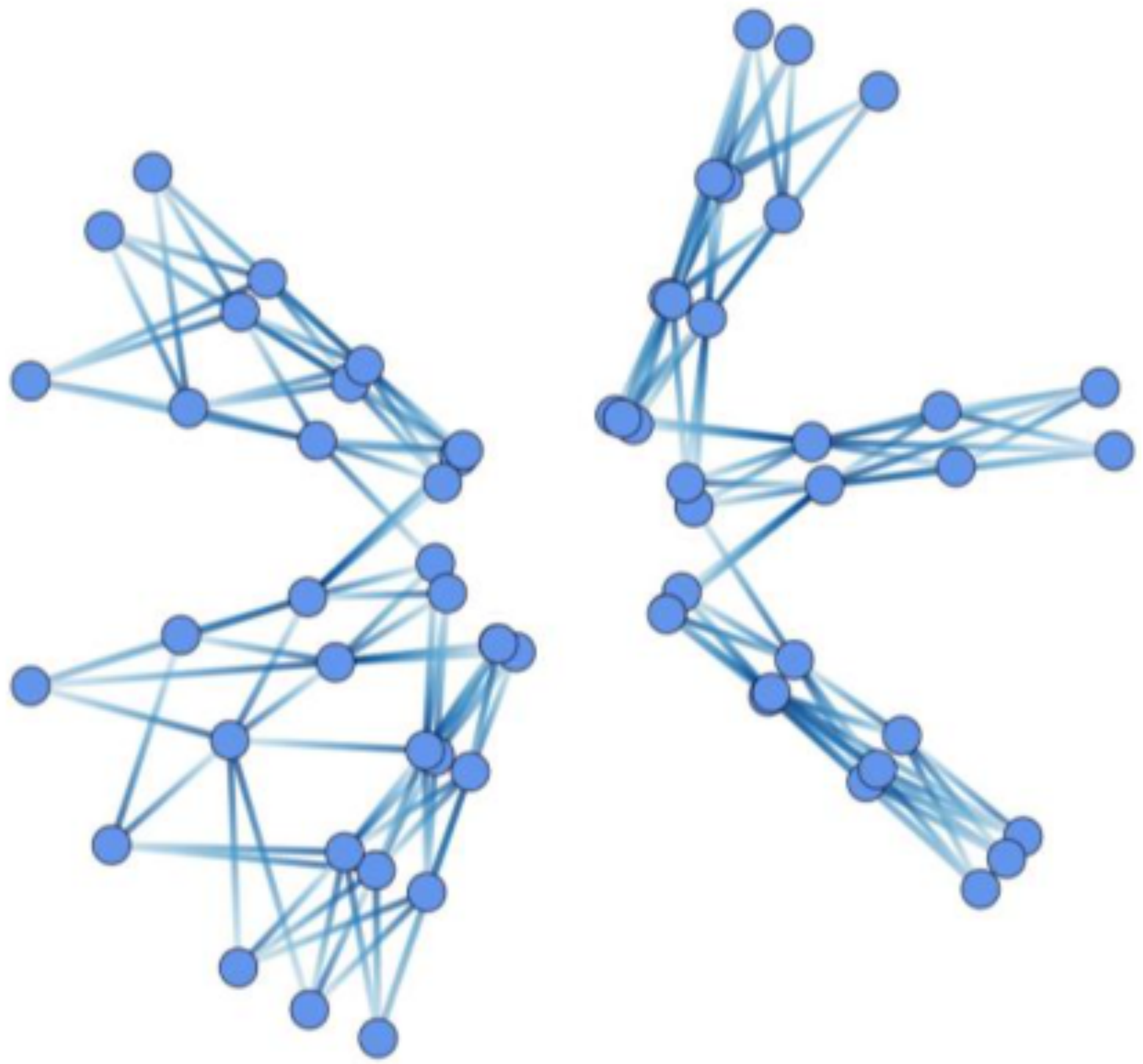
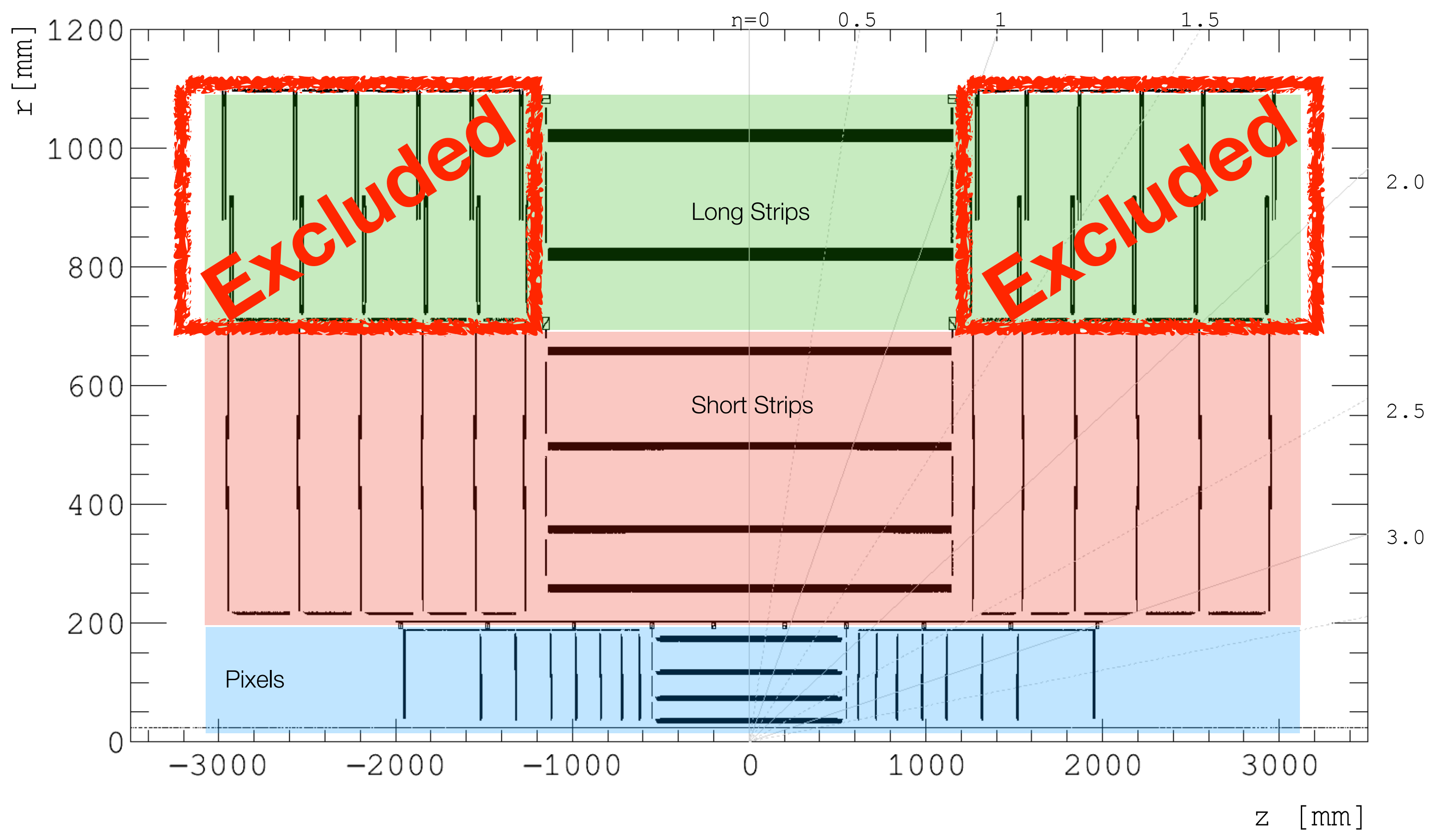
Energy-efficient graph-based algorithm for tracking at the HL-LHC



Heberth Torres (L2I Toulouse)
CHEP conference
24/10/2024

Introduction



- Target: Track finding: Identifying hits belonging to each track.
  (Fit to extract track physics parameters: standard χ^2 fit.)

