

Testbeam Reconstruction

- **Testbeam re-cap**
- **Generic steps**
- **Corryvreckan**

24/06/11

Coming from

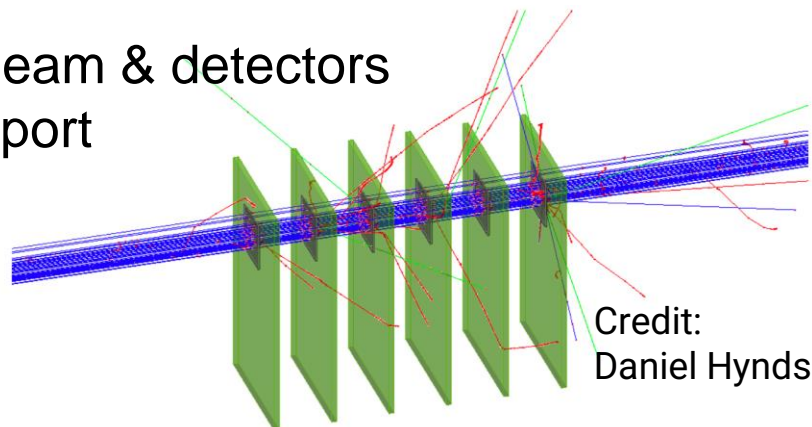
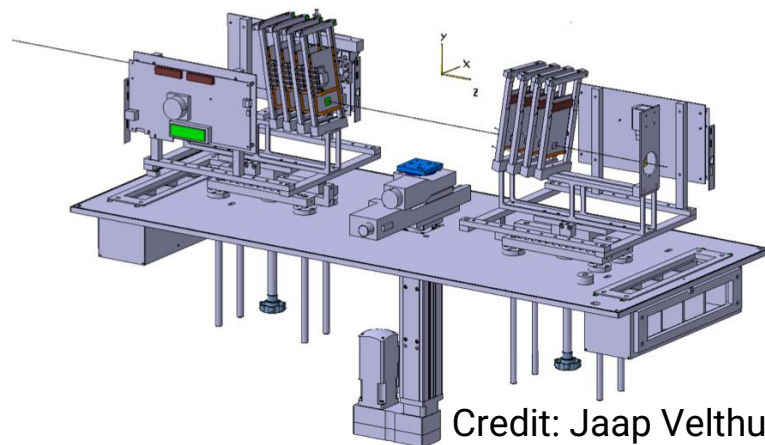
This section leads on from previous Instrumentation topics

Testbeams: [link](#)

1. Part1 - concepts: efficiency & purity
2. Part2 - practice: facilities & activities

Allpix**2

1. [Part1](#) – setting up simulation geometry, beam & detectors
2. [Part2](#) – detector details and charge transport



Building on concepts introduced, we will look at a tool for reconstruction of testbeam data.

Overview

Part I: Testbeam re-cap


- Why bother?
- Set-up
- S/N
- Multiple scattering



Motivations

Part II: Generic reconstruction

- Goal: data \rightarrow determination
- Steps
 - Hits, Clusters, Correlations, Alignment, Tracking
- Output metrics: efficiencies, etc.



Common concepts

Part III: Corryvreckan

- Overview
- Reconstruction steps
- Example using `allpix**2` input



Specific tool

NB often I write pixels when I mean pixels or strips

Part I

Testbeam Re-cap

Re-cap: TB Motivation

Goal: *Quantitative* characterisation of detector *properties*

What can you get in the lab?

- Cosmics
 - Free source of high energy particles
 - But beam parameters not well defined: energy, direction, rate
- Laser
 - Well defined beam parameters: energy, direction, rate
 - But:
 - Limits on beam spot precision
 - Not appropriate if sensor is metallised
- Radioactive sources
 - Well defined beam energy
 - But not defined direction or rate (even if columnated)
- Test charges/pulses
 - Only appropriate for electronics

What you get at testbeam

- Well defined radiation source: energy, direction, rate, beam spot

Re-cap: Set-up

Ugly Truth: complex, multi-detector set-up

“all the detectors”

- Multiple detector planes
 - Perhaps multiple readout formats

“at the same time”

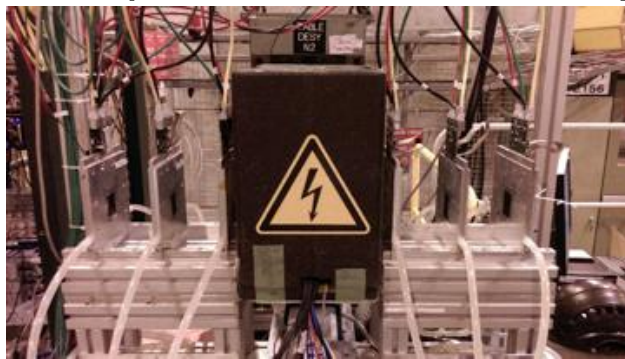
- Coordinate readout: synchronise or associate
 - Triggers or timestamps

“all lined up”

- Make sure beam passes through detector planes
- Understand the relative positions of detector planes

- ◆ Testbeams are very simple but there are many issues.
- ◆ Need to reconstruct the track that goes with your hit
 - Need to look at all detectors at the same time and have them all lined up.
 - Not easy!

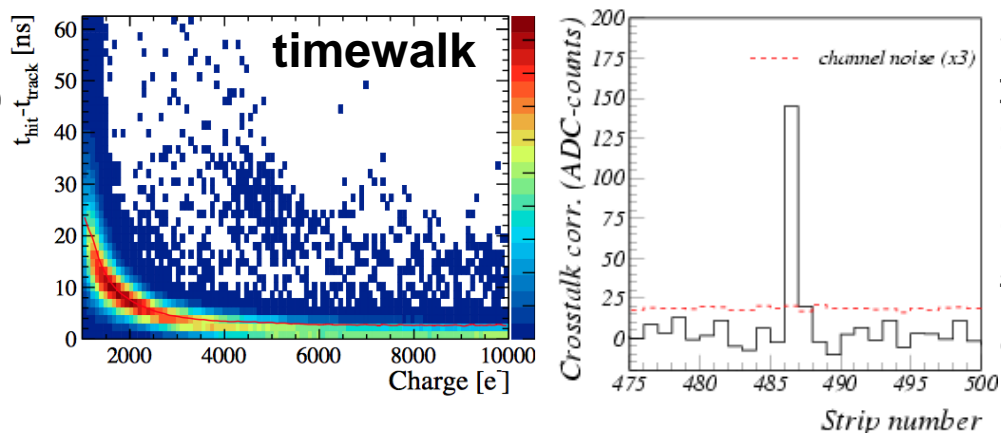
Credit: Jaap Velthuis



Re-cap: Signal & Noise

Deciding reality

- Signal: relevant data (efficiency)
 - maximise true positives
 - minimise false negatives
- Noise: irrelevant data (purity)
 - minimise false positives
 - maximise true negatives



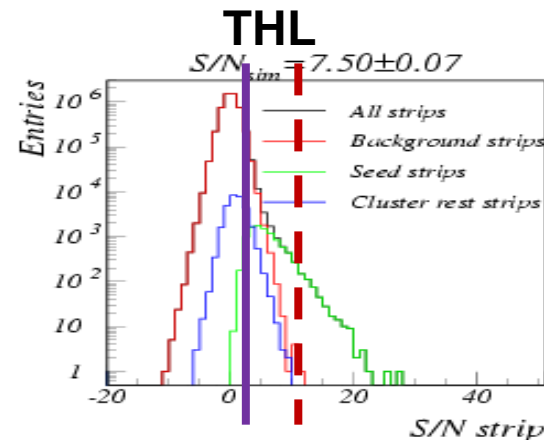
Credit: Jaap Velthuis

Multiple noise sources

- Electronic noise: $raw(i, k) = ped(k) + n_{random}(i, k) + n_{common}(i, k) + q(i, k)$
- Timewalk (*aka* out of time hits)
- Auxiliary scattering
 - Non-parallel tracks
 - Beam contaminants

Noise mitigation

- Channel tuning / masking (specific)
- Charge threshold (general)



Credit: Jaap Velthuis



Re-cap: Multiple Scattering

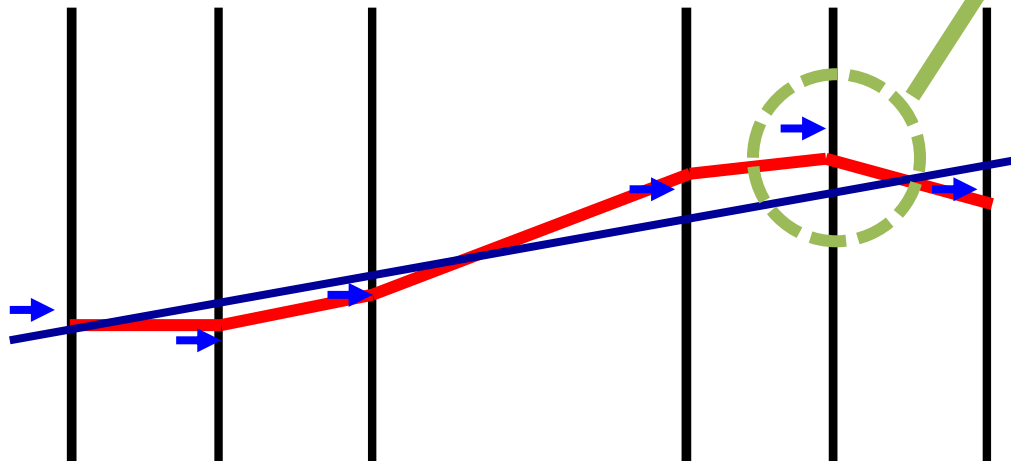
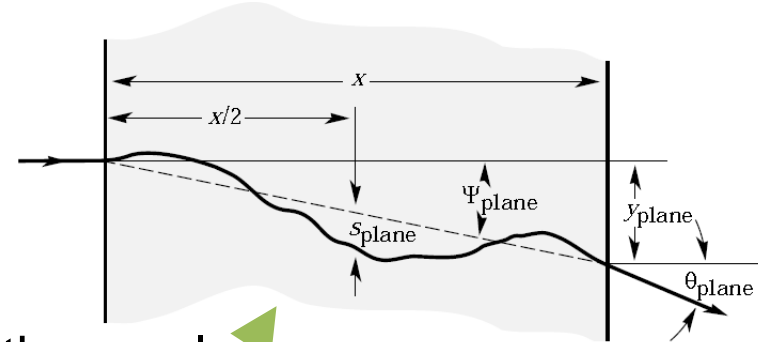
Building a straight (?) track

