Synchronized analysis of testbeam data with the Judith software

9th International "Hiroshima" Symposium on the Development and Application of Semiconductor Tracking Detectors

Garrin McGoldrick, Andrej Gorišek

University of Toronto, Jožef Stefan Institute

September 3, 2013

Aim

To produce an easy-to-use software framework for analysis of data acquired at testbeams with heterogeneous reference detector and DUT that would make it possible to combine and analyze data streams without sophisticated arbitrating of triggering.

Introduction

- Judith is a software package which performs synchronized data analysis of several (pixelated) particle detectors.
- The following is an overview of Judith's features:
  1. Conversion of raw data formats into a ROOT n-tuple format.
  2. Robust synchronization of simultaneously triggered data streams.
  3. Production of clusters and tracks in a set of pixelated detectors.
  4. Alignment of detector planes.
  5. Analysis of the resulting data.
- Judith has been used to analyze testbeam data of the Diamond Beam Montior (DBM).
- The testbeam was carried out at DESY in a beamline of 5 GeV electrons.
- The Kartel telescope was used as a reference detector.

The DESY Beamline

- Electrons had an energy of approximately 5 GeV.
- The reference sensors observed a fluence approximately 5 particles in a time of 100 μs.
- Multiple scattering caused a deviation of tracks of up to few pixels at the last plane of the reference telescope.
- These are challenging conditions in which to reconstruct events.

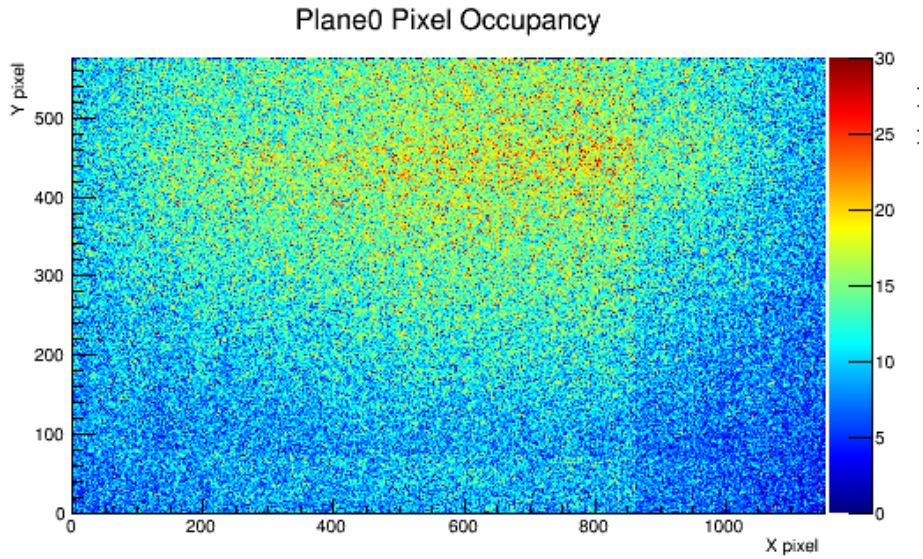
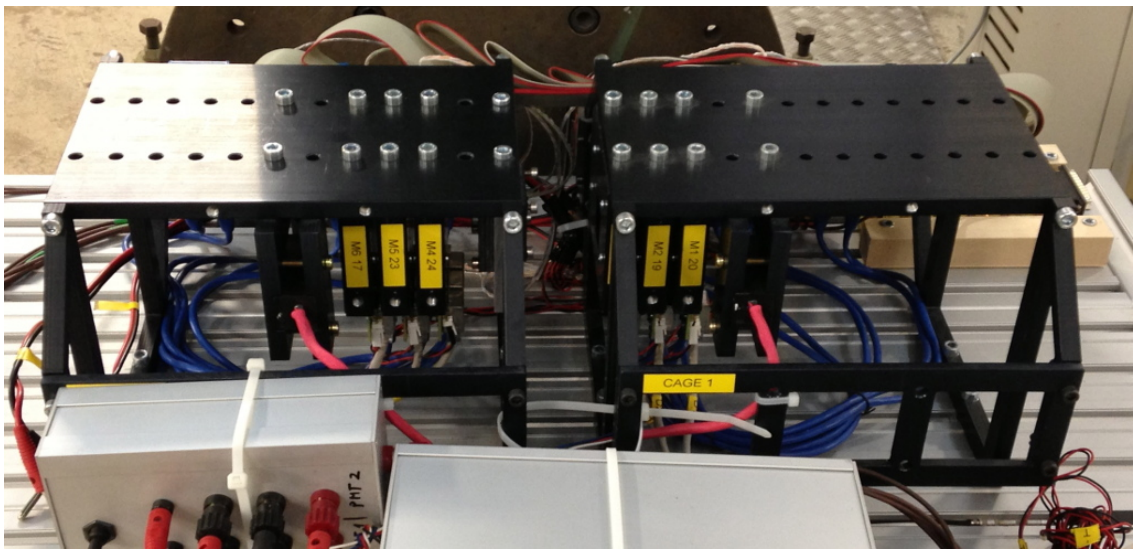


Figure : DESY testbeam profile as seen in a reference plane.

The Kartel Telescope

- The Kartel telescope was used as a reference detector for reconstructing particle tracks.
- The telescope is comprised of 6 Mimosa 26 sensors, each with an area of 20.7 mm x 10.3 mm.
- They are instrumented by pixels with a pitch of 18 μm.
- The telescope is triggered by the coincidence of two scintillators.
- The 6 sensors are synchronized and work in a rolling shutter mode.
- The data is read out in 100 μs frames.