- Status: Work in progress. Today showing preliminary versions of some parts.
  (Still need to propagate the barrel long strip treatment to the endcap long strips.)

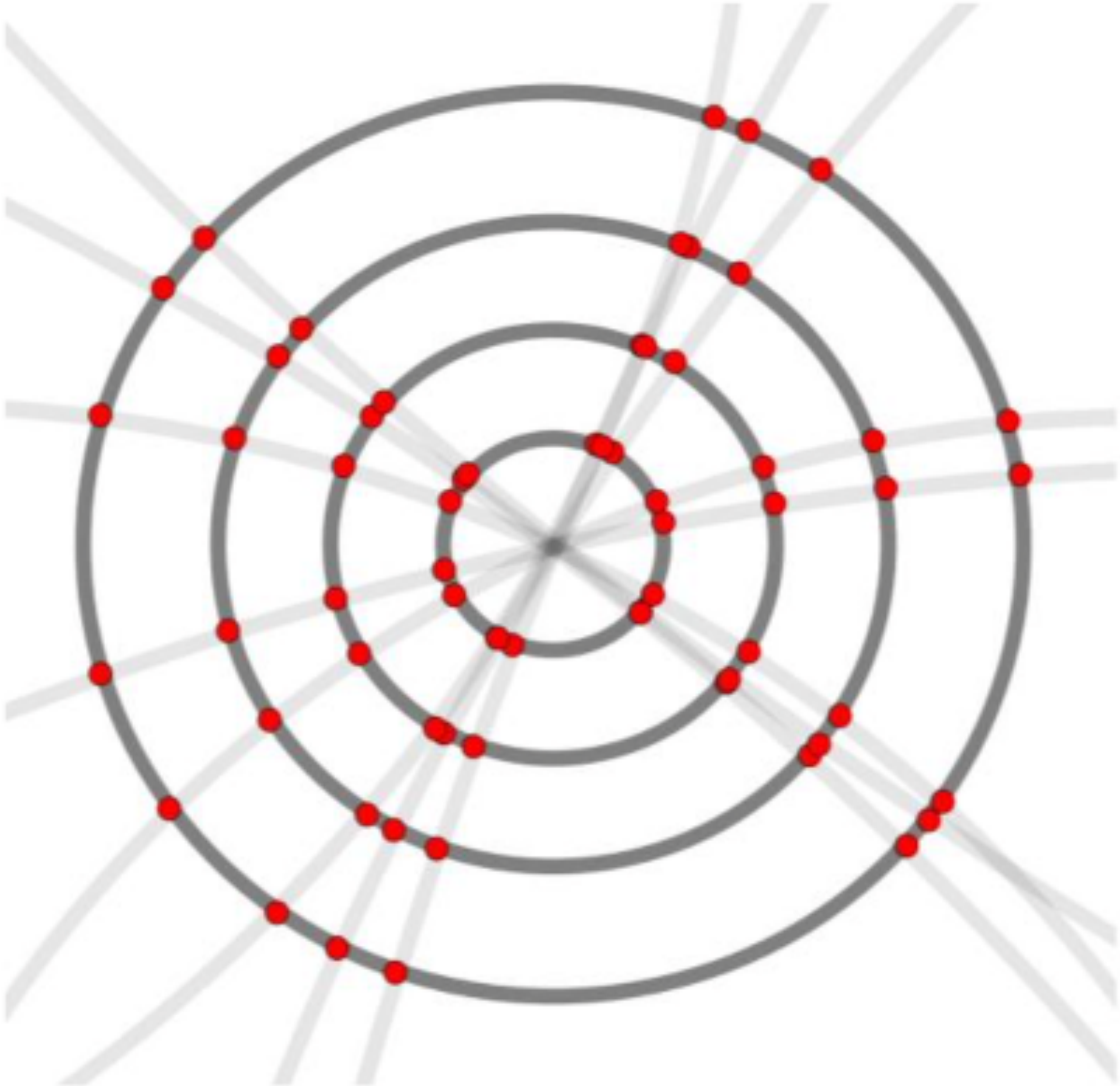
- Sample prepared with ACTS (v36.3.0): Pythia8 $t\bar{t}$ samples, $\langle \mu \rangle = 200$, OpenDataDetector with Geant4 sim.

- Working with space-points from ACTS,
  getting on average ~110K spacepoints per event (simplified setup compared e.g. to ATLAS ITk with 300K/evt.),
  currently excluding the endcap long strips (work in progress to take them into account).

- Target particles: Primary particles, $p_T > 1$ GeV, $|\eta| < 3$, at least 3 hits, excluding electrons.

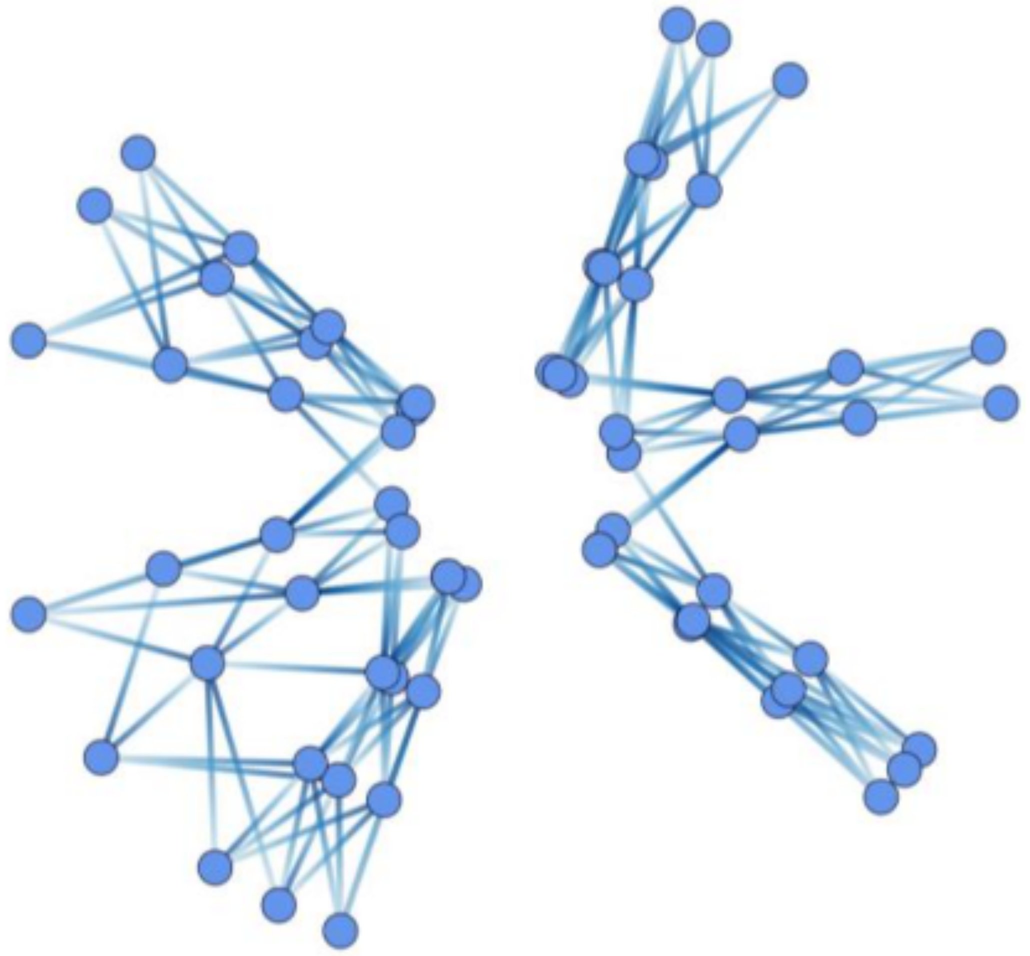
- Execution time in one CPU core: < 0.5 s (std. cluster CPU at CC-IN2P3-Lyon).
  [Quoted values estimated with one "Asimov event" (i.e. number of space-points = average = 110K)].
  Algorithm highly parallelizable for GPU, which should reduce time by factor > 10.