Standard method of fitting a straight line to coincident hits across detector planes

- Works best for high/hard p_t beam

Particles are deflected in lower energy beams

- Thin scattering: angle of deflection
- Thick scattering: angle of deflections and orthogonal offset



Example:
With AIDA telescope

- Low energy:
5GeV e
- High energy:
120GeV π

Part II

Generic Construction

Generic (Re-)Construction

Goal: *Quantitative characterisation* of detector properties

Use data collections recorded while detectors operated in the presence of a *well-understood* particle beam

- *Hope* detectors were functioning properly
 - Measured: detector environment: temp, humidity, current stability
 - But: What if detector channels were broken?
- *Hope* the beam was as expected
 - Measured: flux, scintillator triggers, magnet settings
 - But: What if extra scattering from unexpected material?
- *Hope* combined system worked together
 - Measure: quantity of data recorded
 - But: Maybe it's junk?

Suspicious only settled by reconstruction output

Generic (Re-)Construction

Goal: *Quantitative characterisation* of detector properties

Task is broken into successive intermediate steps:

- Raw data interpretation
 - Goal: read information from *data sources* to *common* format
- Hit definitions
 - Goal: select data from *well-functioning* pixels exposed to radiation from *known source*
- Clustering
 - Goal: gather pixels from *common* incident particle into *single* object
- Correlations & alignment
 - Goal: associate clusters across detectors from *common* particle trajectory
 - Goal: position detector *local* planes to *global* layout
- Metrics – compare DUT response to tracks

Generic: Raw data

Goal: Read information from *data sources* to *common* format

Multiple data sources

- Telescope – usually multiple planes of one or two detector types
- DUT (detector under test) – whatever is of interest
- TLU (trigger logic unit) & scintillators – controlling event structure, timestamps
- DCS (detector control system) – relevant environment and operational parameters (e.g. T, RH, I)

More than enough information

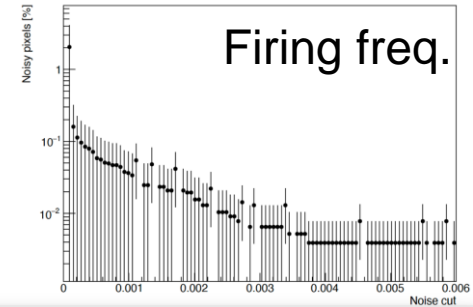
- Need to separate useful from the rest
- Keep enough information to interpret spatial coordinates per detector
 - Possibly charge and timing information as well
- Need to translate to single format for analysis
- Timing may come from detectors or TLU/scintillators

Generic: Hits

Goal: Select data from *well-functioning* pixels exposed to radiation from *known source*

Pixel is *well-functioning*

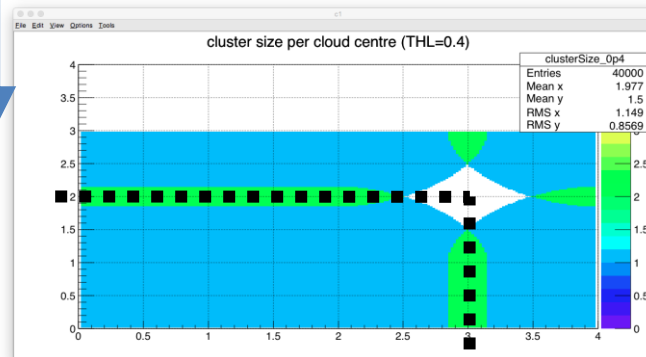
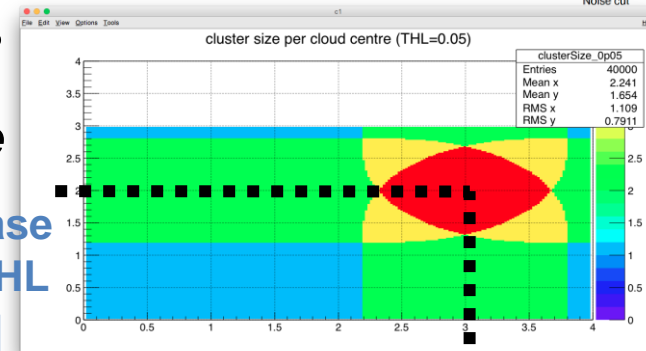
- Not firing constantly or intermittently without radiation
- Filter out pixels with *excessive* firing frequency
 - Compare to distribution of pixel frequencies



Source of pixel response is from the *known source*

- Not electronic noise in ASICs
- Not external contaminant sources
- Not secondary beam components
- Threshold charge deposition using Time Over Threshold (ToT) information
 - Given the expected beam composition/kinematics/profile what is the expected signal?
 - Balance efficiency & purity

Increase
ToT THL



Generic: Clusters I

Goal: Gather pixels from *common* incident particle into *single* object

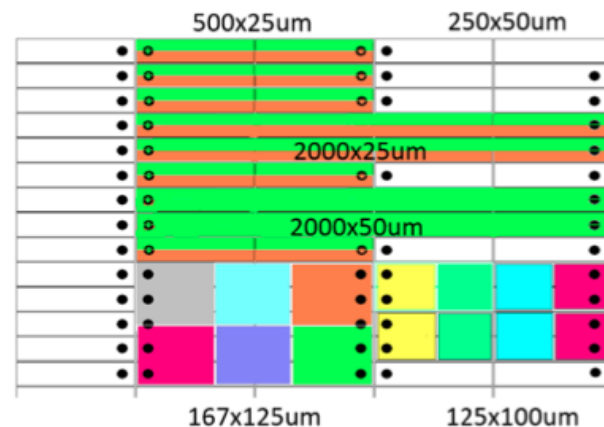
Building cluster

- Intuitively charge deposited in local area of particle track

Common → locally close pixels → neighbouring

Acceptable neighbours?

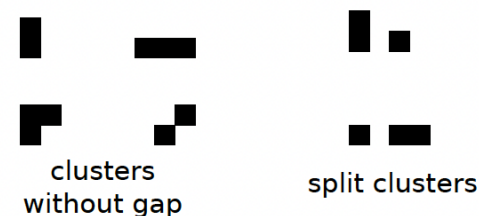
- 4 nearest pixels on square matrix
 - 8 if include diagonals (expect lower ToT)
- What about rectangular/hexagonal matrices?
- next-to-nearest neighbours?



Credit: Marko Milovanovic

Let an algorithm decide based on continuity of cluster?

- Split clusters? – e.g from masked pixel
- Merged/overlapping clusters? – e.g. coincidence



Credit: Jens Kröger

Figure 22: Examples of different clusters with and without a gap.

Generic: Clusters II

Goal: Gather pixels from *common* incident particle into *single* object

Derived cluster parameters – a matter of counting

- Cluster size (in pixels)
- Cluster width in X (in pixels)
- Cluster width in Y (in pixels)
- Cluster charge (sum of pixels)
- Seed/highest/hottest pixel, i.e. greatest ToT

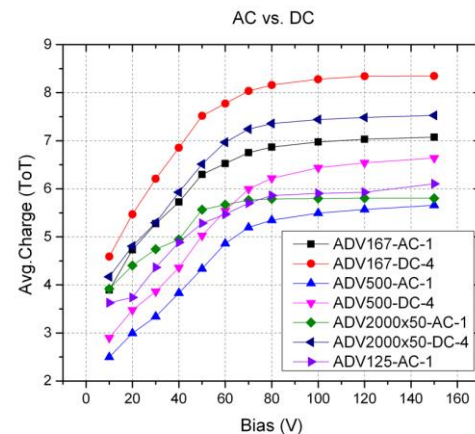
Can be useful parameters themselves: – reflects pitch, depth, charge, V, THL

Defined parameters – a decision

- Cluster position X
- Cluster position Y

Common options

- Average
- Weighted mean – weighted by pixel charge
 - AKA centre-of-mass, barycentre, Newton centre



Generic: Correlations

Goal: *Associate* clusters across detectors

Assuming trajectory of incident particle is straight(ish) between planes, clusters can be *associated* by means of correlation

Compare hits in detector pairs in each dimension

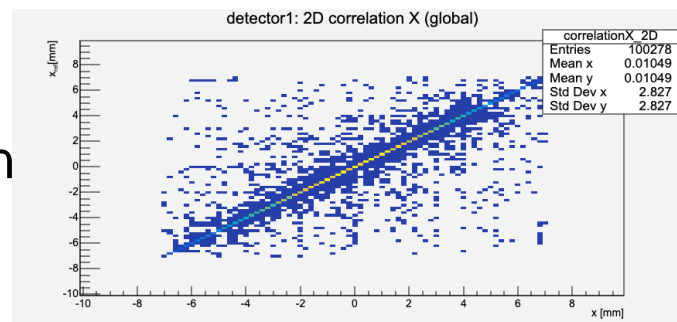
- detector A profile in X/Y Vs detector B profile in X/Y