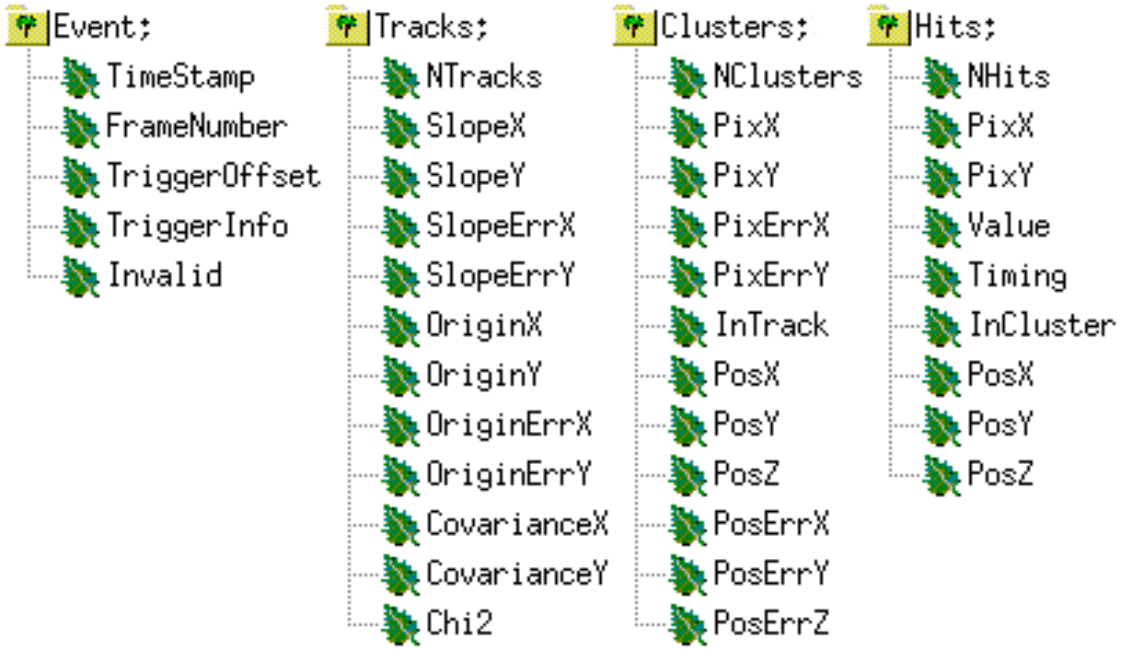


DBM with RCE Readout

- Two devices under test (DUTs) were used: one silicon, and one diamond particle detector based on the FEI4 readout chip.
- The diamond device is a prototype module for the Diamond Beam Monitor (DBM), a new detector which is to be installed in the ATLAS experiment.
- The silicon sensor was used as an "anchor" plane for reference in efficiency studies.
- The FEI4 chip is comprised of 26 880 pixels with a pitch of 250 μm x 50 μm covering an area of 20.0 mm x 16.8 mm.
- It has a read out window of 25 ns which was triggered by the Kartel telescope's scintillator.
- It was read out with the RCE system.

Judith Data Storage

- Judith stores data in ROOT n-tuples which can easily be analyzed in a standalone ROOT session.
- Spatial information is provided in "pixel" and global coordinates as to retain the original data while encoding the detector geometry.



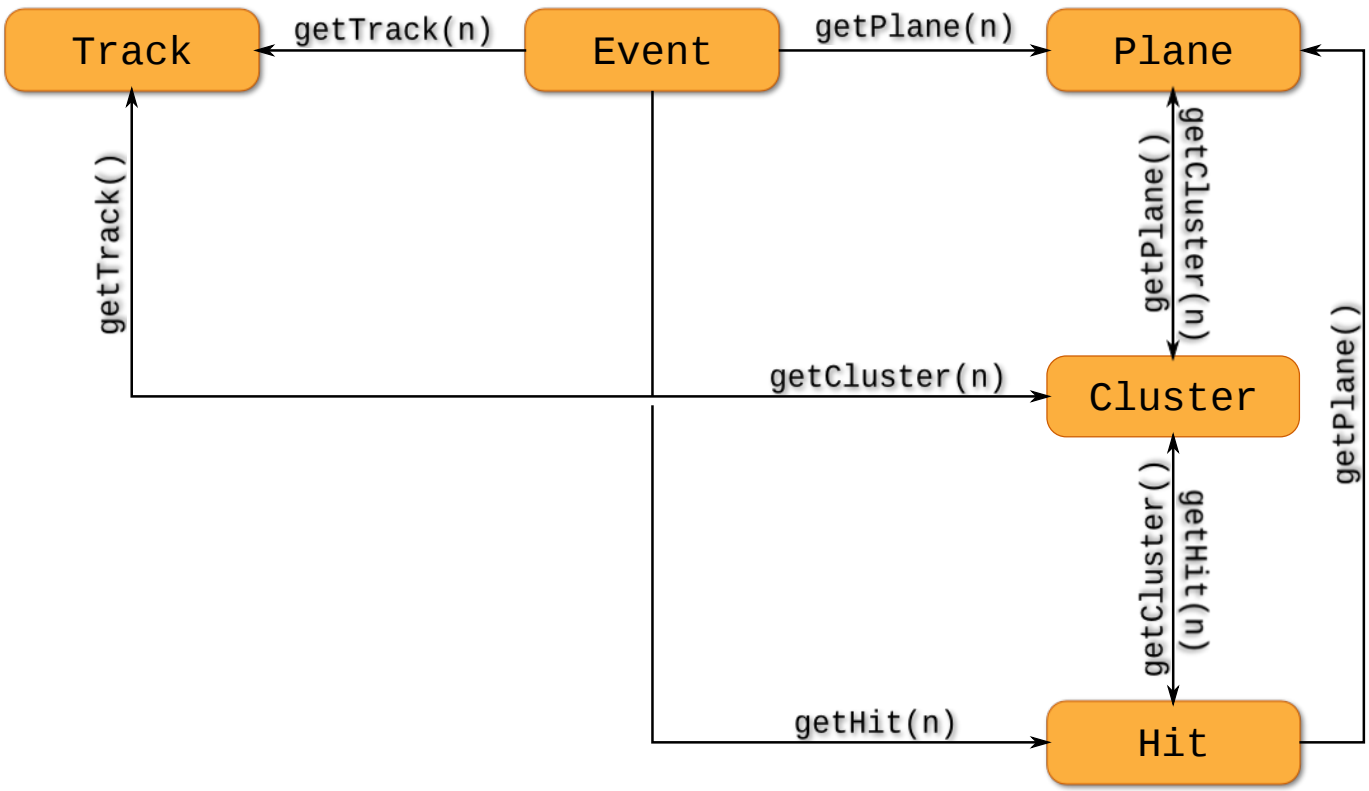
Conversion From Other Formats

- The data storage I/O classes are grouped in a module which can be used independently of the rest of the software.
- Converting from any other format is outlined in the following snippet of example code.

```
const unsigned int treeMask = Storage::Flags::TRACKS | Storage::Flags::CLUSTERS;
Storage::StorageIO storage(name, Storage::OUTPUT, numPlanes, treeMask);

// Loop over events in the input
for (/*...*/) {
  Storage::Event storageEvent(numPlanes);

  // Loop over sensor planes
  for (unsigned int nplane = 0; nplane < numPlanes; nplane++) {
    for (/*...*/) {
      Storage::Hit* hit = storageEvent->newHit(nplane);
      hit->setPix(x, y);
      /* Set additonal hit parameters here */
    }
  }

  /* Set additional event parameters here */
  storage->writeEvent(storageEvent);
}
```