GNN4ITk graph definition



Hits

● Hit or space point in ITk



Graph

- **Graph:** Set of nodes and edges
- **Node:** Hit or space point
- **Edge:** Hypothesis: The two associated nodes represent two successive **hits of the same particle**

Figures from D. Murnane

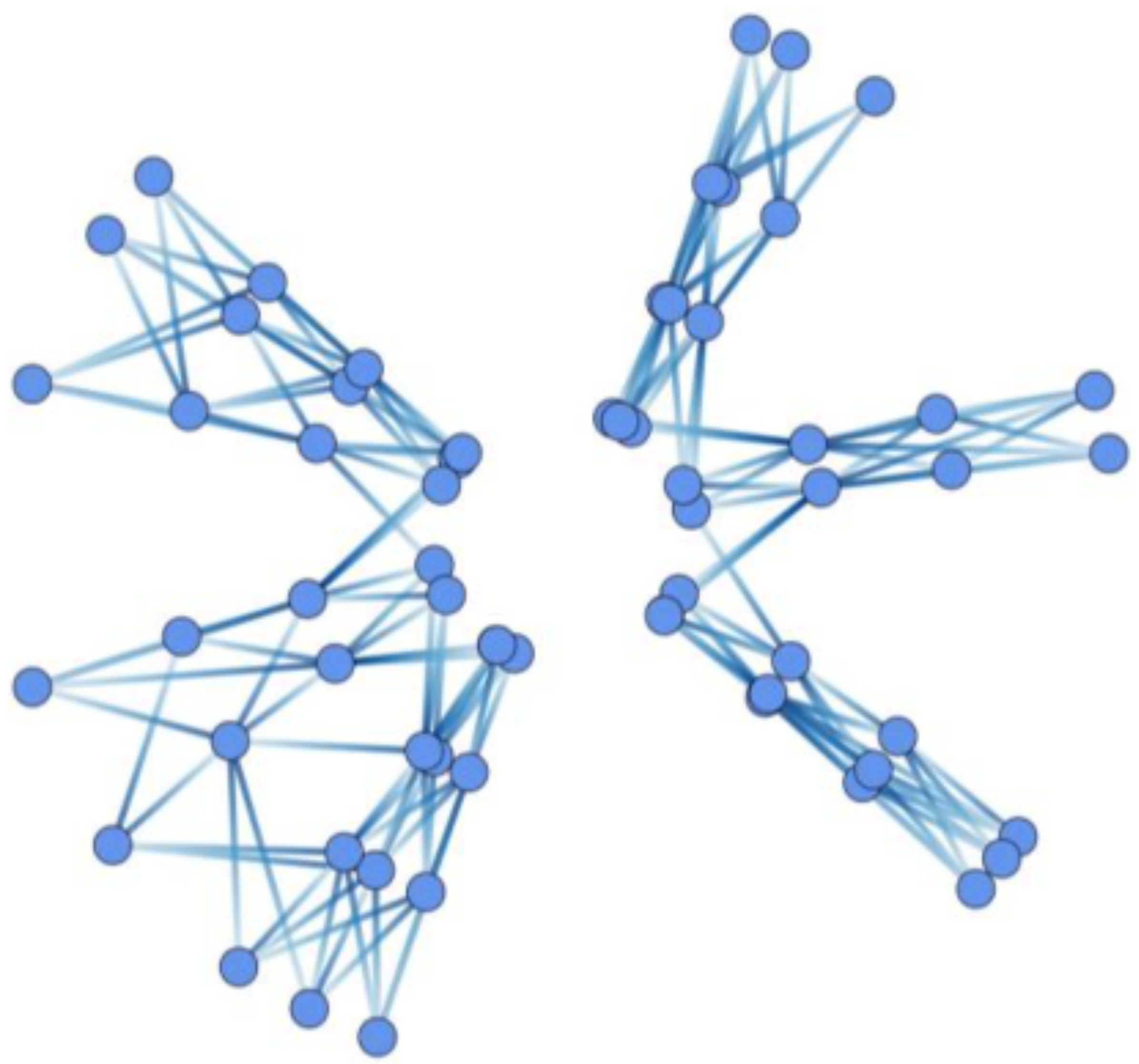
Algorithm overview

1. Graph construction
2. Refinement of strip edges
3. Triplet construction
4. Graph segmentation

    Output: Loose proto-tracks with high hit efficiency

5. Final refinement step, still to add

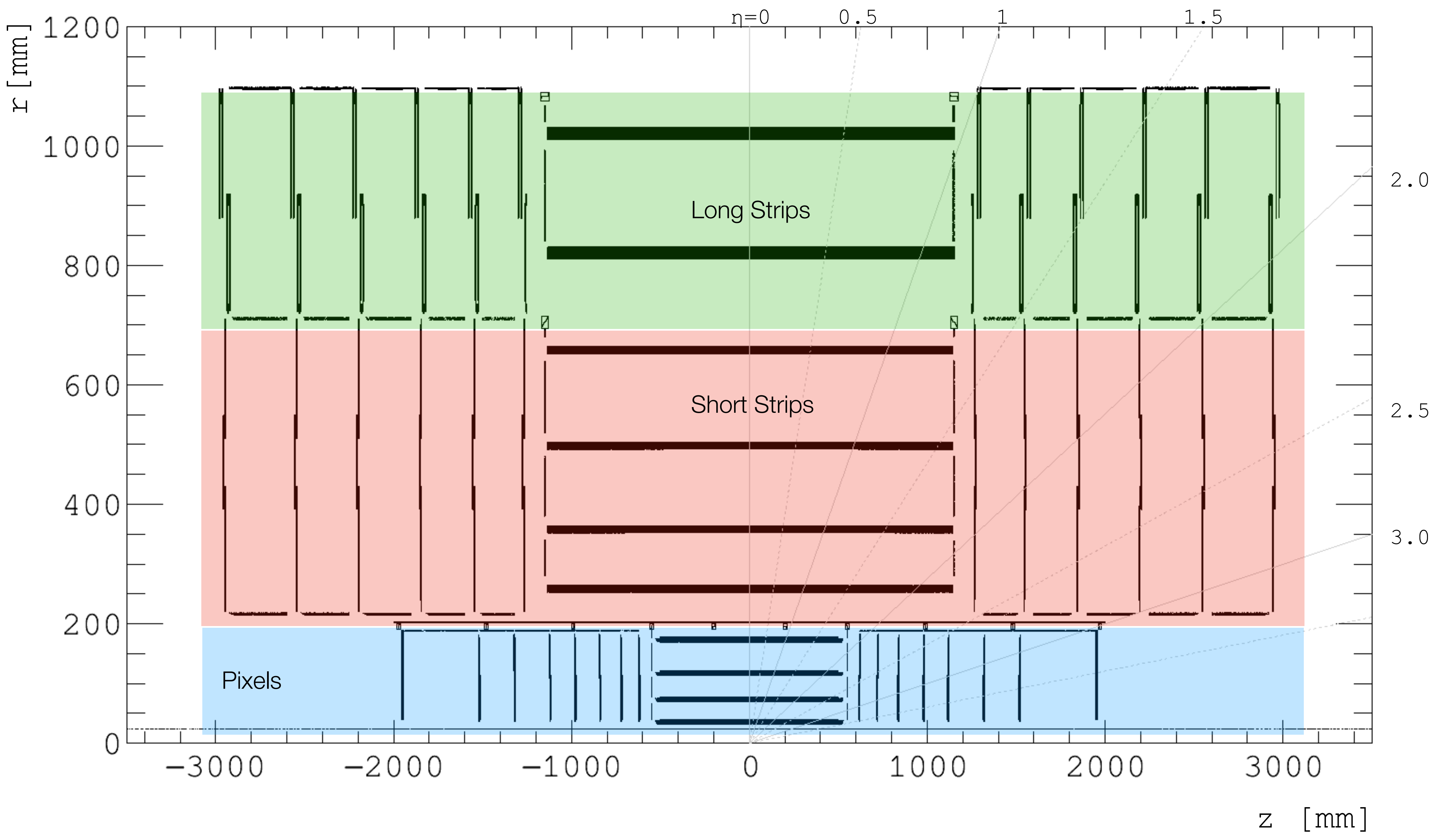