Straight line shows correlation

- Positive (negative) gradient for (anti-)correlation
- No line \rightarrow no correlation
 \rightarrow Mis-orientation or de-synchronisation

Displacement of (anti-)correlation intersection with origin (LHS lower corner)

\rightarrow Relative misalignment of planes



Generic: Alignment

Goal: Position & orient detector *local* planes in *global* layout

Use single plane as reference to align all others

- Adjust positions based on misalignment wrt *reference* plane

Reference coordinates \rightarrow *global* coordinate frame

- Misalignments minimised *assuming* track trajectory between planes is understood

NB Important distinction:

- $x_{\text{correlation}} = x_{\text{cluster}}$ on reference detector – x_{cluster} on *this* detector
- biased: $x_{\text{residual}} = x_{\text{track intercept}}$ on *this* plane – $x_{\text{track cluster}}$ on *this* plane
 - Choice of cluster is biased
- unbiased: $x_{\text{residual}} = x_{\text{track intercept}}$ on *this* plane – $x_{\text{associated cluster}}$ on this plane
 - Choice of cluster is unbiased

Generic: Track building

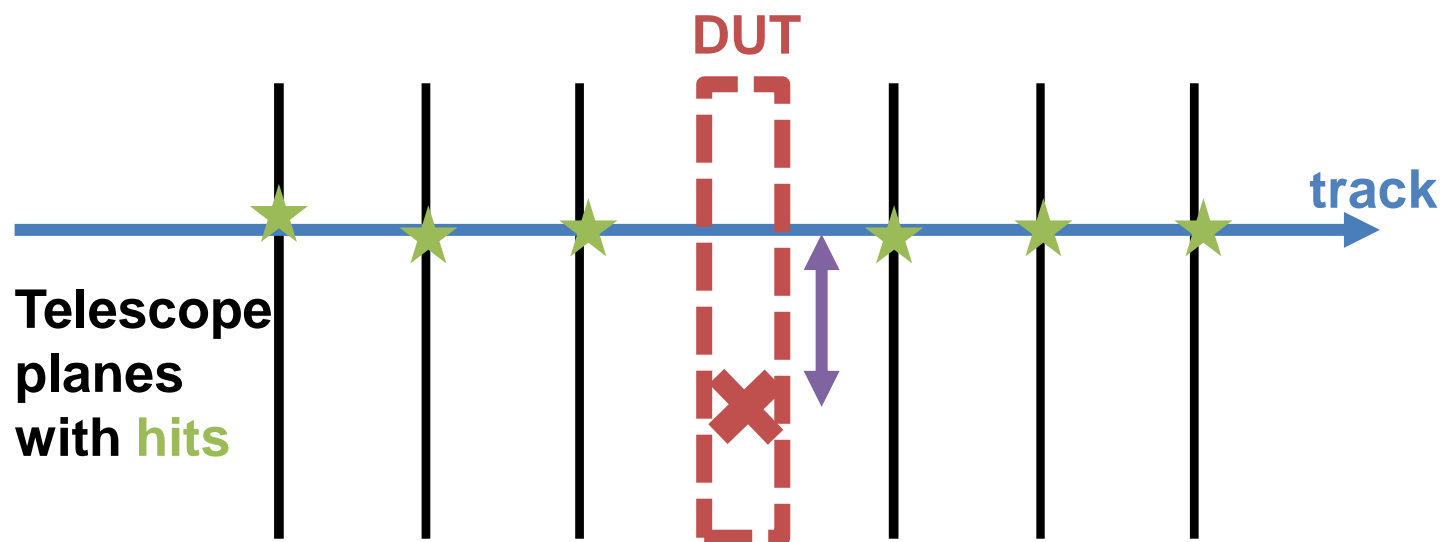
Goal: Build *global* track trajectory from *local* cluster positions in planes

Track defined by minimised misalignments of telescope planes

- DUTs not used in track building → track definition unbiased wrt DUT

Metrics give quantitative characterisation of DUT properties by comparing interpolated track position with DUT response

→ **Residual**: distance



Generic: Metrics

Residuals (distance between interpolated track position and hit cluster)

For binary device (no charge information):

$$\sigma_{position}^2 = \frac{\int_{-p/2}^{p/2} (x_r - x_m)^2 D(x_r) dx_r}{\int_{-p/2}^{p/2} D(x_r) dx_r} = \frac{\int_{-p/2}^{p/2} x_r^2 1 dx_r}{\int_{-p/2}^{p/2} 1 dx_r} = \frac{p^2}{12}$$

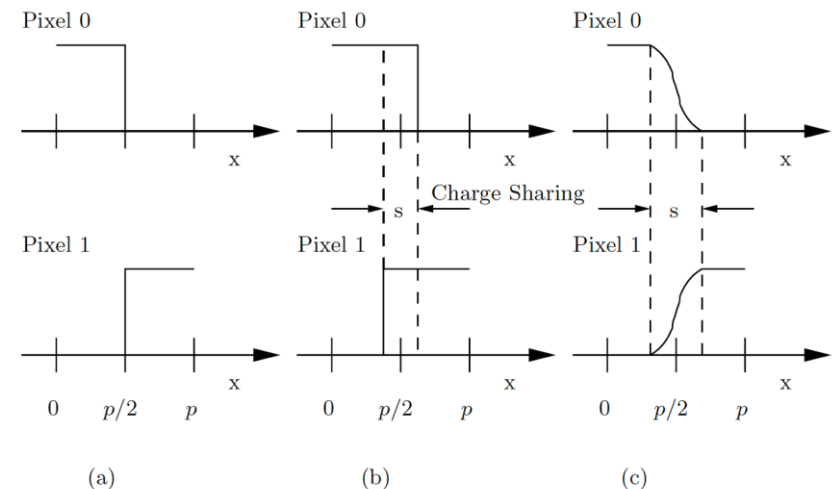
→ Binary residual: $\sigma = p/\sqrt{12}$

Charge sharing improves tracking

- Unlike imaging
- Charge sharing indicates specific hit regions
- Charge weighting improves again

Ideal and limiting case

- Residual should be improved with pixel charge information
- Testbeams are never ideal



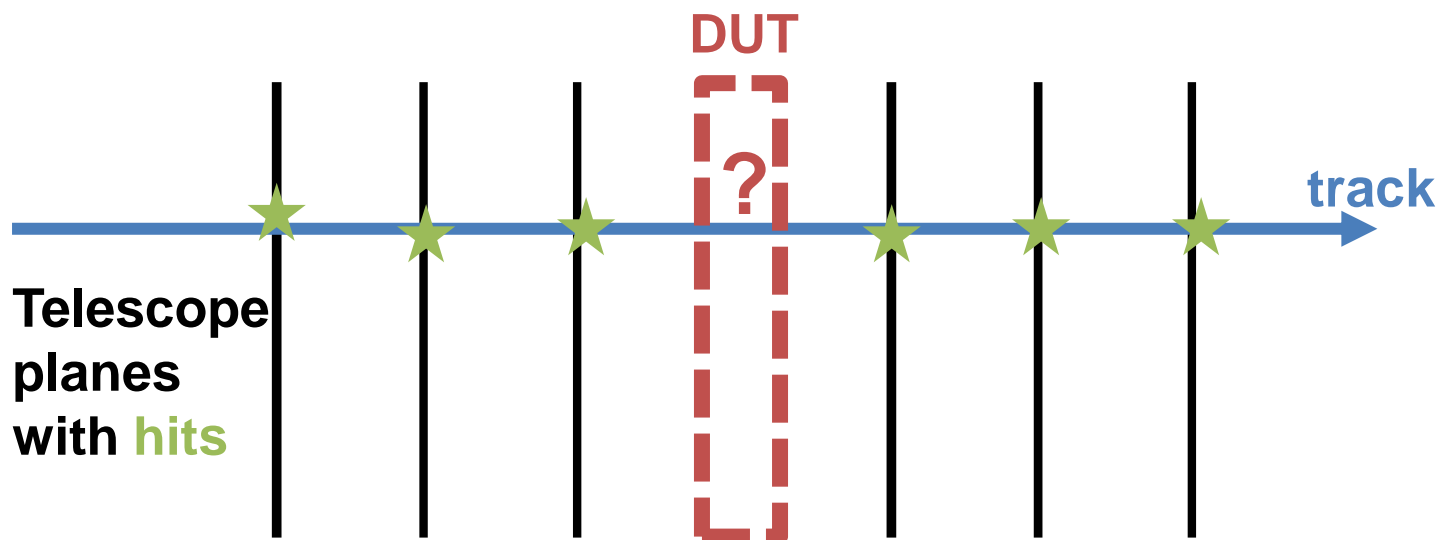
Credit: L. Rossi et al.

