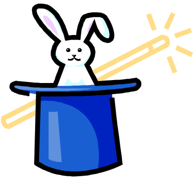
Marlin-based Algorithm for Geometry-Independent Clustering

MAGIC : v01-03



Chris Ainsley (presented by David Ward)

UNIVERSITY OF CAMBRIDGE

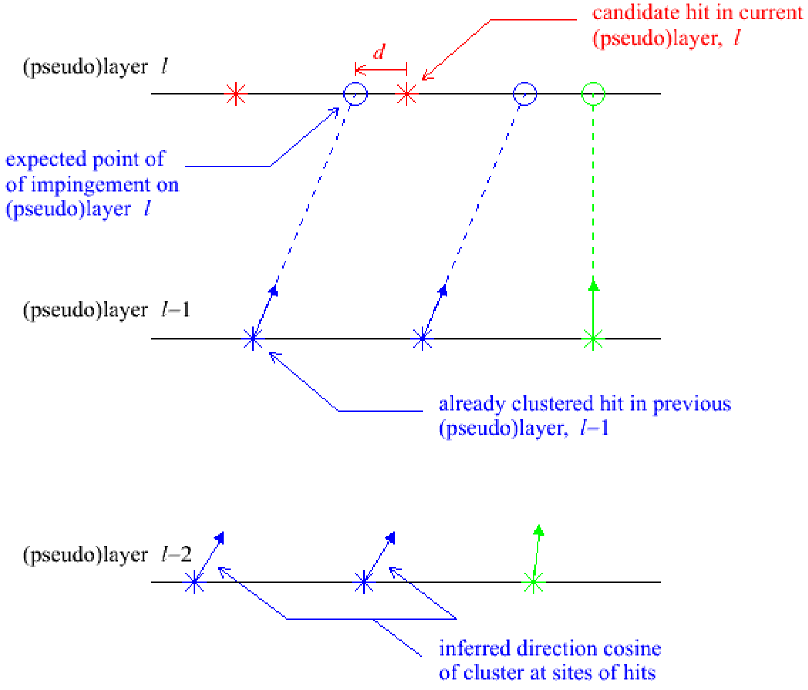
3rd ECFA ILC Workshop
14-17 November 2005, Vienna, Austria

Order of service

- Clustering with *MAGIC* in 4 steps.
- User-controlled steering with MARLIN.
- Charged/neutral shower separation studies.
- Clustering with the CALICE Ecal prototype data.
- Summary.