    Either a GNN, or removing outlier with χ^2 fit, or …



1. Graph construction

2D (r, z) Module Map + $\Delta\phi$ cut

Modified version of the Module Map
(C. Biscarat et al., C. Rougier et al.)



OpenDataDetector

- **2D Module Map:** Omitting the ϕ coordinate, built a lookup table of possible "module ring" pair connections using MC sample.
  Have ~270 modules rings and ~1000 connections.

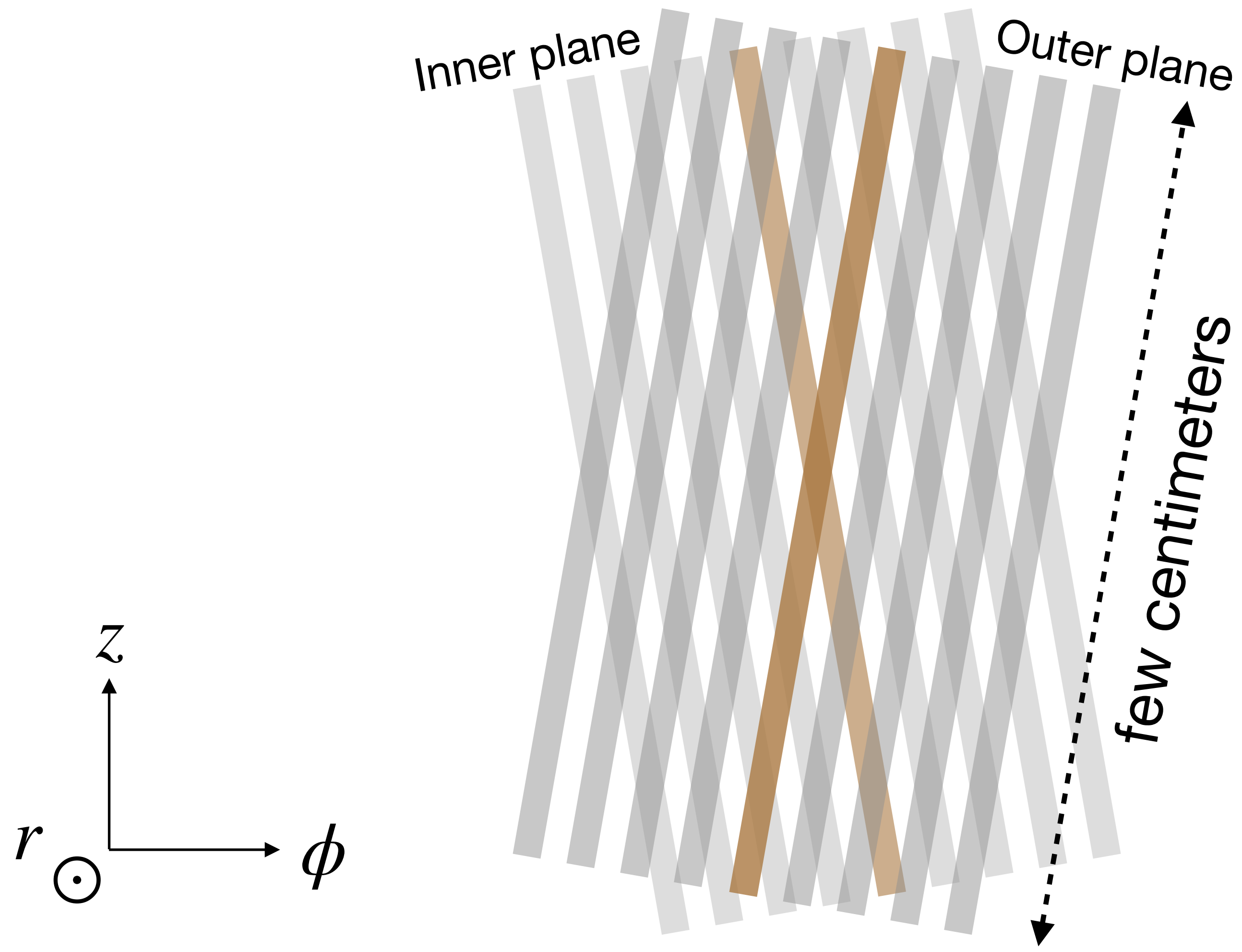
- Graph construction:
  - Build edges (hit pairs) based on 2D MM,
  - considering only hit pairs within a $\Delta\phi$ window
  - and apply a z_0 cut.