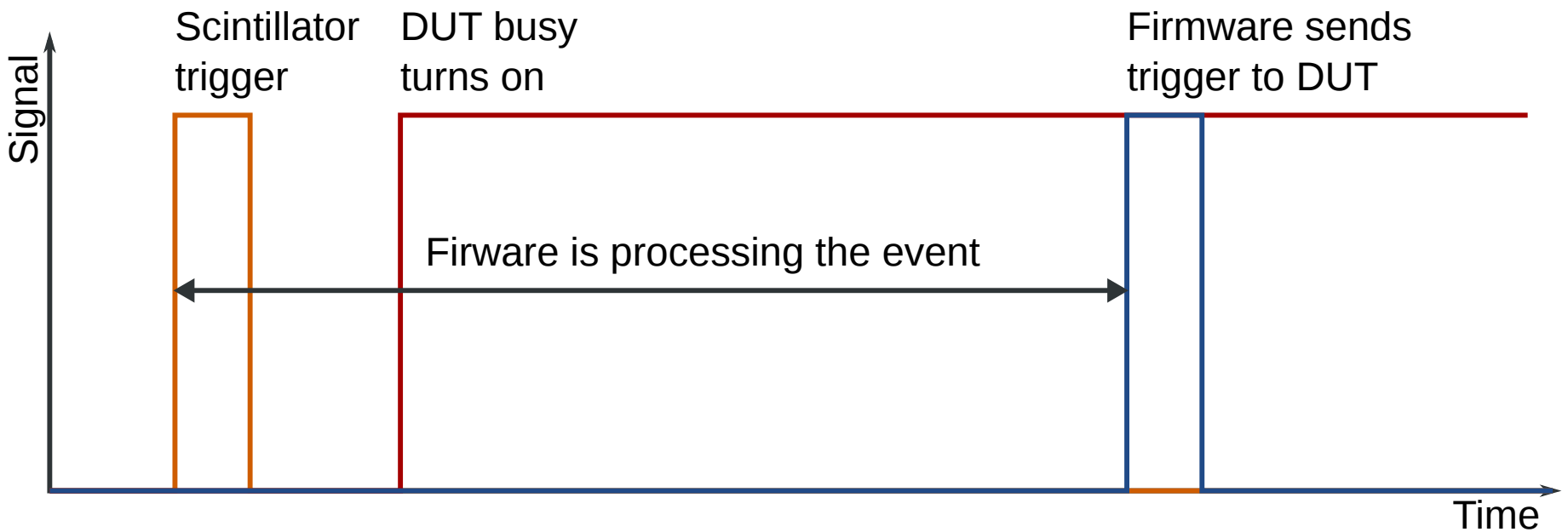
Data in Software

- Once an event has been read into memory from the storage, it can be accessed as follows:



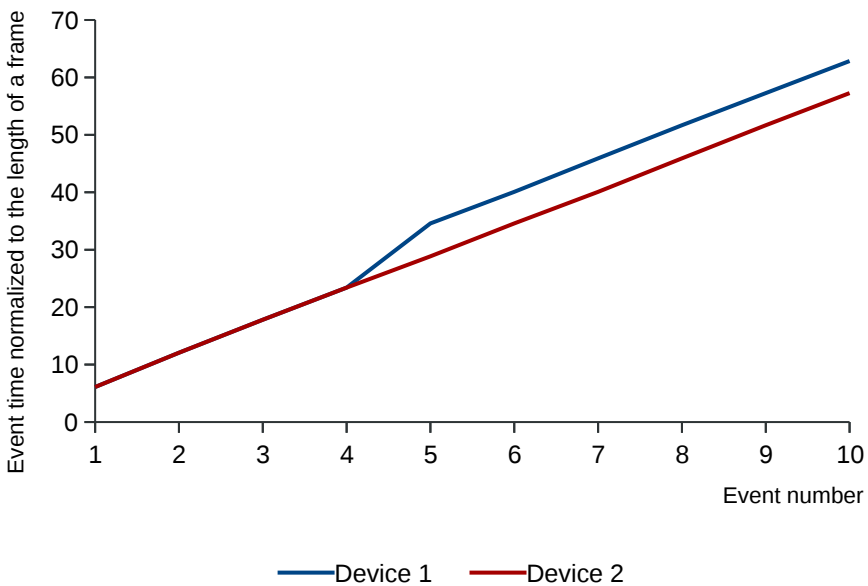
Desynchronization Problem

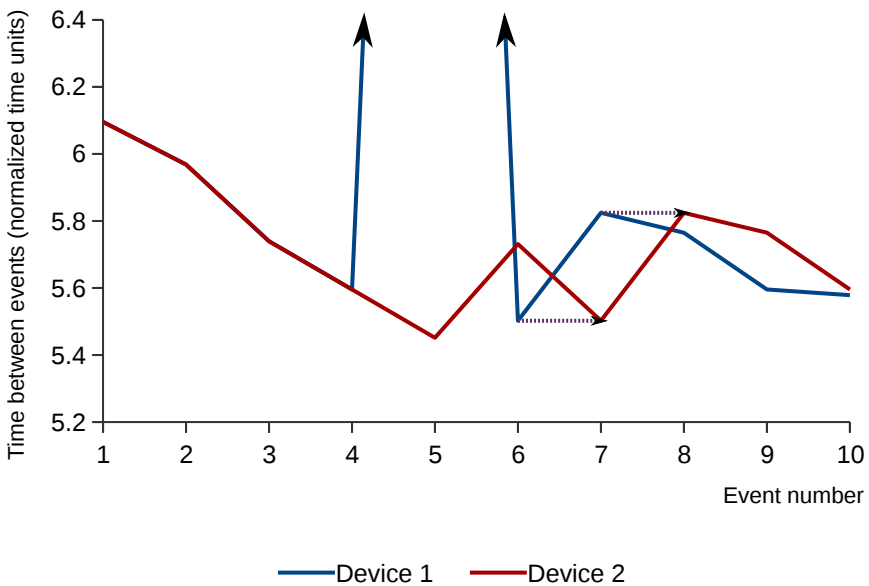
- Desynchronizations occur when the DUT or reference device fails to record a triggered event.
- Such events occurred in the DESY testbeam when the DUT's busy flag went up while the reference was processing the scintillator trigger.
- Judith can handle such desynchronizations without the need for two-way (handshake) communication between the DUT and reference device.