Fig. 2.20. Signals in two adjacent pixels as function of the impact position x . Detector with binary readout without charge sharing (a), with binary readout with charge sharing (b), and with analog readout (c)

Generic: Metrics

Efficiency (whole detector)

$$\varepsilon = \frac{N_{\text{matched hits}}}{N_{\text{expected hits}}}$$



- Regions of Interest (Rols) can be used to focus metric on appropriate areas
- E.g. avoid poor bump-bonding, malfunctioning channels

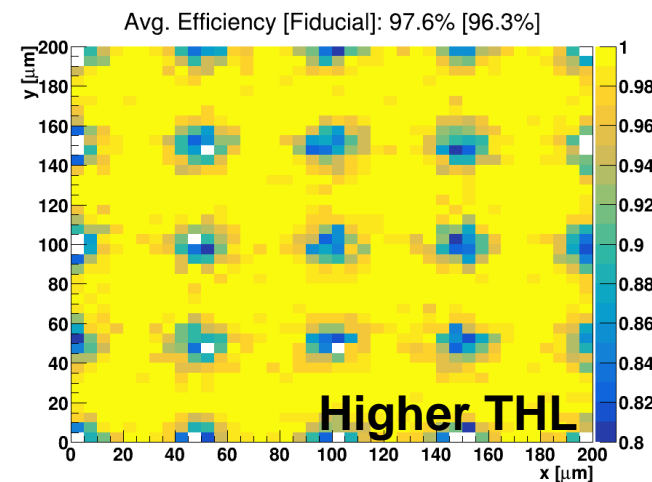
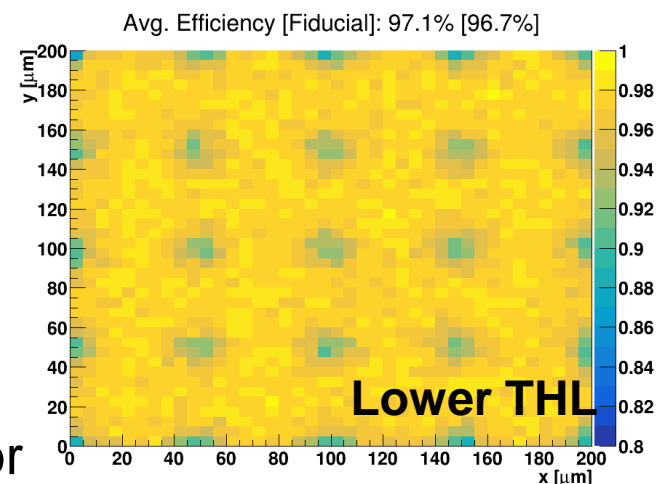
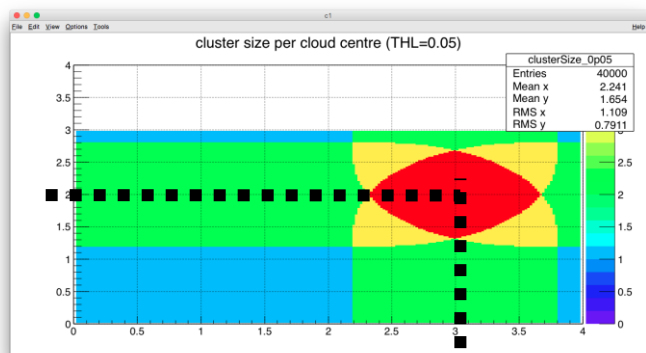
Generic: Metrics

In-pixel Efficiency (specific region)

Sub-pixel resolution

Need high statistics over small region

1. Focus the beam on single pixel
2. Overlap pixel information from across matrix
 - Average pixel response across the detector



Credit: Kenneth Wraight et al.

Part III

Corryvreckan (example set-up in back-up)

Corry: Overview

Testbeams have been around a while; reconstruction almost as long

In the LHC era (my experience) effort was made to standardise software

- EUTelescope: <https://eutelescope.github.io>
- Judith: <https://github.com/gmcgoldr/judith>
- Kepler: <https://gitlab.cern.ch/lhcb/Kepler>

Aimed to be modular and adaptable to various testbeam setups

- Use [EUDAQ](#): Generic Multi-platform Data Acquisition Framework
 - Used by DESY, SPS and other testbeam facilities

Corryvreckan is the most modern reconstruction framework

- Compatible with EUDAQ
- Uses modern modular code base
- Flexible to various testbeam devices and timing
 - Building on the ad hoc solutions in previous frameworks
- Well documented and supported
- (Compatible with Allpix**2)

Corry: Documentation

Useful documents

Website:

<https://project-corryvreckan.web.cern.ch/project-corryvreckan/>

Gitlab repository:

<https://gitlab.cern.ch/corryvreckan/corryvreckan>

- Modules include description, parameters and usage
- Also, for reconstruction:
 - Maintainer, module type, detector type, status

Presentations and tutorials:

<https://project-corryvreckan.web.cern.ch/project-corryvreckan/page/publications/>

- BTTB9 workshop tutorial (Feb 2021):

<https://indico.cern.ch/event/945675/contributions/4184960/>

- Si Pixel Characterisation (July 2020):

<https://www.physi.uni-heidelberg.de/Einrichtungen/FP/anleitungen/F96.pdf>



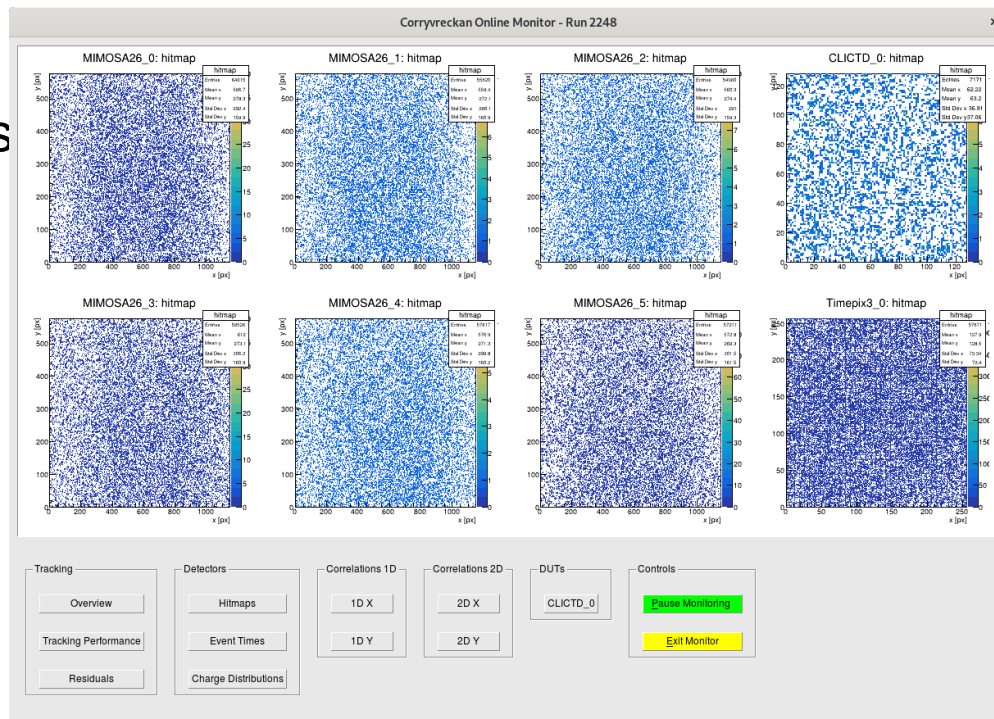
Corry: Functionality

Reconstruction

- Offline data analysis – post-testbeam data taking

On-line monitoring

- On-line data quality check – during data taking
- Looking for
 - Populated plots
 - Data exists
 - Reasonable clusters distributions
 - Devices functioning well
 - Correlations!
 - synchronised planes



Corry: Files

Necessary input to run the software

Global configuration file

- File paths: [corryvreckan] module
 - e.g. geometry file, output histograms
- Reconstruction modules, e.g.
 - Timing: [Metronome], Clustering: [Clustering4D]
 - Find modules: [corryvreckan/src/modules](#)
- Any detector type/name specific configuration

Geometry file

- Plane positions
 - Positions on beam axis (z), x-y plane, rotations (orientations)
- Type of detector
- Find detector scripts: [corryvreckan/src/core/detector](#)

Run reconstruction chain:

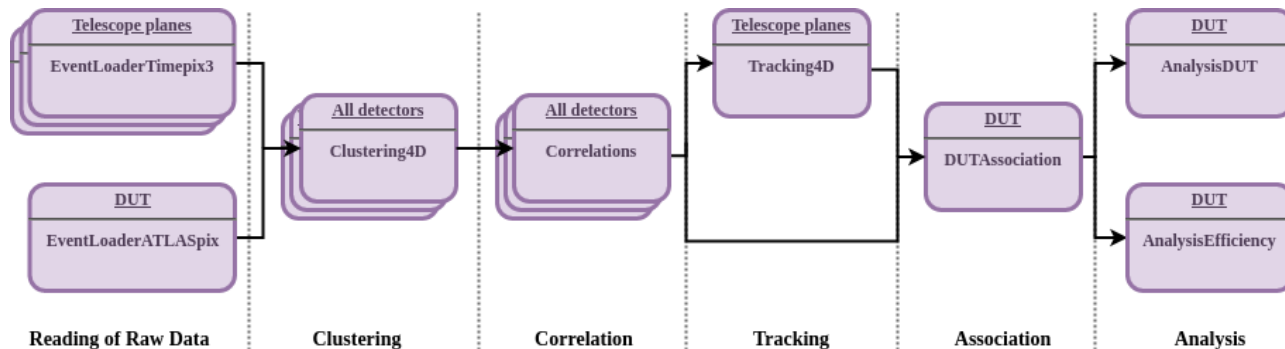
```
> corry -c CONFIGFILE -o OPTION
```

Corry: Steps

Reconstruction chain is configurable to particular case

Typical example:

- Metronome – construct common timing (if necessary)
- EventLoader(s) – reading raw data per detector type
- Clustering – collect pixels into clusters
- Correlations – check relative offsets of detector planes
- Tracking – build track trajectory from subset of planes
- Association – comparison of DUT hits with estimated track position
- Analysis – additional metrics