- Advantages: For MM training, enhances MC statistics by a factor equal to number of ϕ modules per ring, speed up production using directly hit ϕ instead of module granularity.

- Execution time: 210 ms (graph construction + strip edge treatment).
  Algorithm speed up: First organize hits on groups per module and consider only relevant group pairs.
  Hits are ϕ-sorted per group, which is time convenient for the $\Delta\phi$ window cut.

2. Strip edge refinement

Calculation of strip-hit position



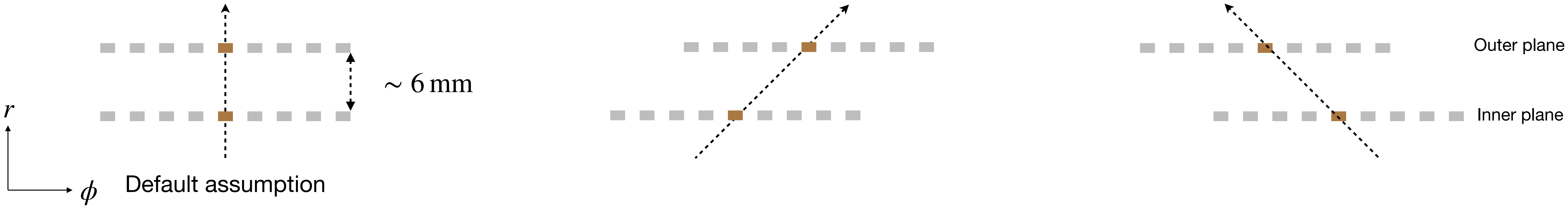
Double strip sensor planes in barrel module

Two strips fired by a particle in brown

Where did the particle hit the inner plane?

Note: This strip edge refinement:
- Barrel done ✓
- Endcap still to do

Some posible options:



Default hits:
Poor σ_z ~ centimeters

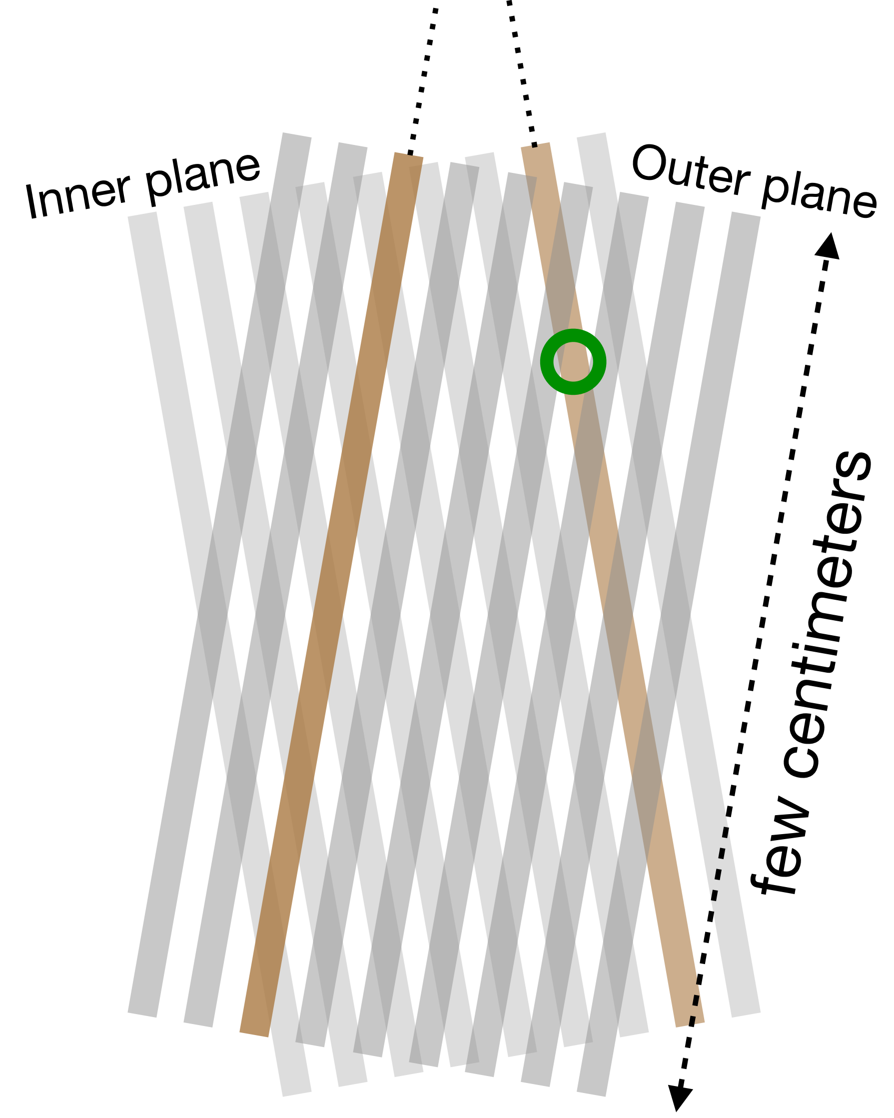
➡

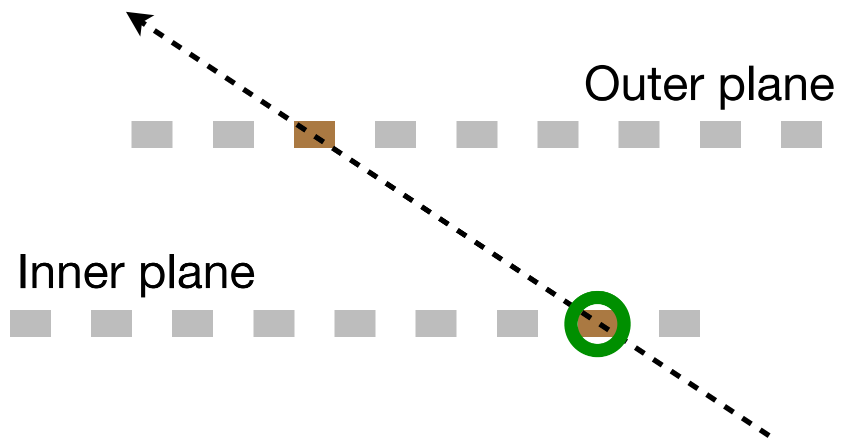
Use hit pair info to estimate the **particle's direction** when traversing the strip planes

