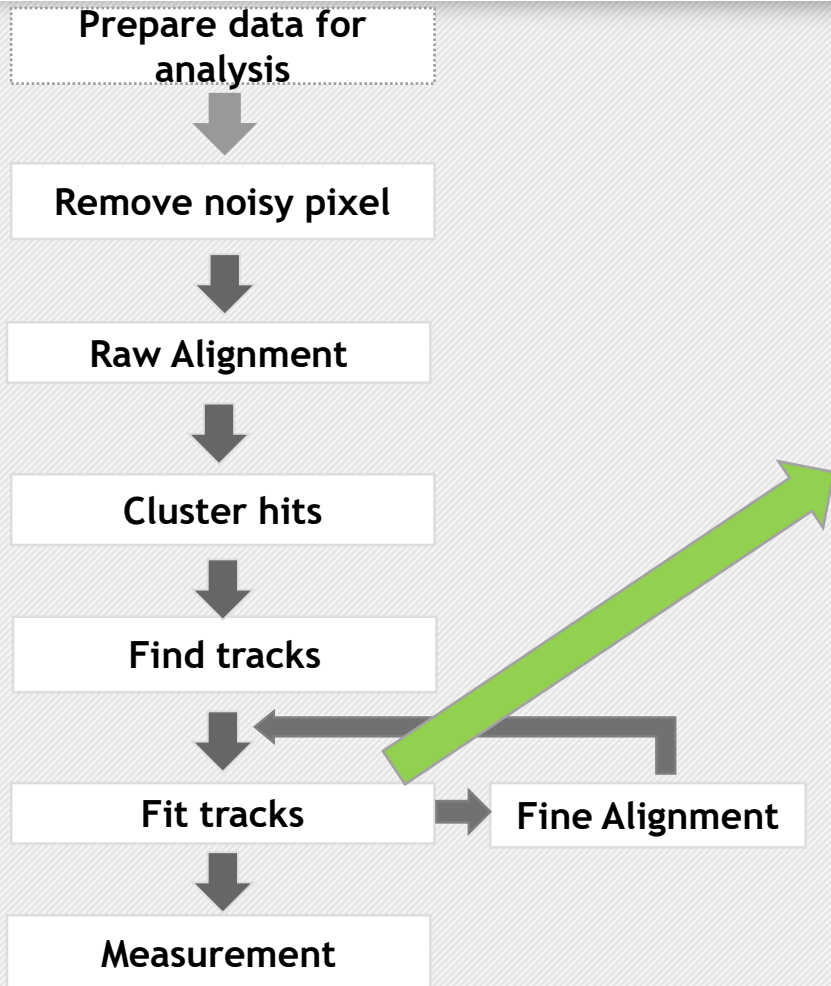


Multiple tracks can happen at the same time

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

Analysis Workflow

6

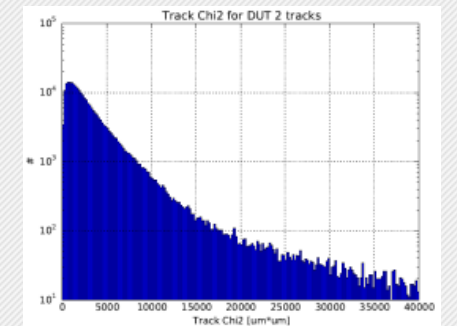
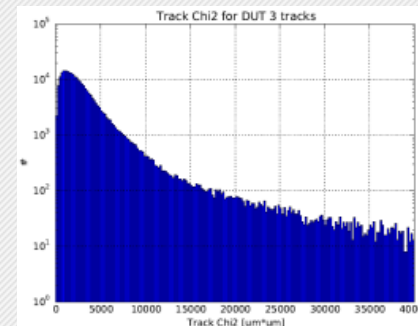


Straight line that fits best: X^2 minimization

Basically two lines of code:

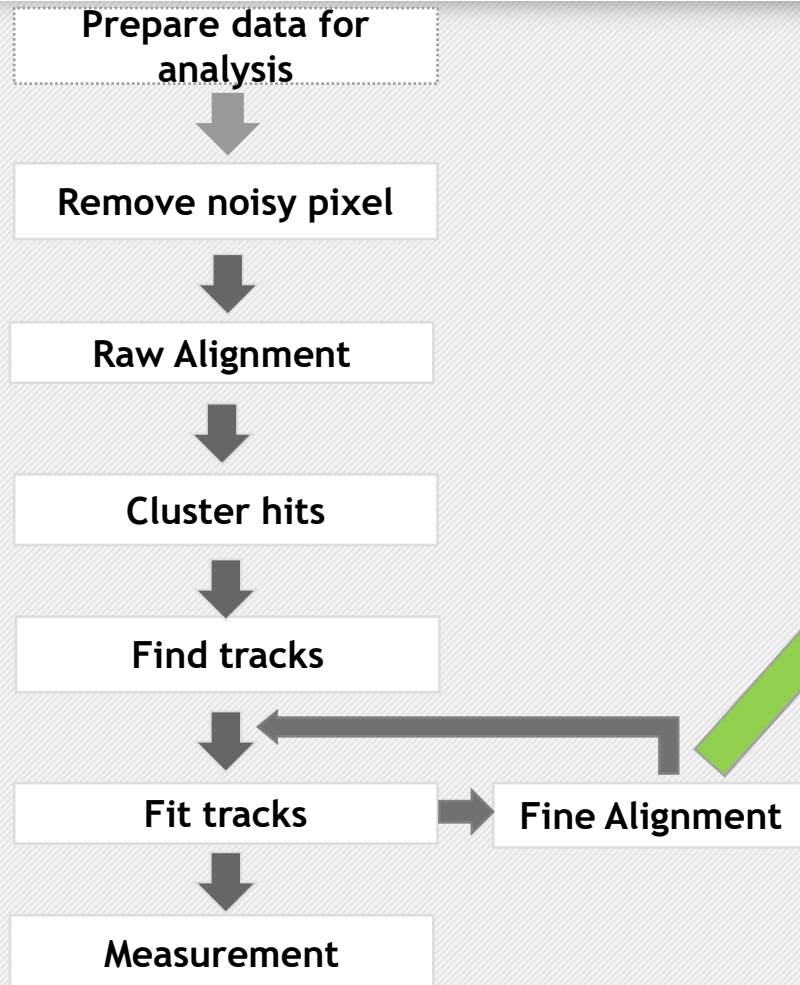
```
datamean = data.mean(axis=0)
offset, slope = datamean, np.linalg.svd(data - datamean)[2][0]
#http://stackoverflow.com/questions/2298390/fitting-a-line-in-3d
```

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



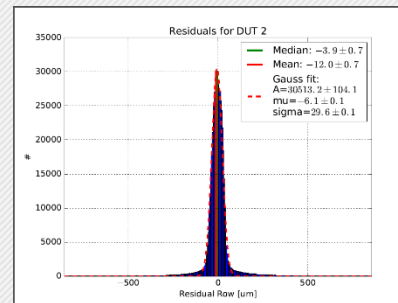
Analysis Workflow

6

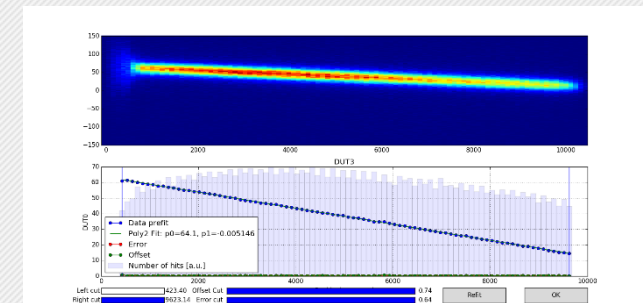


- Finds the relative positions and angles between planes
- Look at correlation between hit position and residual (difference between measured and predicted position)