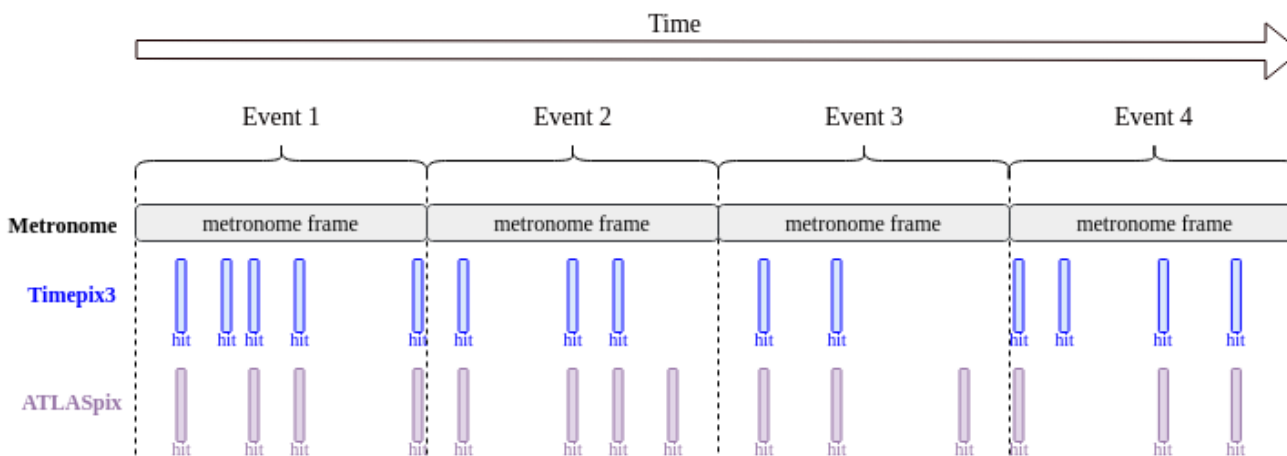
Corry: Metronome

Goal: Common timing across planes → *event definition*

Define modules

- Metronome: set event duration

Timestamps from raw data will be mapped onto metronome “event rhythm”



Corry: EventLoading

Goal: Read information from *data sources* to *common* format

Multiple data sources

- Telescope – usually multiple planes of one or two detector types
- DUT – whatever is of interest
- TLU (& scintillators) – controlling event structure, timestamps
- DCS – relevant environment and operational parameters (e.g. T, RH, I)

Define modules, e.g.

- EventLoaderTimepix3: timepix3 detectors
- EventLoaderATLASpix: prototype atlas ITk detector

Set type/name value to specify which detectors should be read by the module

Outputs (per plane)

- 2D hit position maps
- Pixel ToT distribution

Corry: Clustering

Goal: Gather pixels from *common* incident particle into *single* object

Define modules, e.g.

- ClusteringSpatial: use position information for clustering
- Clustering4D: use position and time information for clustering

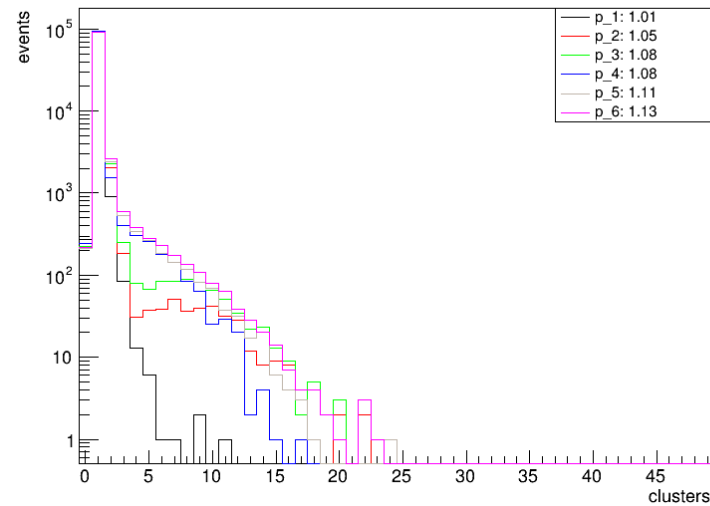
Set distance/time vicinity to gather pixels into clusters

Outputs (per plane)

- Cluster parameters per plane: cluster size, cluster size X&Y, multiplicity, cluster charge
- Cluster seed charge
- Pixel time distributions (if appropriate)
- 2D cluster position maps

Corry: Clustering II

clusterMultiplicity



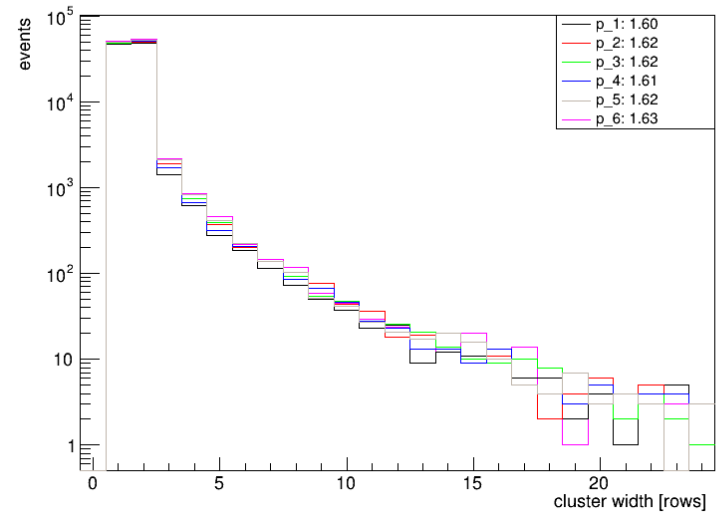
Sample:

- 100k events
- 120GeV π^+
- Telescope only

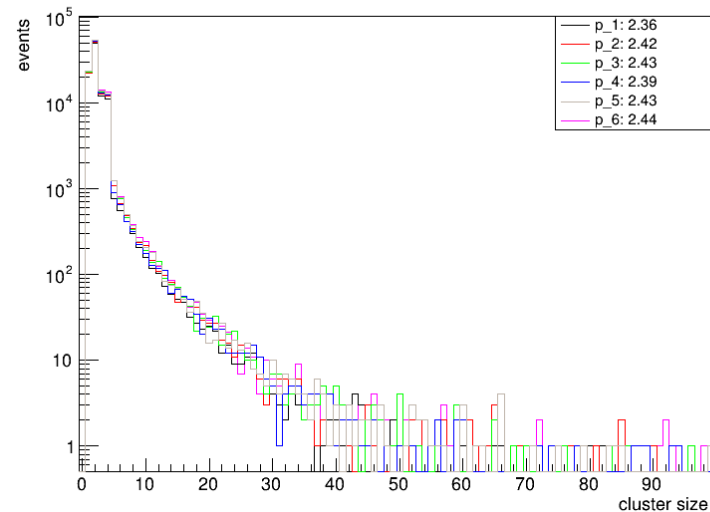
NB

- Cluster sizes increase downstream
- Multiplicity increases downstream

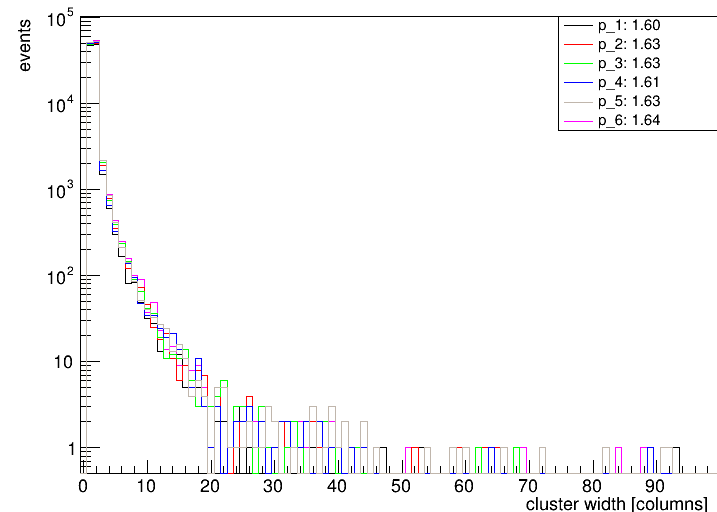
clusterWidthRow



clusterSize



clusterWidthColumn



Corry: Clustering III

	name	max	mean	rms	entries
0	clusterMultiplicity:1	95189	1.009329	0.143971	96470
1	clusterMultiplicity:2	93610	1.052959	0.555012	96470
2	clusterMultiplicity:3	93045	1.079631	0.709691	96470
3	clusterMultiplicity:4	93158	1.080367	0.613131	96470
4	clusterMultiplicity:5	91966	1.110729	0.750807	96470
5	clusterMultiplicity:6	91453	1.132756	0.858308	96470

Highest value in
column
highlighted

	name	max	mean	rms	entries
0	clusterWidthRow:1	47799	1.596813	0.849705	97370
1	clusterWidthRow:2	49926	1.618313	0.891837	101579
2	clusterWidthRow:3	51411	1.621890	0.881944	104152
3	clusterWidthRow:4	51436	1.607589	0.874616	104223
4	clusterWidthRow:5	52975	1.622856	0.881150	107152
5	clusterWidthRow:6	54297	1.628899	0.894488	109277

Sample:

- 100k events
- 120GeV π^+
- Telescope only

NB

- Cluster sizes increase down stream
- Multiplicity increase downstream

	name	max	mean	rms	entries
0	clusterSize:1	49254	2.360942	2.012990	97370
1	clusterSize:2	50541	2.416353	2.147912	101579
2	clusterSize:3	51754	2.425142	2.130053	104152
3	clusterSize:4	52393	2.387144	2.086118	104223
4	clusterSize:5	53078	2.426878	2.144334	107152
5	clusterSize:6	54029	2.442701	2.168152	109277

	name	max	mean	rms	entries
0	clusterWidthColumn:1	47788	1.600933	0.994484	97370
1	clusterWidthColumn:2	50054	1.626208	1.054914	101579
2	clusterWidthColumn:3	51651	1.629876	1.025566	104152
3	clusterWidthColumn:4	51304	1.613187	1.035824	104223
4	clusterWidthColumn:5	52826	1.634145	1.185968	107152
5	clusterWidthColumn:6	54097	1.638164	1.106995	109277

Corry: Correlations

Goal: *Associate* clusters across detectors

Define modules, e.g.