➡

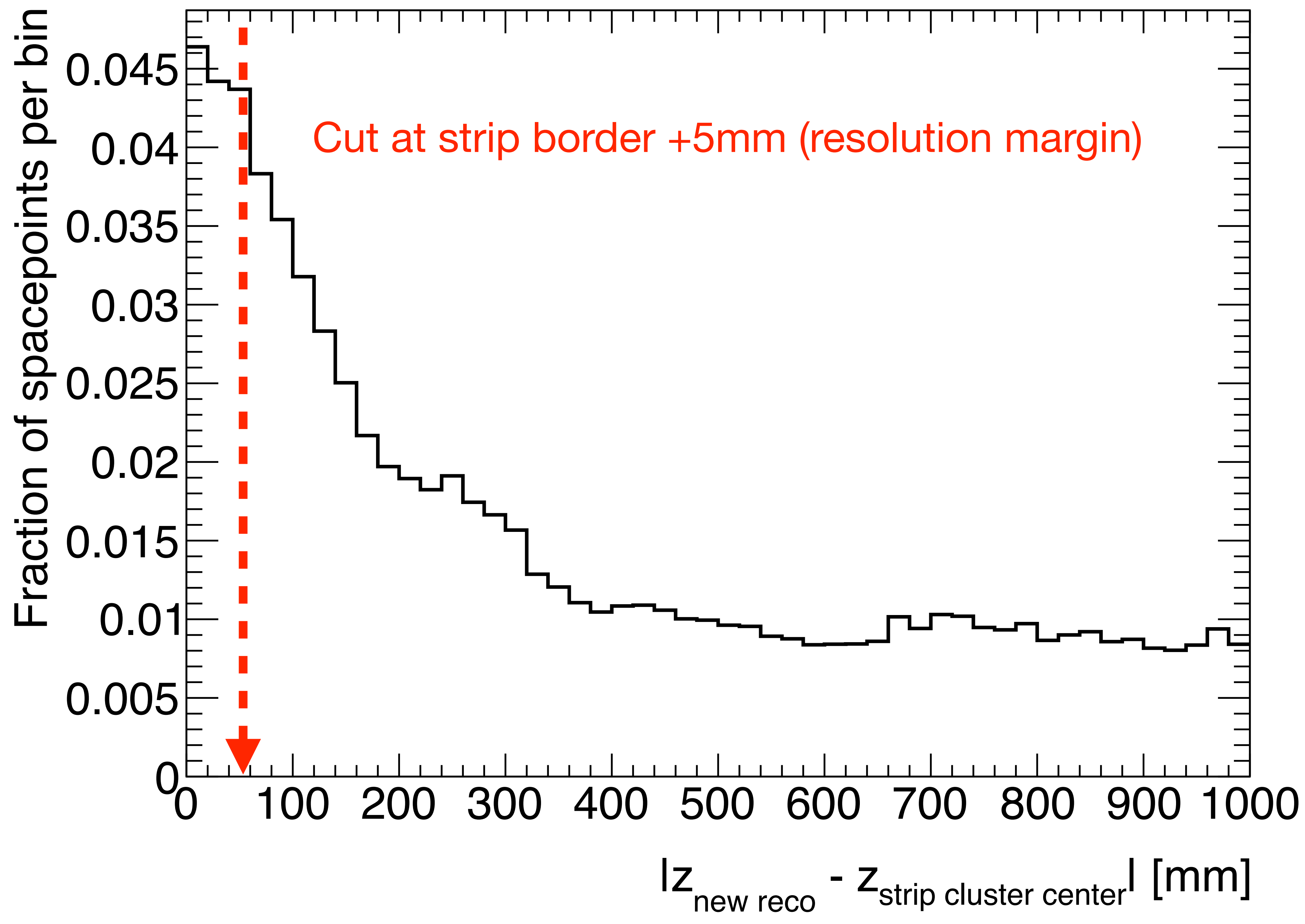
New hits re-calculated taking into account particle's direction
→ Improve to σ_z ~ 2 mm

2. Strip edge refinement





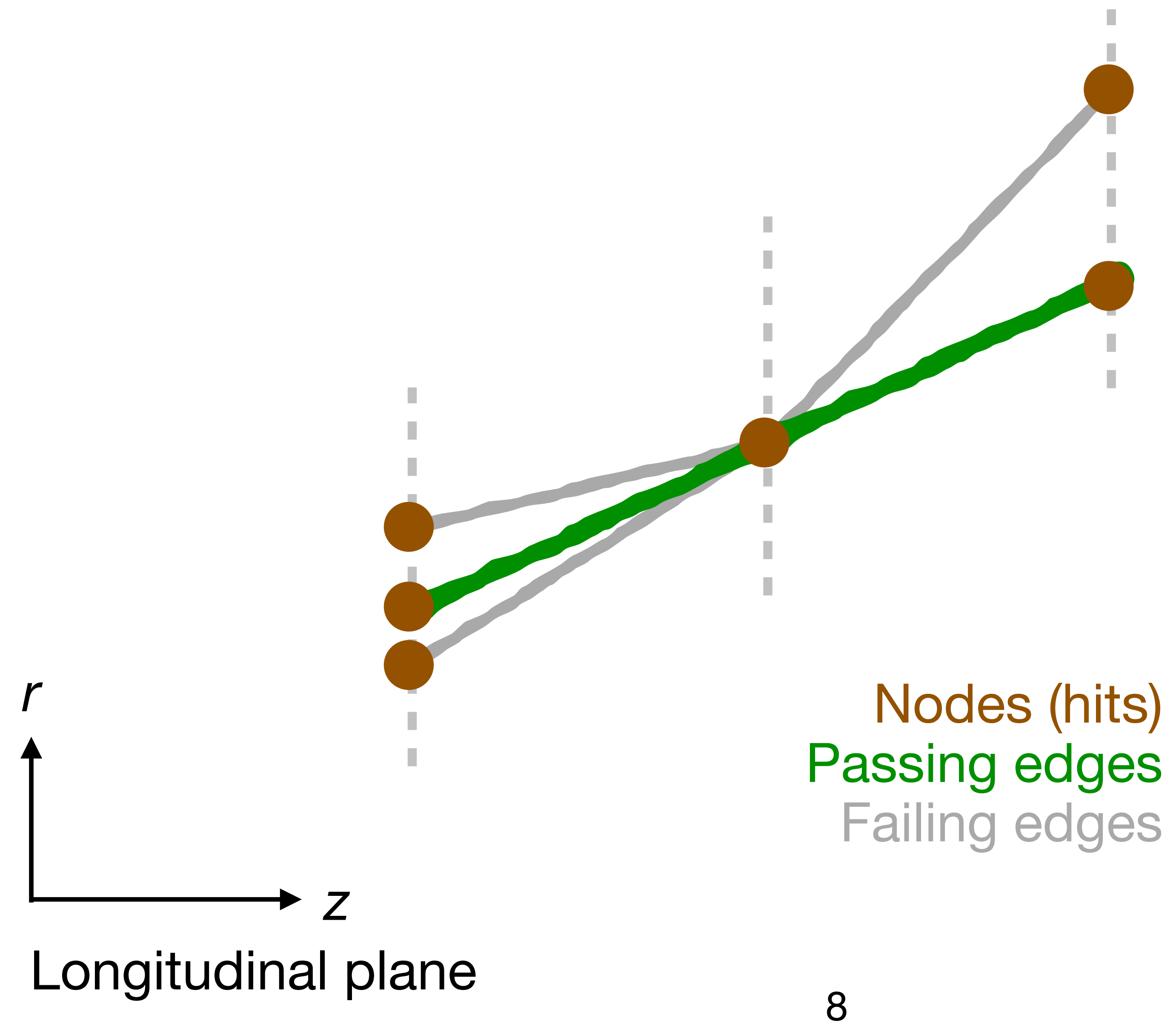
Calculated strip spacepoints
for all considered strip edges (dominated by fakes):
Randomly wrong direction → nonsensical spacepoints out of strip length



Removal of inconsistent strip edges by requesting
$$\left|z^{\text{hit}}_{\text{rel.}}\right| < z^{\text{strip border}}_{\text{rel.}} + 5\text{mm}$$
removes ~80% of fake strip edges,
with true edge inefficiency < 2‰

Execution time: 210 ms
(graph construction + strip edge treatment).