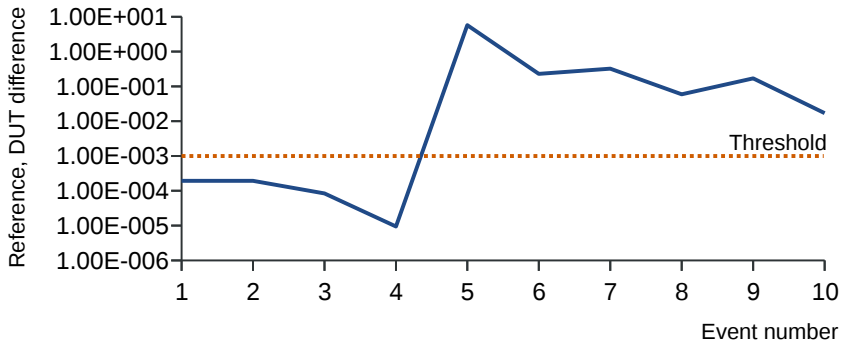
Identifying Desynchronizations

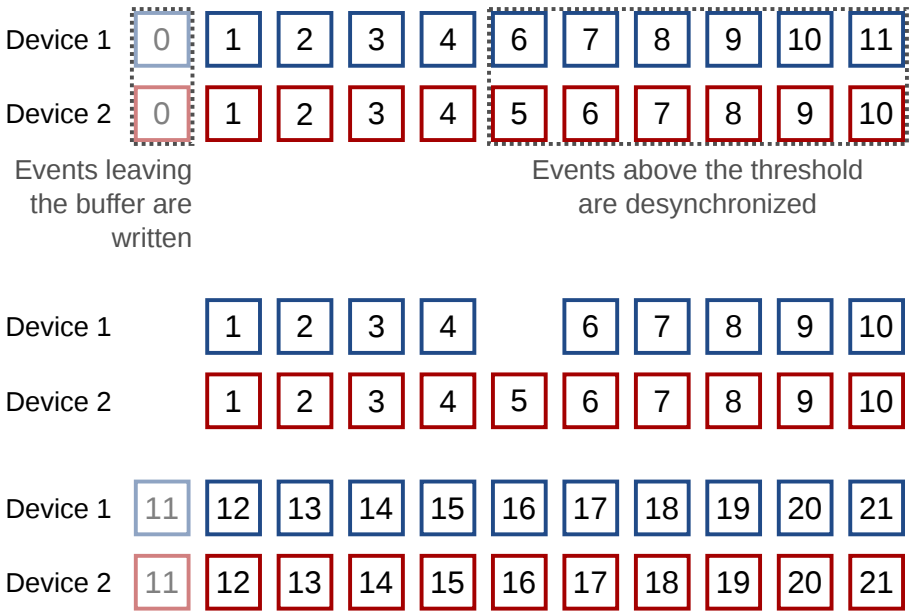
- Judith includes a fast robust synchronization routine.
- The DUT and reference devices need to output a time stamp with a finer resolution than the time between consecutive triggers.
- The time difference between consecutive frames is used to locate desynchronizations.





Fixing Desynchronizations

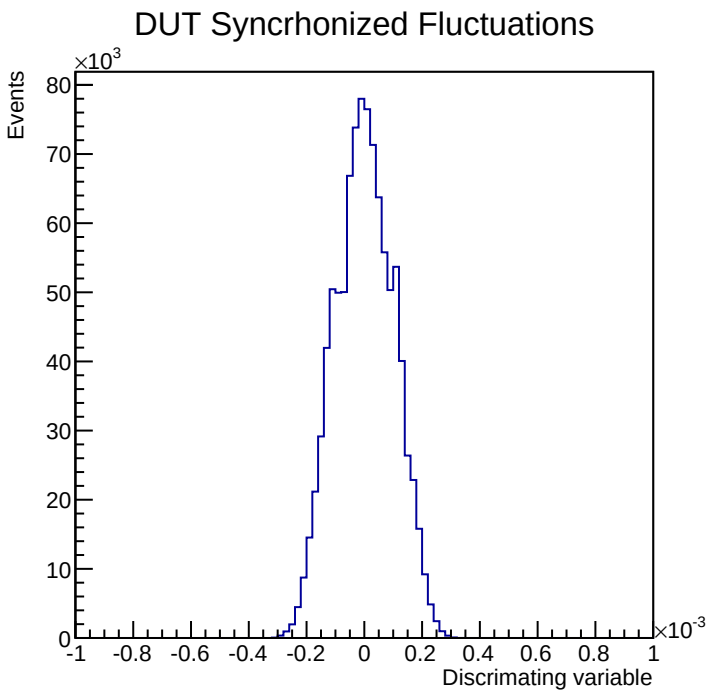


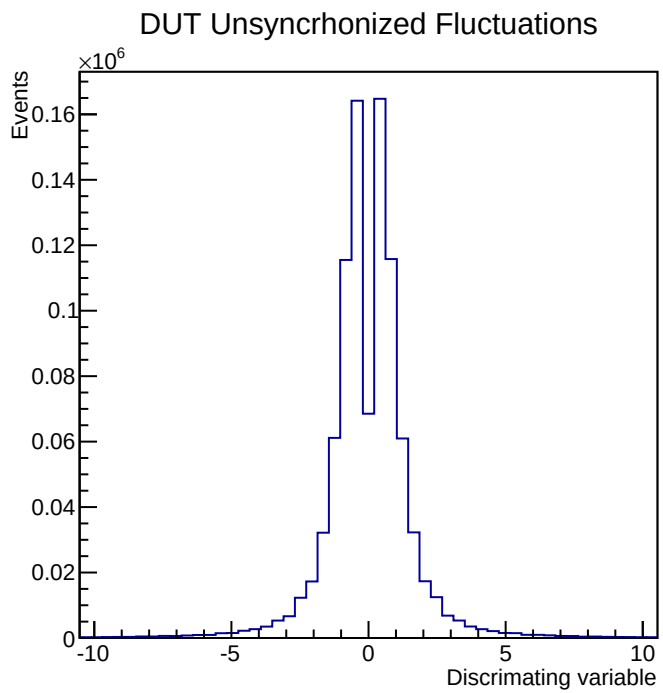


- A circular buffer is employed.
- The buffer compares the reference and DUT frame time changes.
- Those events above a threshold are desynchronized.
- The buffer is re-aligned once it is filled with desynchronized events.
- The buffer is cleared and event writing resumes once the next buffer is re-filled.

Synchronization Robustness

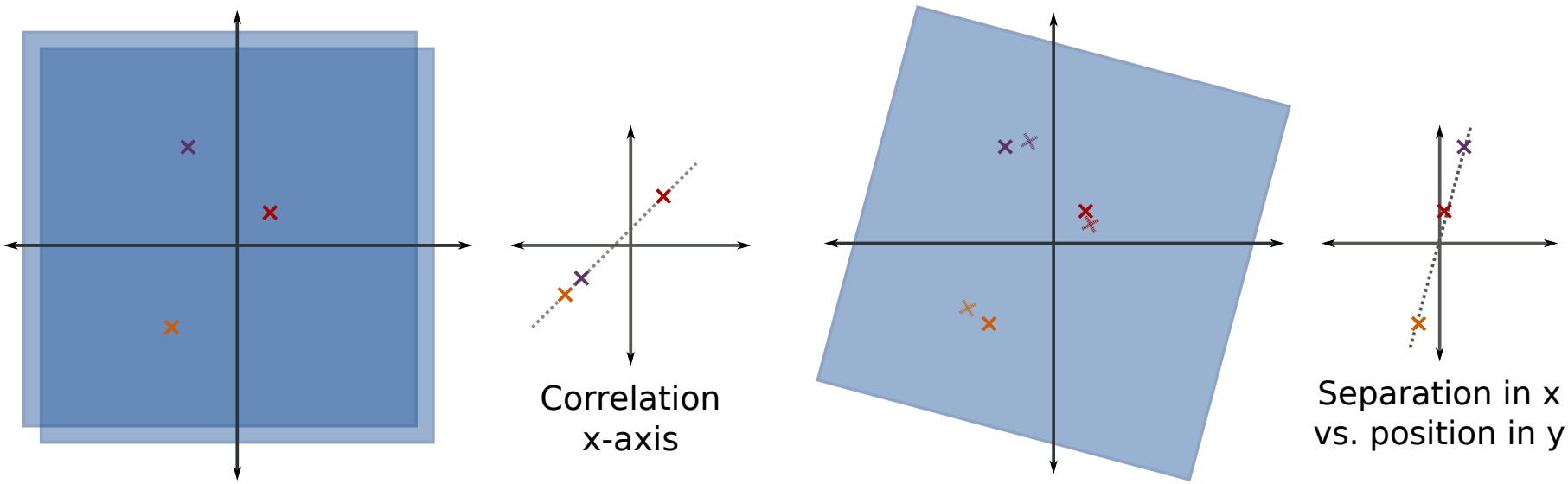
- The devices need not be well synchronized.
- The algorithm isn't affected by the frequency and proximity of desynchronizations.
- The synchronization isn't cumulative.
- The discrimination of synchronized and desynchronized events is very powerful.