- Correlations: correlation and timing plots

Reference detector flagged in geometry file using *role* parameter

Outputs (per plane)

- Col/row to col/row 2D correlation maps
- Time 2D correlation maps (if appropriate)
- X/Y to X/Y 2D correlation maps
- Pixel/cluster hit maps
- X/Y correlation profiles

Computationally intensive step

- Not required for every reconstruction iteration *or* all events
 - Can be neglected when satisfied with orientations

Corry: Correlations II

	name	max	mean	rms	entries
0	correlationX:1	97420	0.000000	0.370143	100278
1	correlationX:2	88028	0.004270	0.780464	104573
2	correlationX:3	87407	-0.003160	0.911781	106888
3	correlationX:4	79815	-0.003207	0.943961	106014
4	correlationX:5	77795	-0.004196	1.080399	108903
5	correlationX:6	75531	-0.006043	1.156216	110899

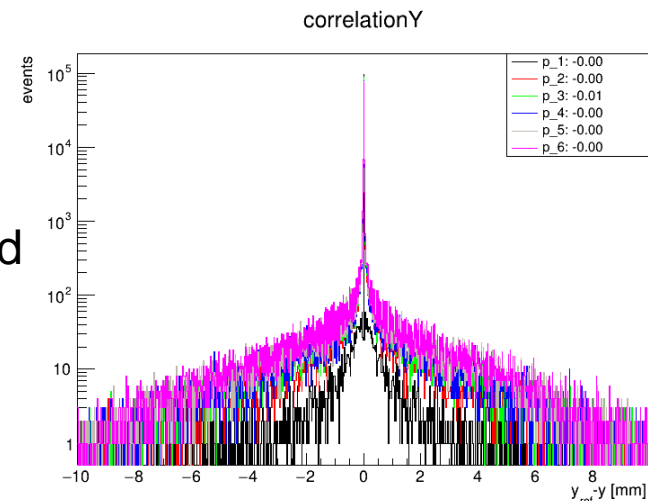
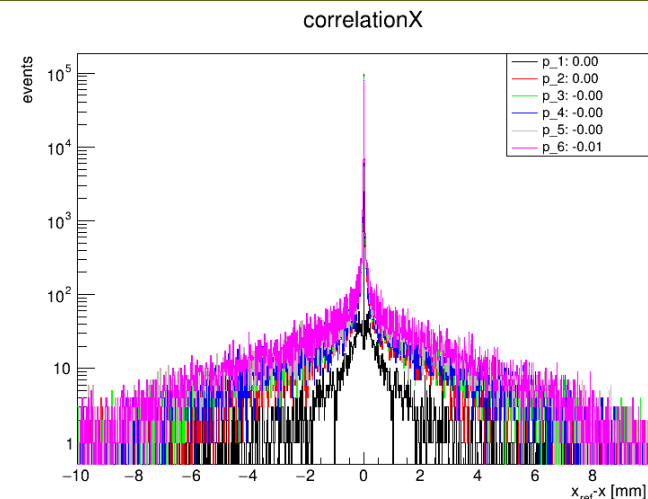
Sample:

- 100k events
- 120GeV π^+
- Telescope only

NB

- Multiplicity increases downstream
- RMS increases downstream
- Means ~ 0.0
→ well aligned

	name	max	mean	rms	entries
0	correlationY:1	97408	-0.000100	0.388928	100278
1	correlationY:2	88061	-0.000492	0.773498	104573
2	correlationY:3	87420	-0.005868	0.893432	106888
3	correlationY:4	79880	-0.002877	0.941641	106014
4	correlationY:5	77709	-0.001136	1.064107	108903
5	correlationY:6	75606	-0.001076	1.129742	110899



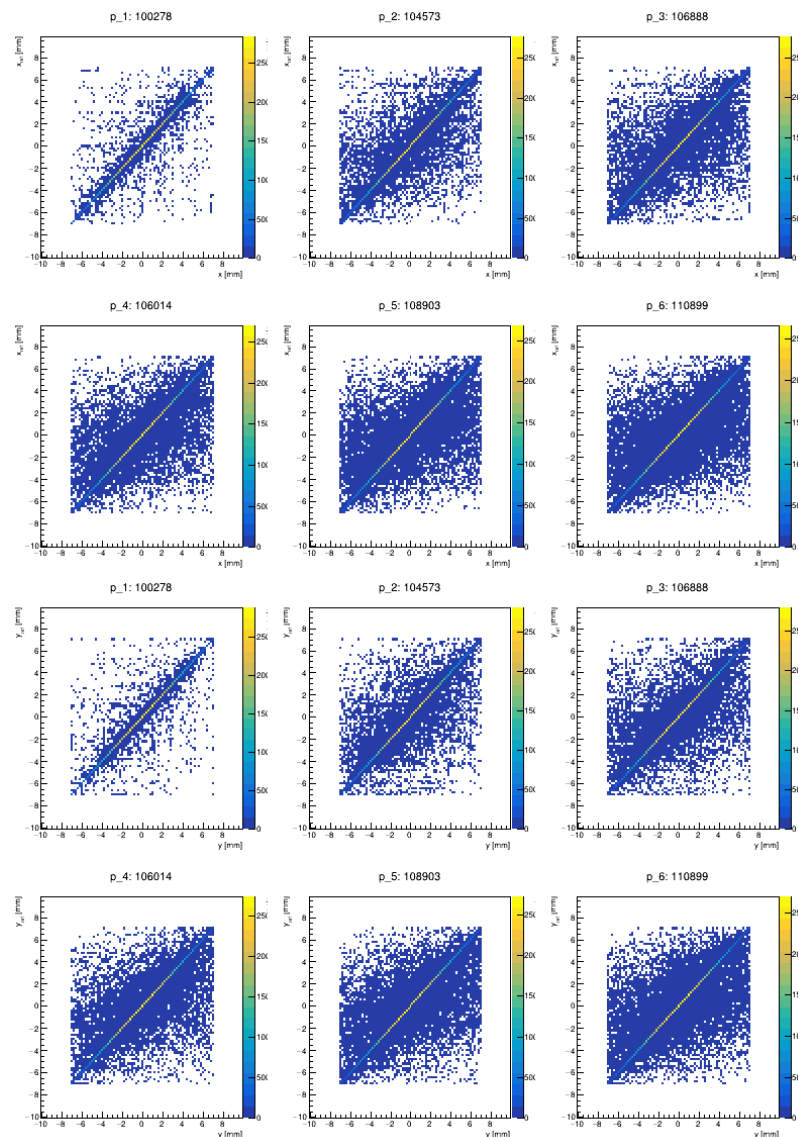
Corry: Correlations II

Sample:

- 100k events
- 120GeV π^+
- Telescope only

NB

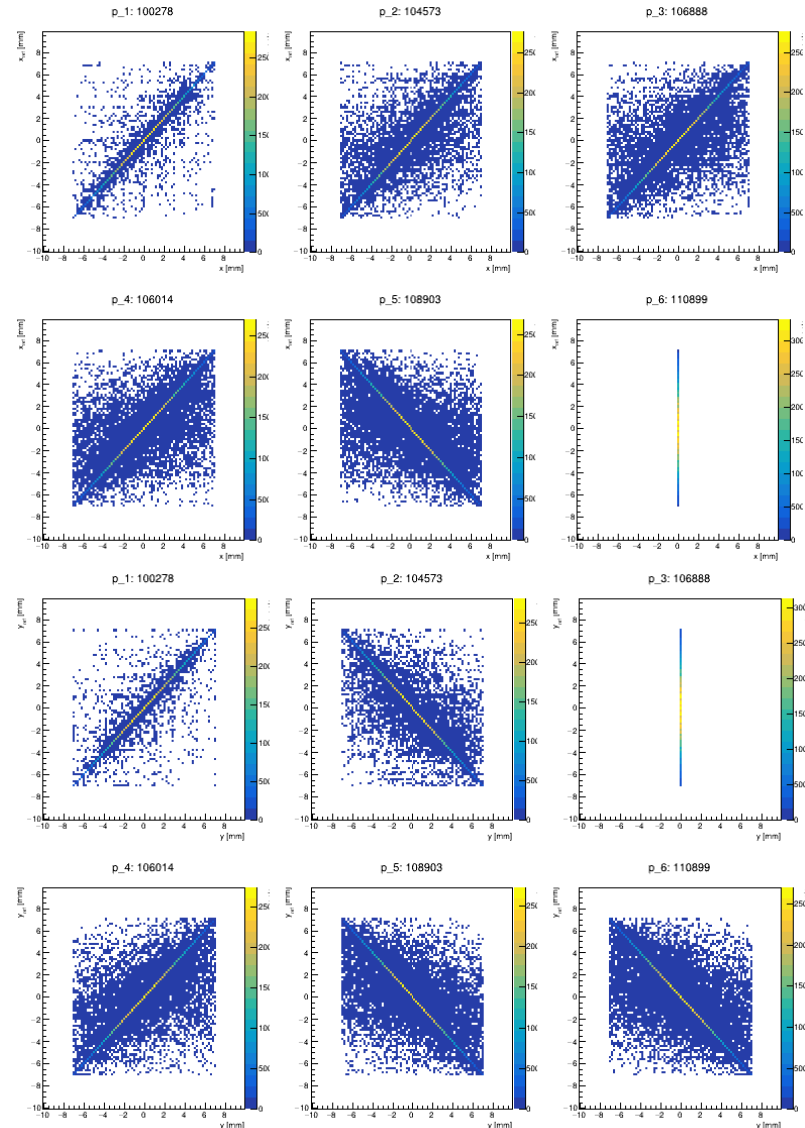
- Correlations BL to TR
→ good orientation
- Origin offset ~ 0.0
→ well aligned
- Multiplicity increases downstream
- Correlations “spread” moving downstream