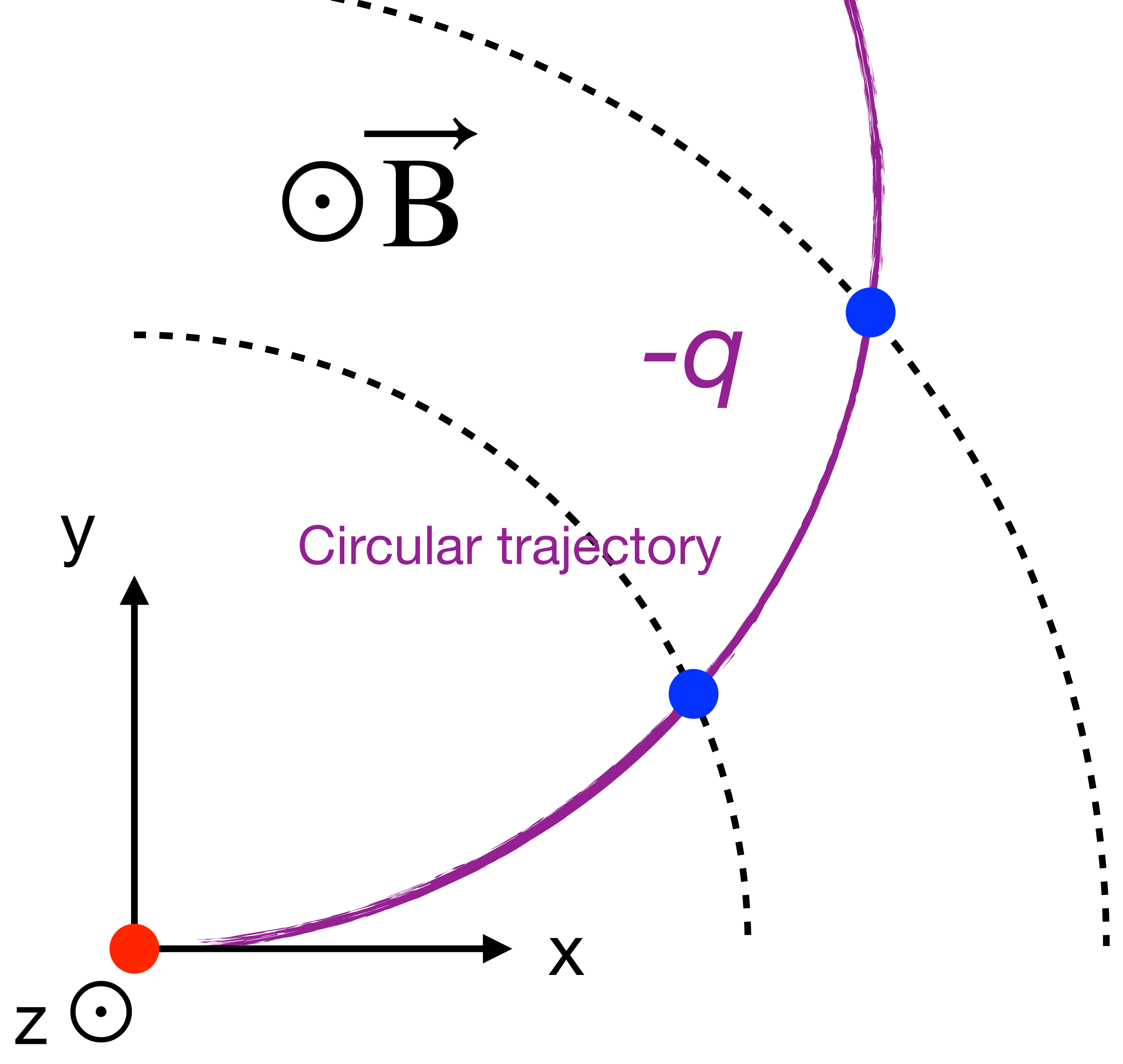
3. Triplet construction



- At each node, compare each incoming with each outgoing edge. If they are compatible, build up a hit triplet.
- Edge compatibility tested based on two edge features: η direction and an estimator of q/p_T (see next slide).
- For each MM module pair σ_η and σ_{q/p_T} are pre-estimated, as well as a calibration factor for q/p_T to take into account the magnetic field inhomogeneity.
- For each pair of edges, compute $\chi_i = \Delta x_i/\sigma_i$ (with $i = \eta, q/p_T$).
- Build a triplet if $\chi^2 = \chi^2_\eta + \chi^2_{q/p_T} < \chi^2_\mathrm{cut}$. Edges not part of any triplet are discarded.
- Execution time: 130 ms

3. Triplet construction

q/p_T



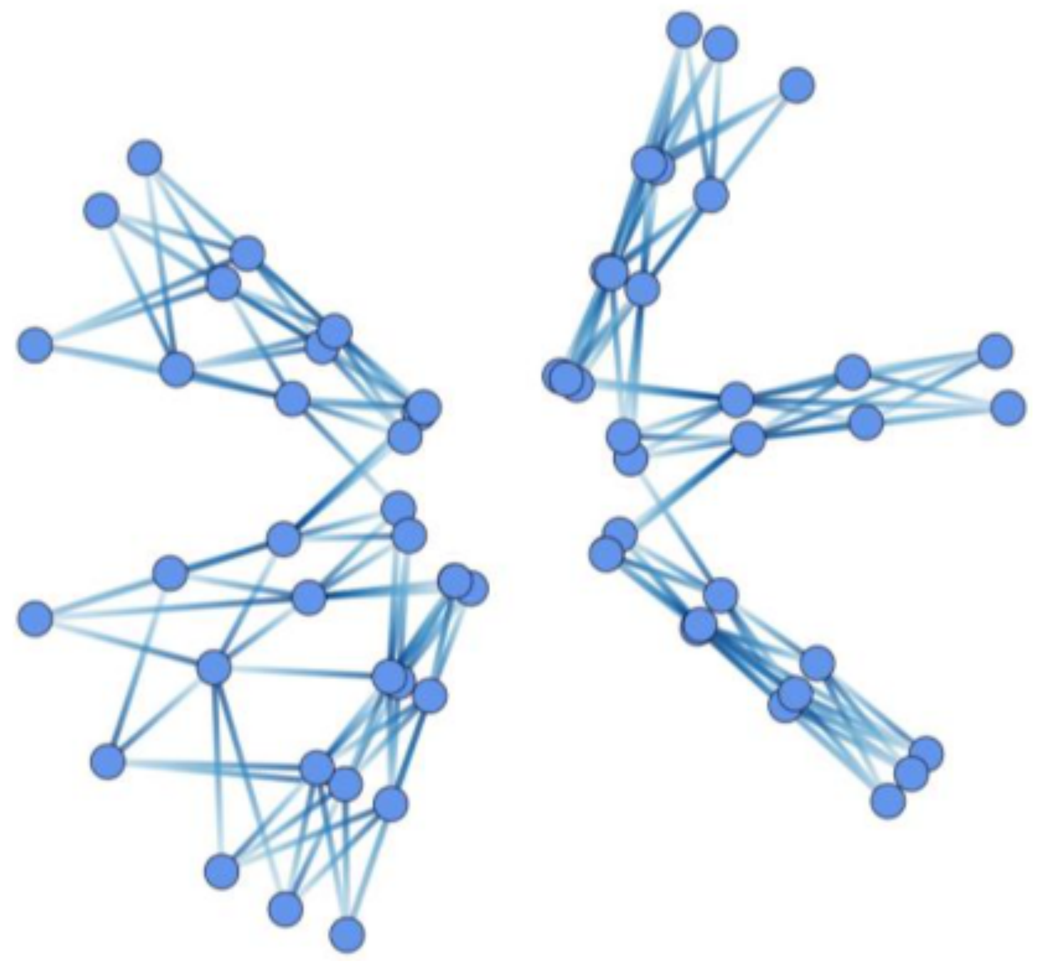
- For each edge, assuming the particles to have $d_0 = 0$ gives us a 3rd space point in addition to the hit pair, a triplet.
- With a triplet and assuming an homogeneous magnetic field (circular trajectory in the transverse plane), we get