Alignment Procedure

- The alignment of multiple sensors in the same beamline is performed in two steps:
  1. Coarse alignment based on inter-plane correlations.
  2. Fine alignment based on unbiased residuals of reconstructed tracks.
- This procedure detects X, Y translations and Z rotations.
- The procedure is not sensitive to displacements in Z (Z translations, X and Y rotations) due to the collimation of particle beams.



Unbiased Residual Alignment

- A particle track can be resolved to sub-pixel accuracy.
- Using track residuals allows for sub-pixel alignment.

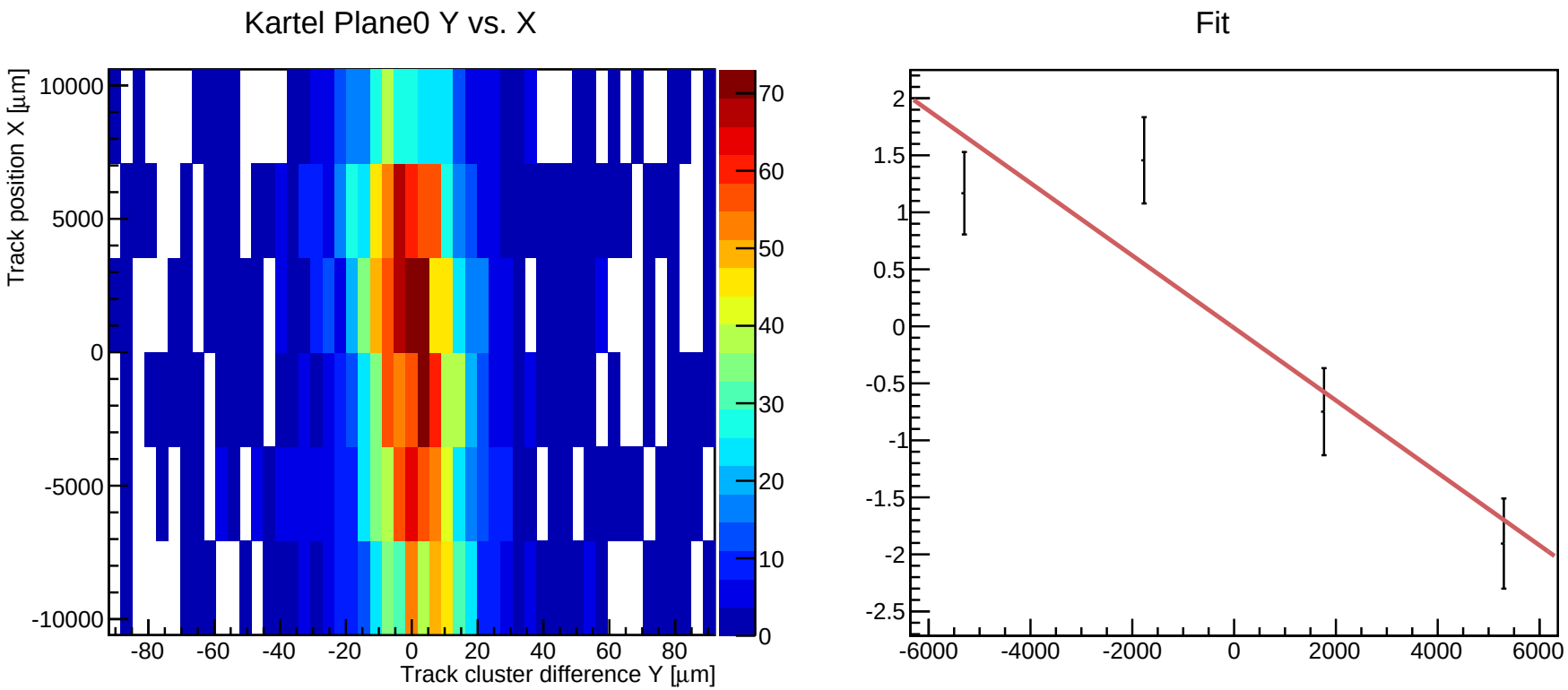


Figure : Depiction of the residual alignment procedure at an intermediate step. Note that the fit graph is rotated by 90 degrees.

Performance in the DESY testbeam

- The tracks fits in Kartel achieved a resolution of approximately 3 μm.
- The residual plots for the reference sensors shows they were aligned to very close to that resolution.
- The algorithm can be adapted to accommodate various beam conditions and sensor pitches.