Corry: Correlations IV

BAD geometry file example

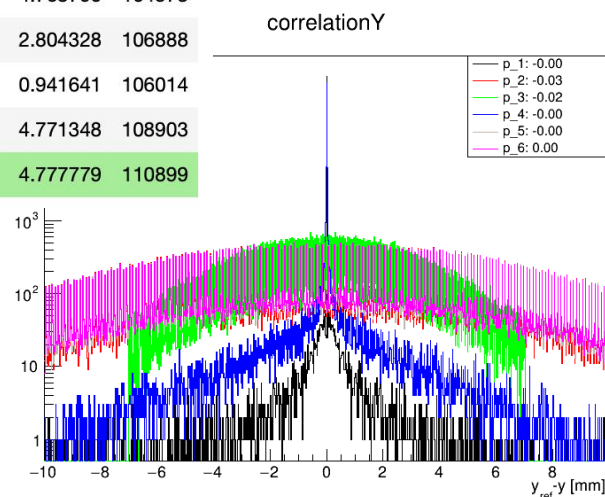
- 180° rotation
 - Reverse correlation (TL to BR) in map
 - Peak lost in profile
 - Low max value
 - Large RMS
- 90° rotation
 - Lost correlation (2D) & peak (1D)



X correlations

Y correlations

	name	max	mean	rms	entries
0	correlationY:1	97408	-0.000100	0.388928	100278
1	correlationY:2	520	-0.033893	4.763790	104573
2	correlationY:3	693	-0.019865	2.804328	106888
3	correlationY:4	79880	-0.002877	0.941641	106014
4	correlationY:5	525	-0.002883	4.771348	108903
5	correlationY:6	517	0.000803	4.777779	110899



Corry: (Pre-)Alignment

Goal: Position & orient detector *local* planes to *global* layout

Define modules, e.g.

- Prealignment: translational plane alignment (not rotations)
- AlignmentMillepede: implementation of the Millepede module

Reference detector flagged in geometry file using *role* parameter

Outputs

- Updated geometry file
 - Limit to what can be automated
 - Won't flip orientations
 - Can get stuck in local χ^2 minima

Corry: Track Finding

Goal: Build *global* track trajectory from *local* cluster positions in planes

Track Finding

Define modules, e.g.

- Tracking4D: using position and time from tracking planes
- TrackingMultiplet: build from upstream and downstream tracklets

Reference detector flagged in geometry file using *role* parameter

Then add Track Association

Define modules, e.g.

- DUTAssociation: establish association between DUT clusters and tracks

Corry: Track Finding II

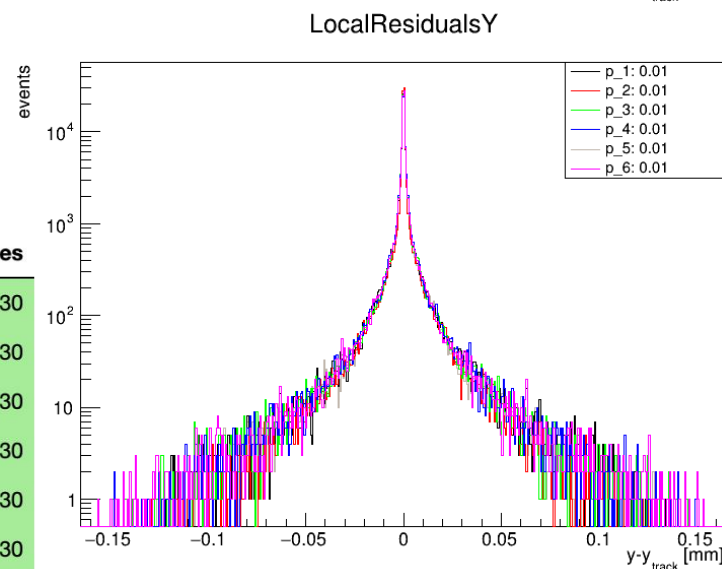
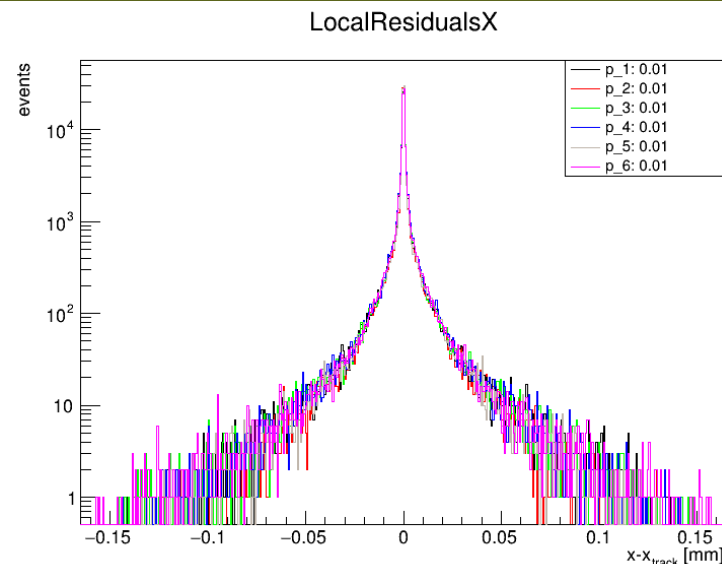
Telescope planes

Residuals are biased

- Planes are used in track fitting
→ $\sigma \ll \text{pitch}/\sqrt{12}$
- Distributions very close to zero

	name	max	mean	rms	entries
0	LocalResidualsX:1	27819	-0.000023	0.009582	95630
1	LocalResidualsX:2	30089	0.000019	0.008213	95630
2	LocalResidualsX:3	26889	0.000004	0.009773	95630
3	LocalResidualsX:4	26929	0.000027	0.009595	95630
4	LocalResidualsX:5	30065	-0.000017	0.008725	95630
5	LocalResidualsX:6	27966	-0.000010	0.010350	95630

	name	max	mean	rms	entries
0	LocalResidualsY:1	27576	0.000019	0.009786	95630
1	LocalResidualsY:2	30177	0.000016	0.008262	95630
2	LocalResidualsY:3	27040	-0.000044	0.009895	95630
3	LocalResidualsY:4	26812	0.000004	0.009839	95630
4	LocalResidualsY:5	30334	-0.000002	0.008773	95630
5	LocalResidualsY:6	27863	0.000006	0.010442	95630



Corry: Metrics

Goal: *Quantitative characterisation* of detector properties

Analysis examples:

- AnalysisDUT: generic analysis module for all types of detectors
 - Including residuals
- AnalysisEfficiency: comparing cluster positions with the interpolated track position
 - Including in-pixel efficiencies
- AnalysisTimingATLASpix: in-depth timing analysis of the ATLASpix
 - Specific to detector type

Corry: Metrics

Residuals

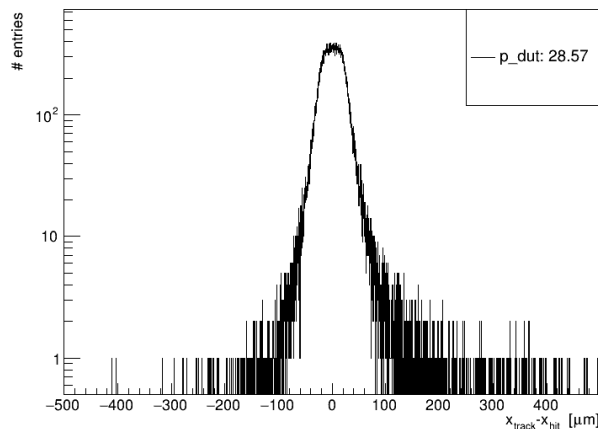
Lower population in DUT than in telescope planes (~95k)

- Matching not always successful (less than required planes hit)
- Detector not always efficient

Residuals much wider than telescope (X:~0.01, Y: ~0.01)

- DUT planes not used in track fitting (i.e. unbiased residuals)