$$q/p_T = -\frac{\sin \Delta\phi}{0.3\, d_T},$$

  where d_T : hit separation in the transverse plane.

- A calibration factor is applied to take into account the inhomogeneous magnetic field.
- The actual d_0 distribution of the target particles is taken into account by the uncertainty σ_{q/p_T} , specially for small *r* values.

4. Graph segmentation



Initial graph definition

- **Graph:** Set of nodes and edges
- **Node:** Hit or space point
- **Edge:** Hypothesis: The two associated nodes represent two successive **hits of the same particle**

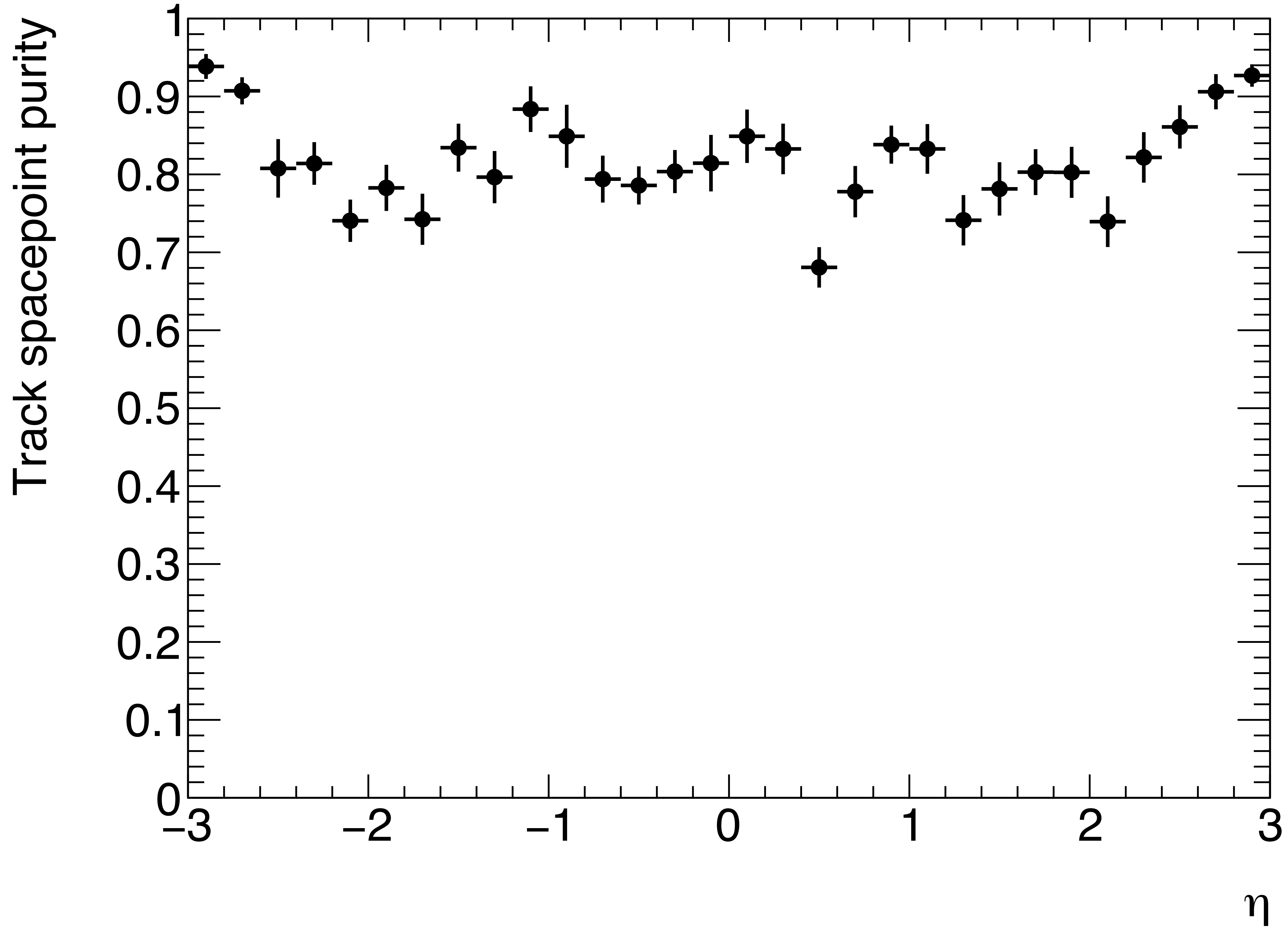
- Change of graph definition:
  - **Node**: Hit pair (previous edges).
  - **Edge**: Hit triplet, involving two hit pairs.
- A Connected Component algorithm is applied (Z. Zhang's algorithm).
- Each group of connected hit pairs represents a proto-track, which includes all hits involved in the pairs.
- Note: Each individual hit can belong to more than one proto-track.
- Execution time: 30 ms

Performance result example

for these loose proto-tracks

- For a triplet cut $\chi^2_{\mathrm{triplet}} < 9$
  (χ^2 with ndf = 2 tail prob. = 1%)

- Spacepoint purity vs. η for standard matching (> 50% purity) tracks



- Tracking efficiency for std. matching: 99%.

- Requiring 100% spacepoint efficiency, tracking efficiency: 93%.

Summary

- Have an energy-efficient graph-based algorithm for track finding.
  Takes **370 ms** in one CPU core for this ODD sample, and can be easily parallelized for GPU.

- To do list:
  - Include the endcap long strips (same method as for barrel).
  - Test different options for the final proto-track purity refinement step, for example:
    - Feed output graphs into a GNN, either as one graph per event or as proto-track mini-graphs.
    - Or feed those loose proto-tracks into the χ^2 fit and remove outliers, or …
  - Check computing and physics performance with an ATLAS ITk sample (a more realistic sample).
  - Plan to implement this algorithm in ACTS, to make it available to different tracking chains.