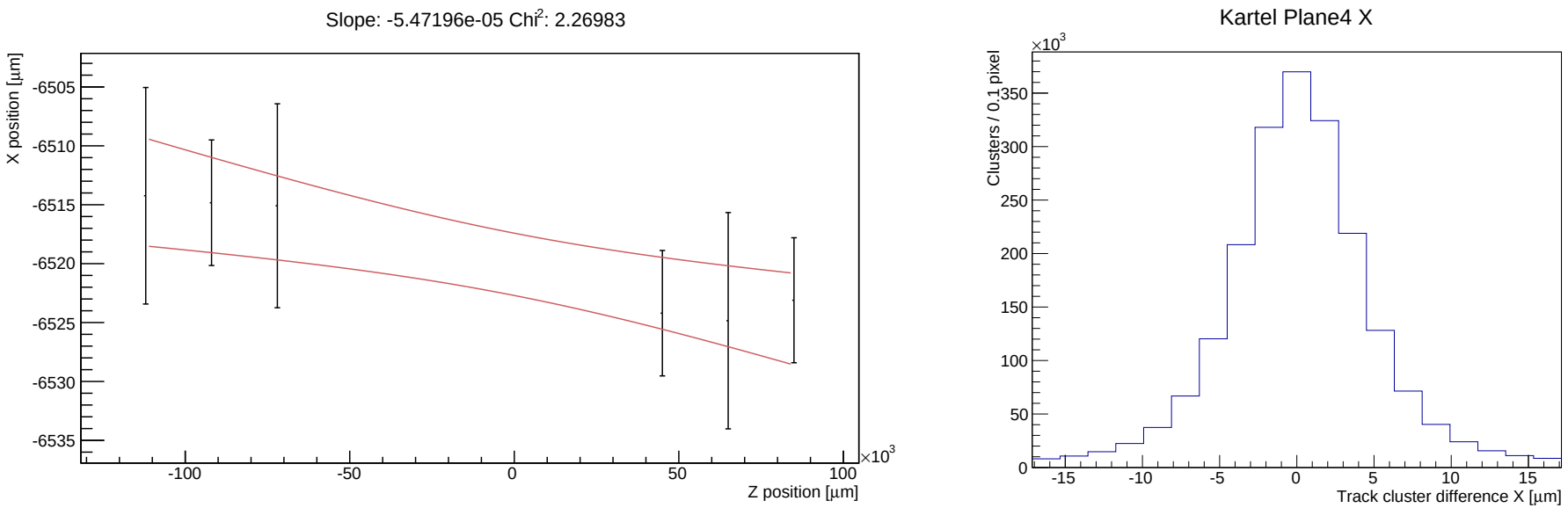
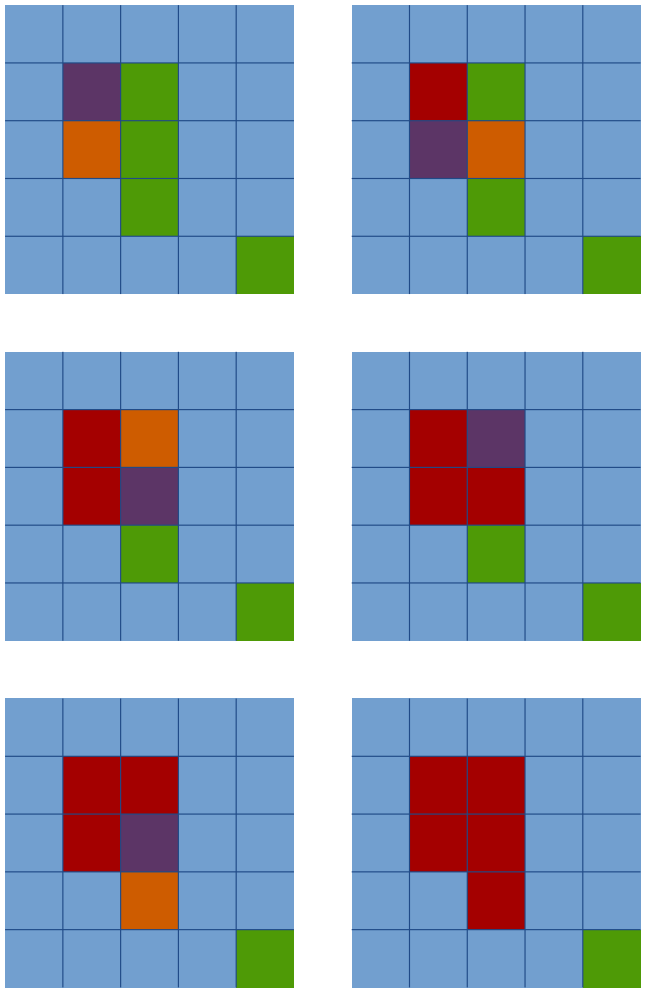


Figure : Depiction of a track's uncertainty band and the residual achieved in a reference plane.

Clustering Algorithm

- The clustering algorithm is a recursive search for neighbouring hits.
- The algorithm is depicted in the following figure.
- Green: unclustered hit
- Purple: seed for neighbour search
- Orange: found neighbour
- Red: hit belonging to the cluster



Tracking Algorithm Example

- The tracking algorithm chooses a seed cluster in the first 2 planes.
- The seed then searches up to 2 planes further for a cluster within a certain "beam spot".
- The track can bifurcate if more than one cluster is found. The best is then chosen.
- The track must meet a minimum requirement of clusters.
- Track **1** is too short, and track **2a** is chosen over track **2b**.

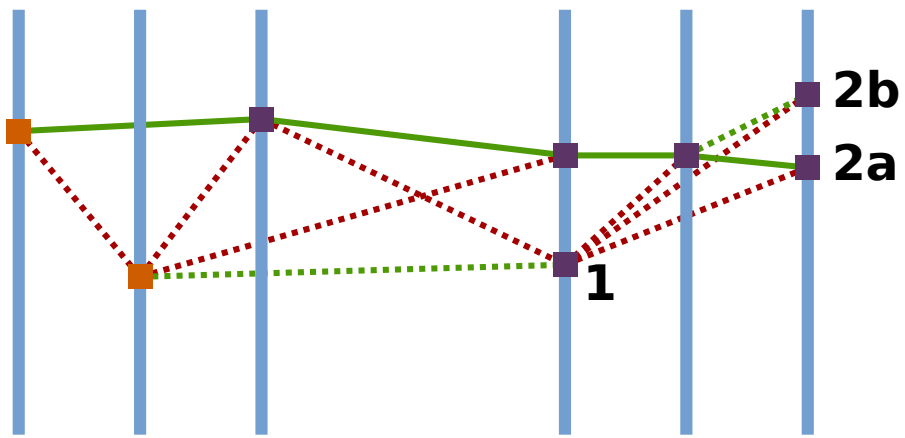


Figure : This is a projection of six reference planes making a telescope. Red lines are rejected track segments.

Software Analyzers

- The analysis is performed by a collection of analyzer classes.
- The classes are derived from a base class which defines the interface with other aspects of the program.
- Analyzers get access to each event during the loop.
- Analyzers interface with a set of standard event, track, cluster and hit cuts which are configured from files.
- Common analysis tasks are already implemented in Judith.