Residual



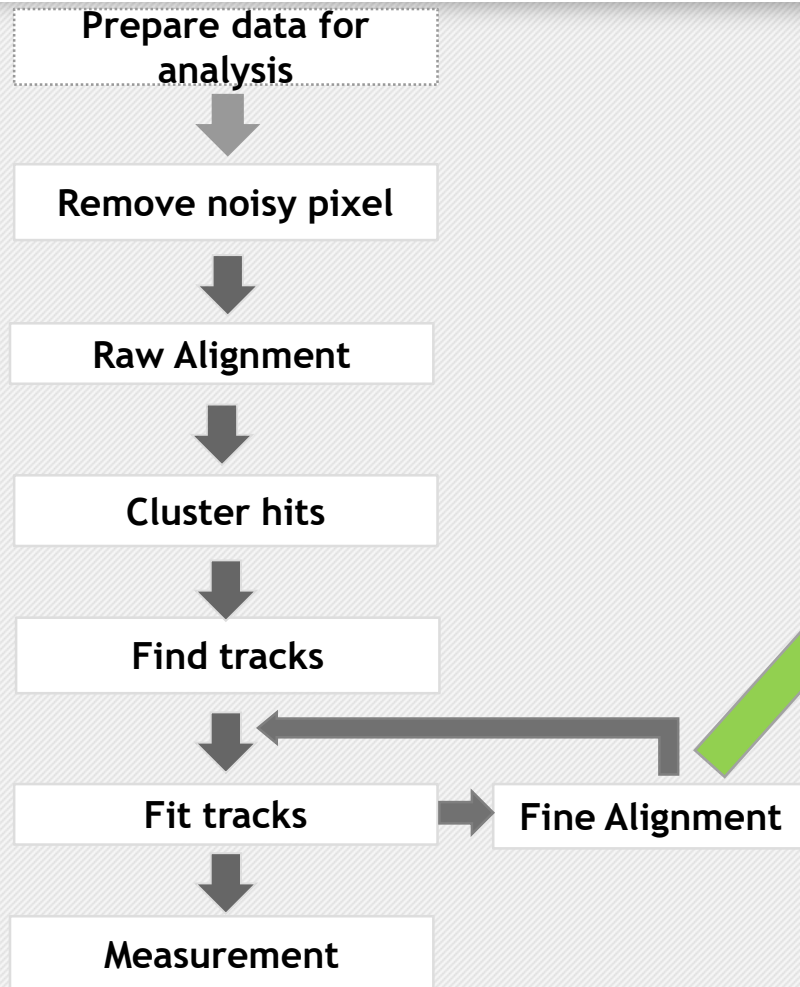
Correlation



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

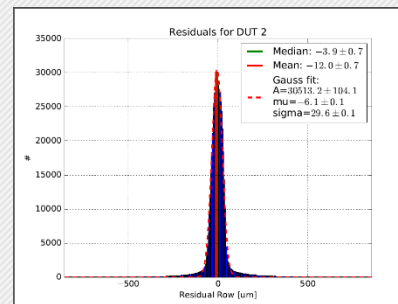
Analysis Workflow

6

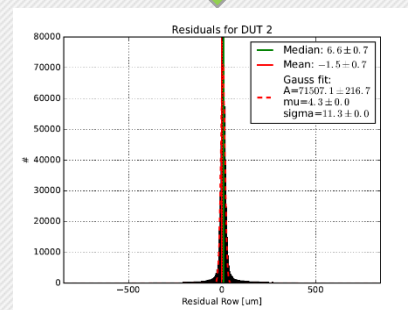


- Finds the relative positions and angles between planes
- Look at correlation between hit position and residual (difference between measured and predicted position)

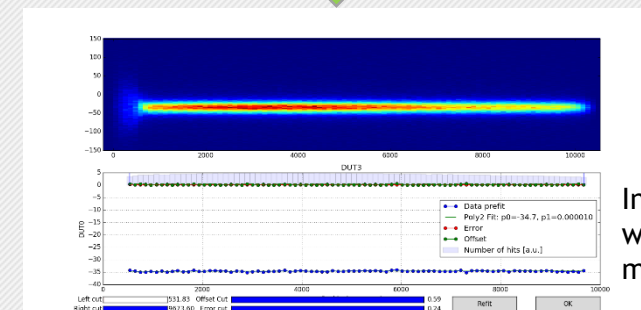
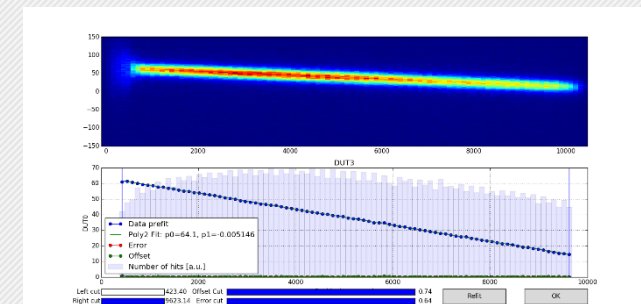
Residual