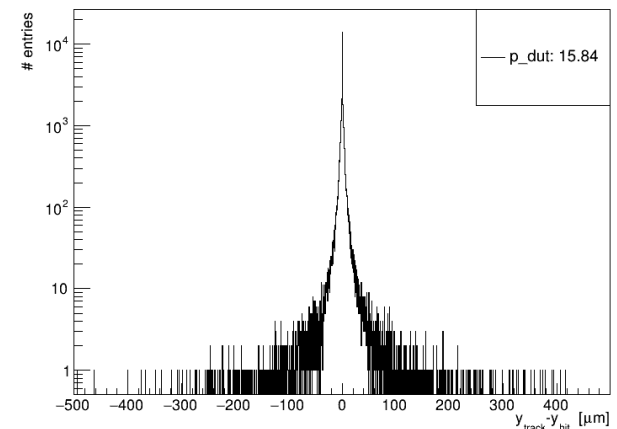
residualsX



	name	max	mean	rms	entries
0	residualsX:dut	390	2.168379	28.571923	83673

0	residualsX:dut	390	2.168379	28.571923	83673
---	----------------	-----	----------	-----------	-------

residualsY



	name	max	mean	rms	entries
0	residualsY:dut	14176	0.003974	15.838982	83673

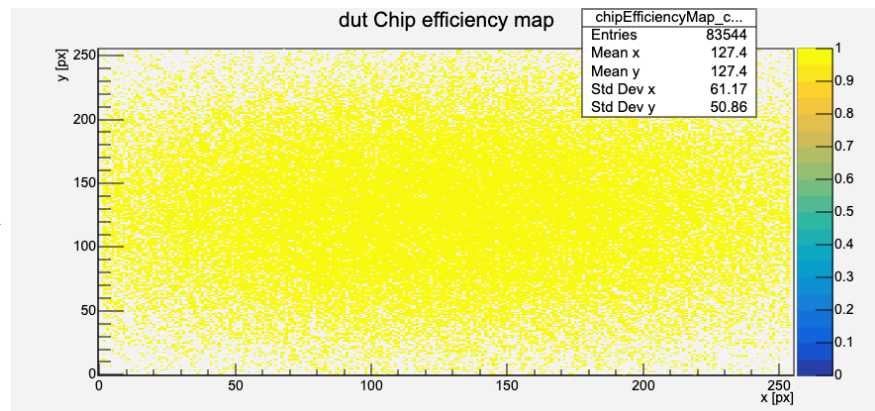
0	residualsY:dut	14176	0.003974	15.838982	83673
---	----------------	-------	----------	-----------	-------

Corry: Metrics II

Efficiency

Map of efficiency across pixel matrix

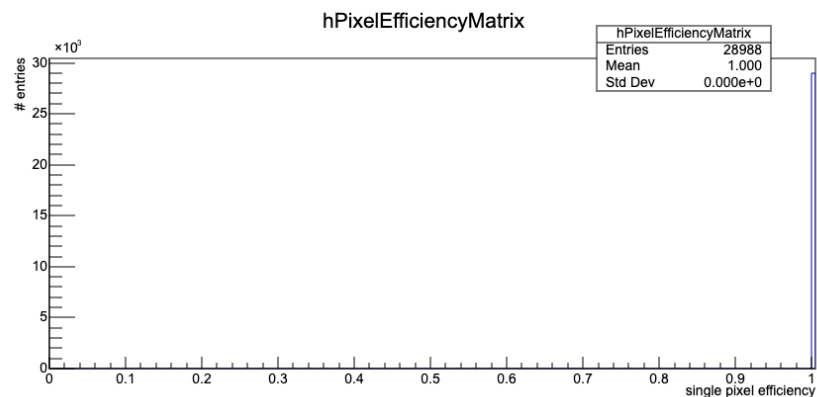
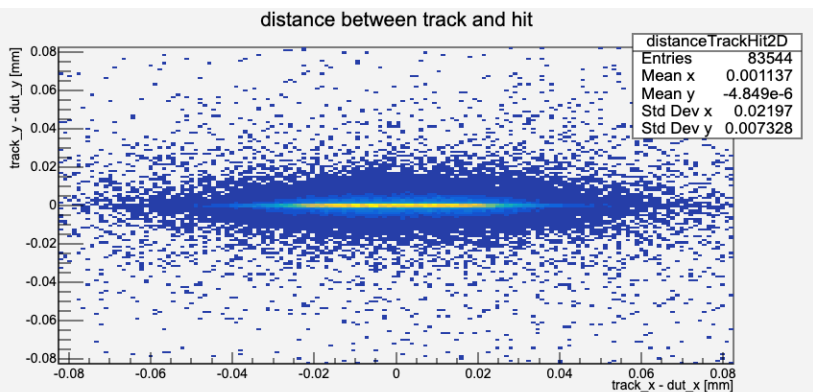
- Excellent efficiency in simulation
 - Uniform colour \rightarrow consistent matrix
 - High value \rightarrow efficient device



Distribution of pixel efficiencies

- Excellent efficiency in simulation
- Entries = number of pixels

Distance between track and hit \approx 2D residual plot

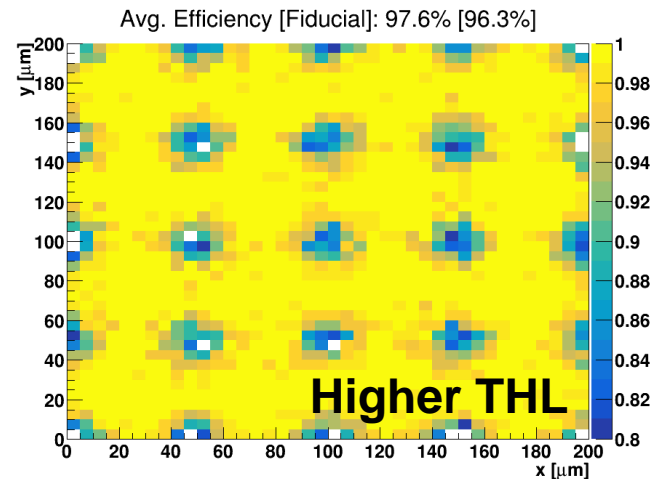
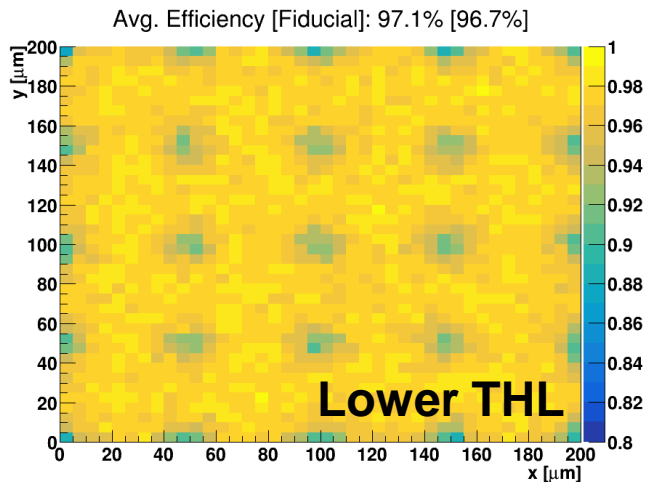
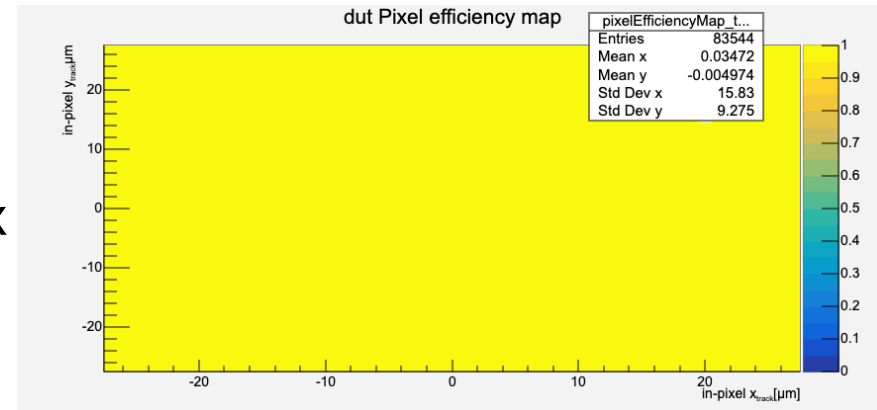


Corry: Metrics III

In-pixel efficiency

Map of efficiency across pixel matrix

- Excellent efficiency in simulation
 - Uniform colour → consistent matrix
 - High value → efficient device



Corry: Allpix**2

Allpix**2 generates corryvreckan output

- Module: CorryvreckanWriter - [documentation](#)

Corryvreckan can read in file

- Module: FileReader – [documentation](#)

Allpix**2 contains event and detectors hit info formation

- Replaces Metronome and EventLoader modules

More Generally...

Corryvreckan can also read in part completed analysis

- Don't need to run all steps in one go
 - Can focus on single step
 - Split alternative analysis options
 - e.g. clustering
 - binary/analog
 - Neighbour definition

Summary

Part I: Testbeam re-cap

- Testbeams offer unique tests based on control of radiation source
 - Complex logistics & data gathering
 - Complex data interpretation

Part II: Generic reconstruction

- Common reco. steps: Hits, Clusters, Correlations, Alignment, Tracking
- Qualitative characterisation via metrics via tracks comparison to DUT

Part III: Corryvreckan

- Modern, well-documented, free reconstruction framework
- Compatible with Allpix**2 simulation framework

Back-up

Setting up tutorial

Set-up repos

Clone repositories

Pick your favourite directory

- Clone corryvreckan repo.
 - git clone <https://gitlab.cern.ch/corryvreckan/corryvreckan.git>
- Go into repository
 - cd corryvreckan
- Build local docker image (optional)
 - docker build -f etc/docker/Dockerfile -t corry .

Back to your favourite directory

- Clone tutorial repo.
 - git clone https://gitlab.cern.ch/jekroege/fp_pixel_sensor_characterisation.git
- Go into repository
 - cd fp_pixel_sensor_characterisation/scripts/
- Download data
 - ./download_data.sh

Running corry

Run corryvreckan container

From top directory of corryvreckan repo.

- `docker run --interactive --tty --volume "$(pwd)":/data -v $(pwd)/../fp_pixel_sensor_characterisation/./examples --name=corryvreckan gitlab-registry.cern.ch/corryvreckan/corryvreckan bash`
- Running interactively with bash and local volumes mounted for output data and examples
- For Windows users: `$(pwd) → $PWD`
- If using (optional) local image:
`gitlab-registry.cern.ch/corryvreckan/corryvreckan → corry`

Run corryvreckan package

Inside container

- `corry -c /examples/01_example.conf`

You can find the output file (*01_example.root*) in the *output* directory

This can be viewed locally (i.e. outside container)

- `root 01_example.root`
- `TBrowser a`