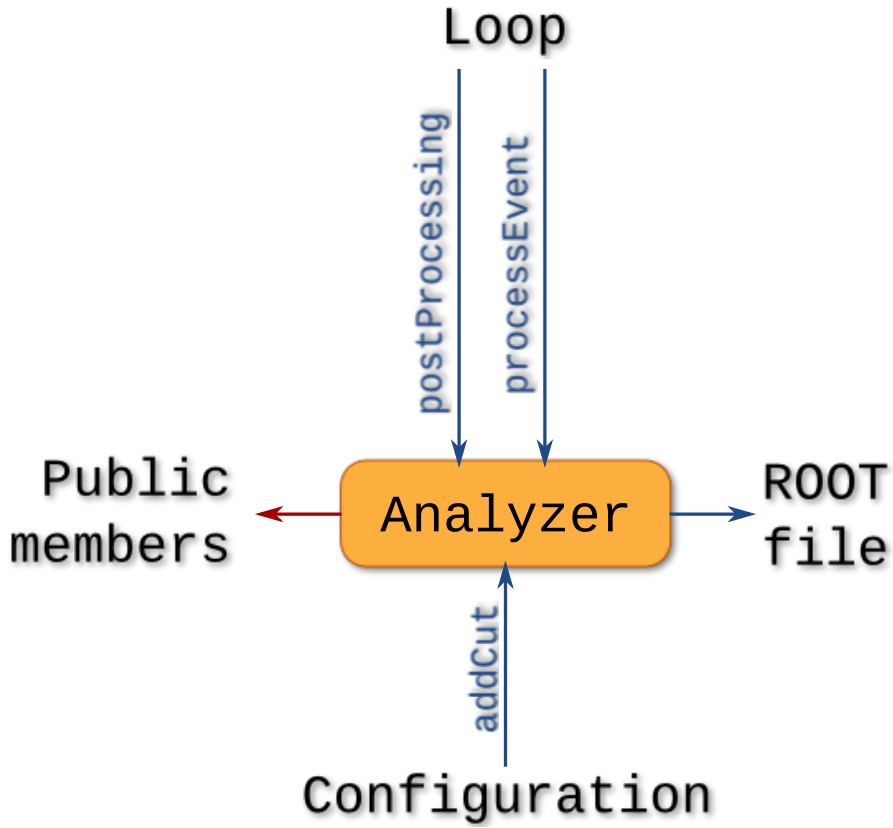


Figure : Depiction of the analyzer base class. The blue arrows depict data flow prescribed by the interface.

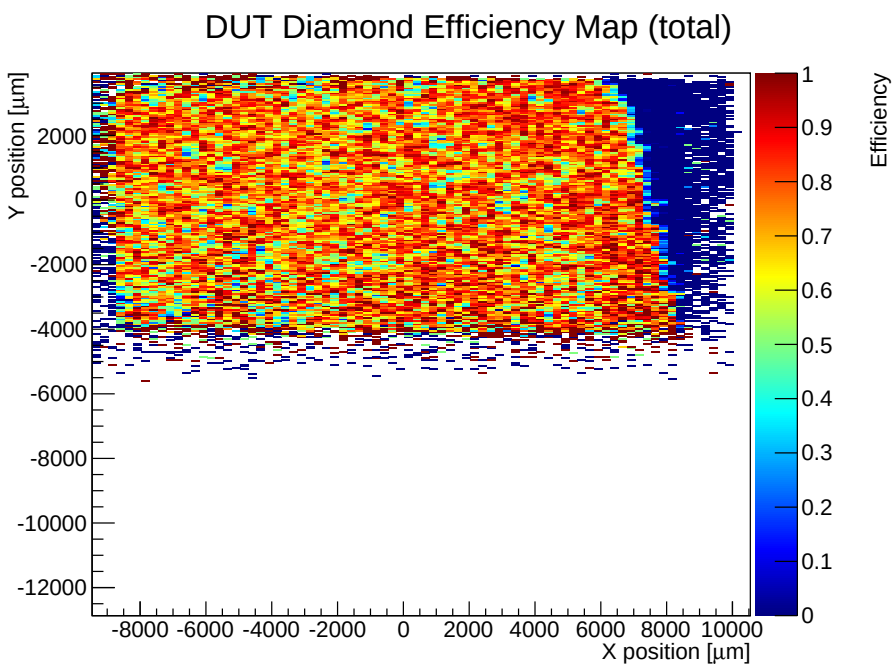
Testbeam Efficiency Analyzer

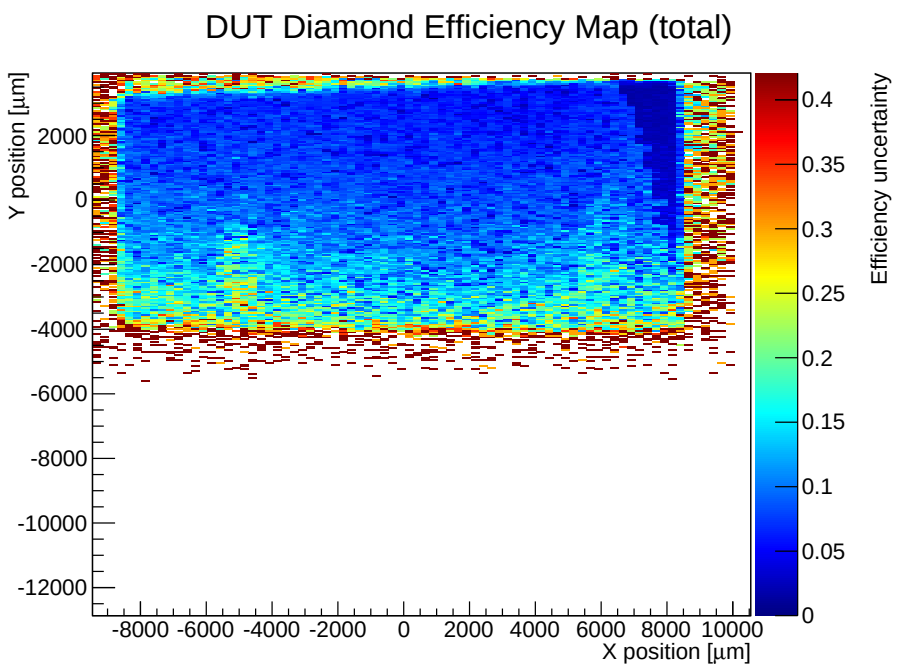
- The following shows how the efficiency analyzer was used in the DESY testbeam.
- The analyzer is configured from a file.
- Parameters are set which will influence the behaviour of the analyzer and its output.
- Cuts are passed to select good events for efficiency analysis.

```
[Efficiency]
  active       : true
  suffix       : C     # Suffix appended to histogram names
  relative to  : 1     # Consider only events with a match in this DUT plane
  pix group x  : 1     # Group this many pixels in the map
  pix group y  : 1
  pix bins x   : 40    # Divide one pixel into this many bins in X
  pix bins y   : 8
  cut cluster matchdist max: 7
  cut track clusters equal: 6
  cut track chi2 max: 5
[End Efficiency]
```