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



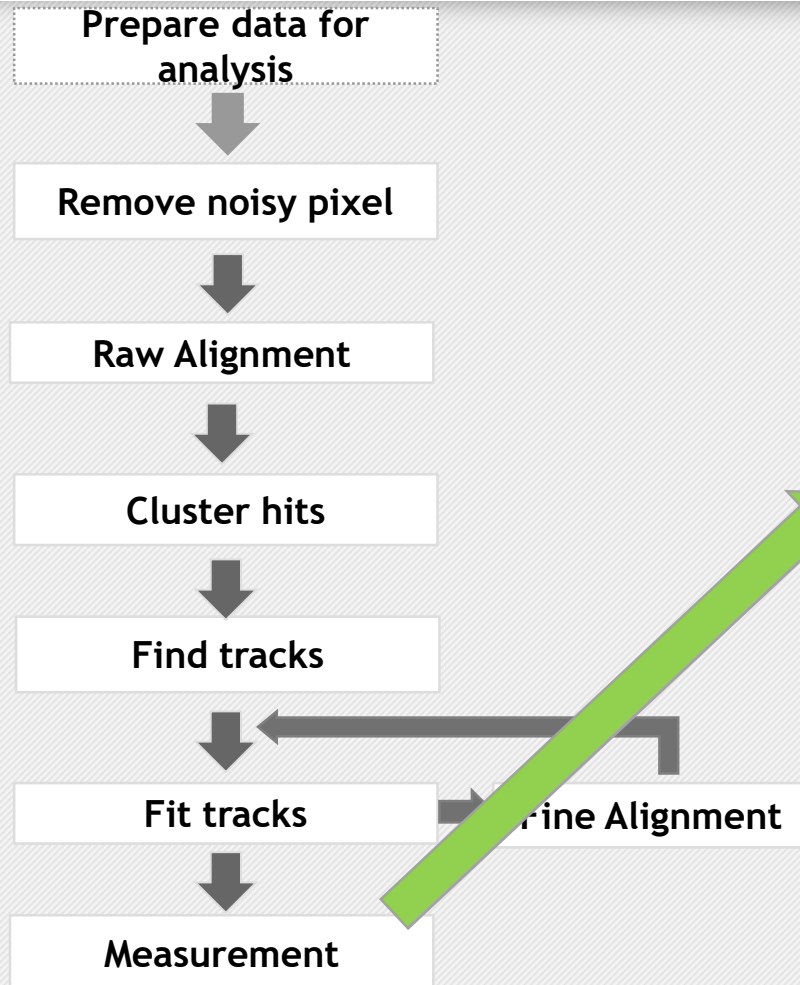
Correlation



Interactively done with matplotlib.widgets

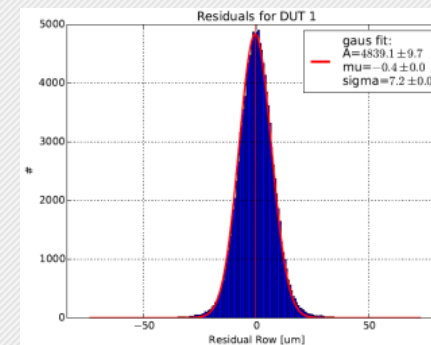
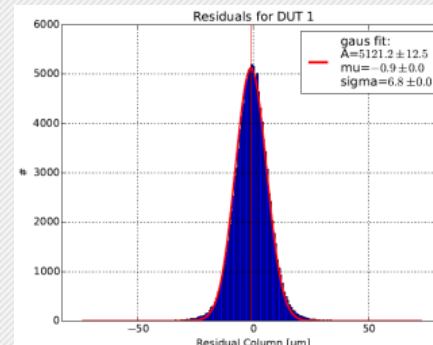
Analysis Workflow

6



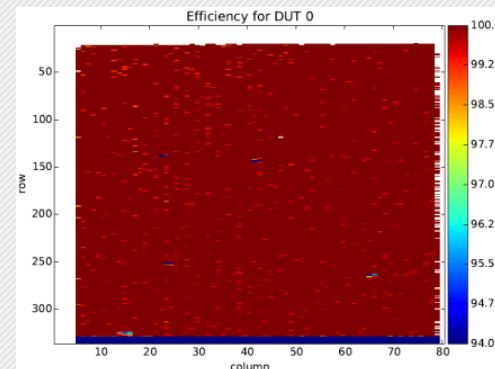
Several ones:

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.

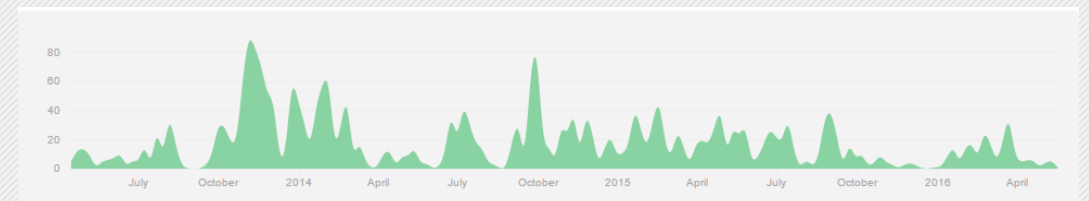
And others...

Applications

7

- 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
- Development and improvements ongoing...

All possible alternatives to the current Atlas pixel sensors



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

9

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