Testbeam Efficiency Map

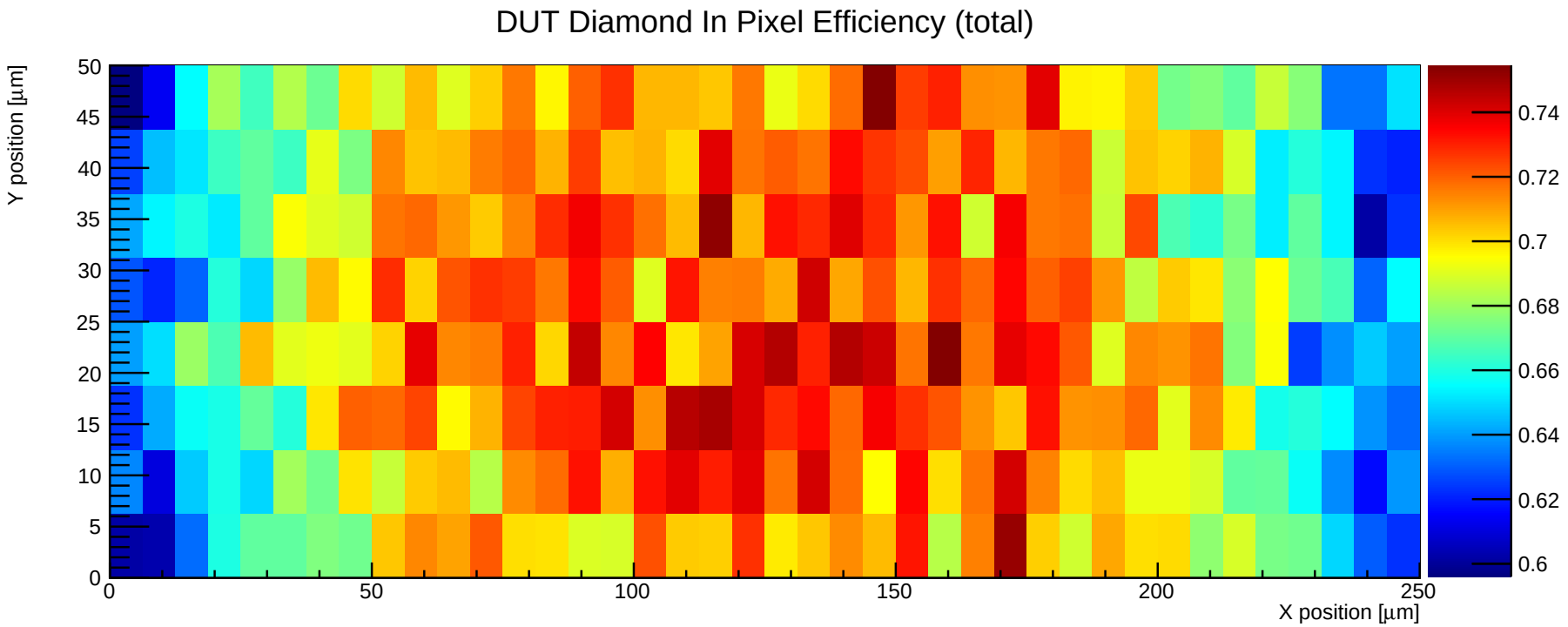
- The following efficiency map is produced along with its corresponding uncertainty map.
- The shape of the overlap of the two scintillator trigger windows is visible in the uncertainty plot.
- The top right corner is defective on this module.





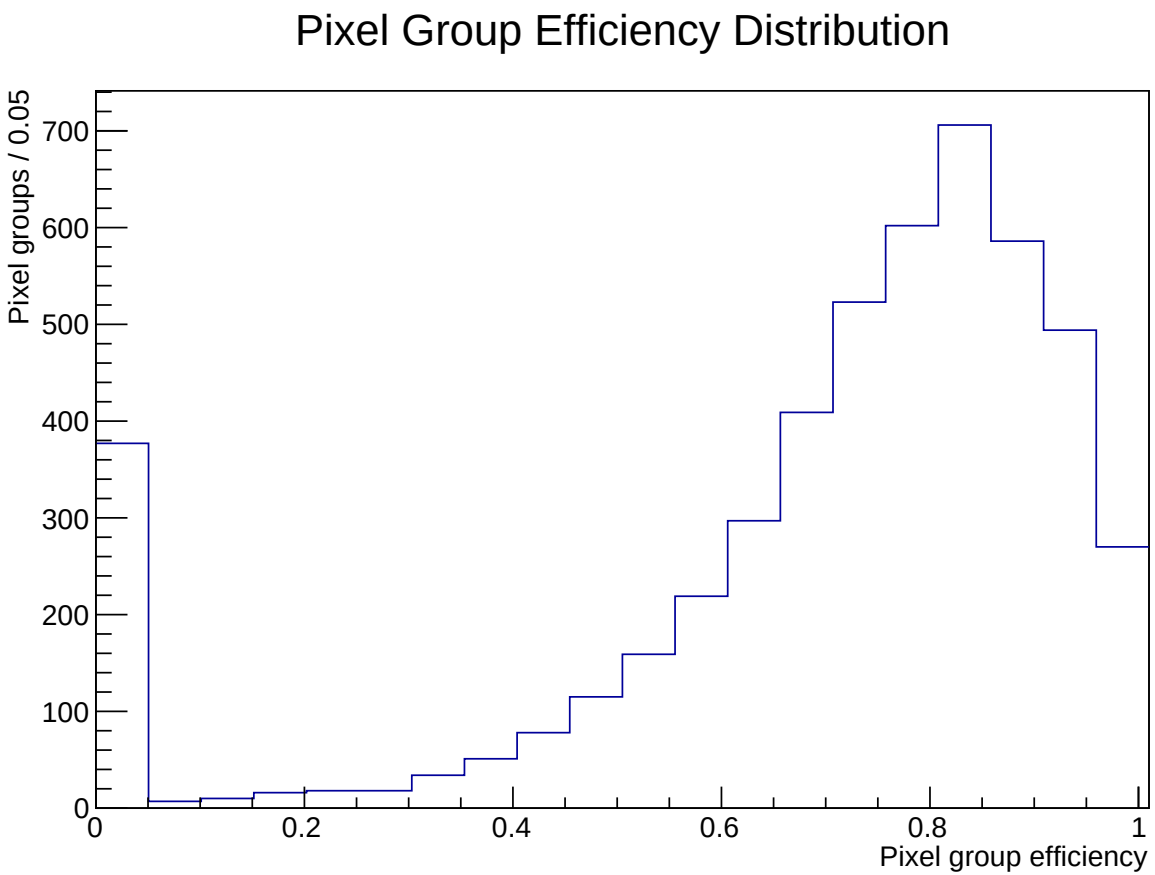
Testbeam In Pixel Efficiency

- The analyzer produces a plot of the efficiency for a track to be detected in a pixel, based on where the track intercepted the pixel.
- This is useful provided that the track resolution is much smaller than the DUT pixel pitch.
- The effect of charge sharing with neighbouring pixels is evident in this plot.



Testbeam Efficiency

- The distribution of pixel efficiencies can also be produced.
- The following only includes those pixels for which the uncertainty is smaller than 10%.



Conclusion

- Judith is a software package which performs synchronized data analysis of several pixelated particle detectors.
- The Judith data is stored in ROOT n-tuples which can be analyzed independently of the Judith software.
- Judith's data I/O module can provide navigable object based access to event data in a standalone analysis.
- The full Judith software features a robust synchronization routine.
- and provides an extensible analysis framework.
- Judith is based on robust, simple algorithms allowing for fast computation on a personal computer.
- Judith can be configured at run time from a series of simple configuration files.