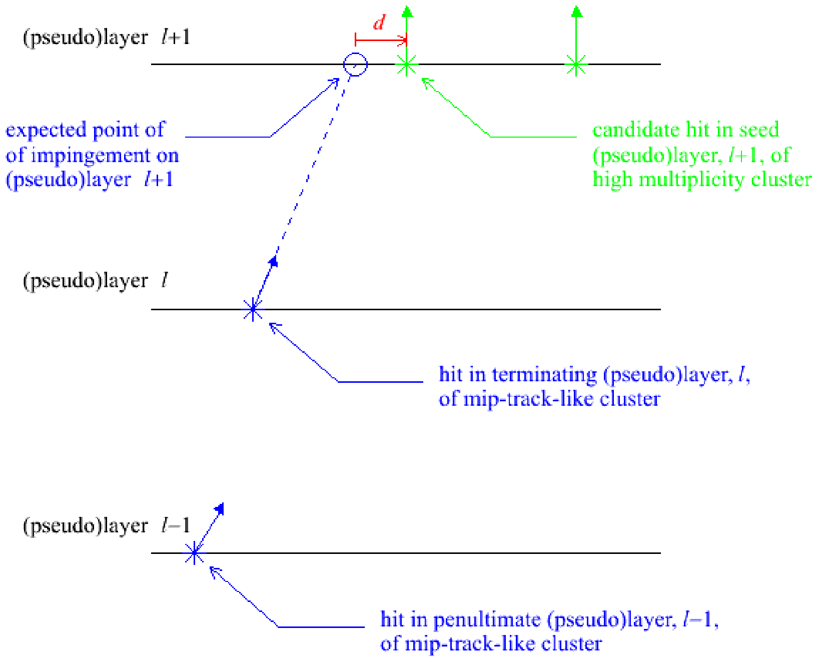
Clustering with MAGIC: stage 1

- Form coarse clusters by ***tracking*** closely-related hits ***layer-by-layer*** through the calorimeter:
  - for a candidate hit in a given layer, l, minimise the distance, ***d***, w.r.t all (already clustered) hits in layer l-1;
  - if ***d*** < **`distMax`** for minimum ***d***, assign candidate hit to same cluster as hit in layer l-1 which yields minimum;
  - if not, repeat with all hits in layer l-2, then, if necessary, layer l-3, *etc*., right through to layer l-**`layersToTrackBack`**;
  - after iterating over all hits in layer l, seed new clusters with those still unassigned, grouping those within **`proxSeedMax`** of hit of highest remaining density into same seed;
  - assign a direction cosine to each layer l hit:
    - if in Ecal, calculate density-weighted centre of each cluster's hits in layer l; assign a direction cosine to each hit along the line joining its cluster's centre in the seed layer (or (**`iPx`**, **`iPy`**, **`iPz`**) if it's a seed) to its cluster's centre in layer l;
    - if in Hcal, assign a direction cosine to each hit along the line from the hit to which each is linked (or (**`iPx`**, **`iPy`**, **`iPz`**) if it's a seed) to the hit itself;
  - iterate outwards through layers.


candidate hit in current (pseudo)layer, l
d
(pseudo)layer l
expected point of impingement on (pseudo)layer l
(pseudo)layer $l-1$
already clustered hit in previous (pseudo)layer, $l-1$
(pseudo)layer $l-2$
inferred direction cosine of cluster at sites of hits

Clustering with MAGIC: stage 2

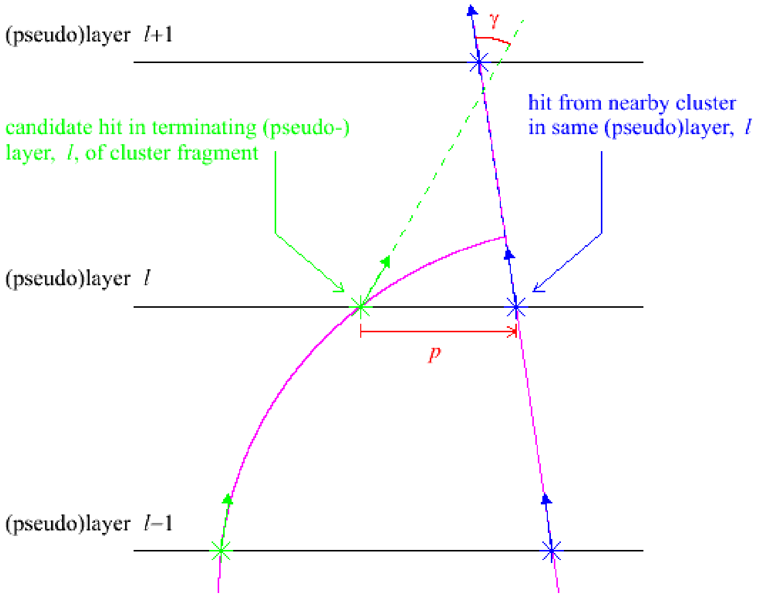
- Try to merge mip-track-like clusters (mean hit-number per layer < `hitsPerLayerMax` and hit multiplicity > `clusterSizeMin`), which terminate prematurely inside the calorimeters, with later-seeded clusters:
  - for each hit in the terminating layer, l, of a candidate mip-track-like cluster, minimise the distance, d, w.r.t. all hits in the seed layer of the later-seeded cluster of earliest seed layer, if its hit multiplicity > `clusterSizeMin`;
  - if d < `distMax` for minimum d, merge the clusters containing the respective hits into one;
  - if not, repeat with all hits in layer l-1 of the terminating cluster, then, if necessary, layer l-2, *etc.*, right through to l-`layersToTrackBack`;
  - if still not, repeat above steps with the later-seeded cluster of next earliest seed layer, *etc.* ;
  - iterate over clusters.



Clustering with MAGIC: stage 3

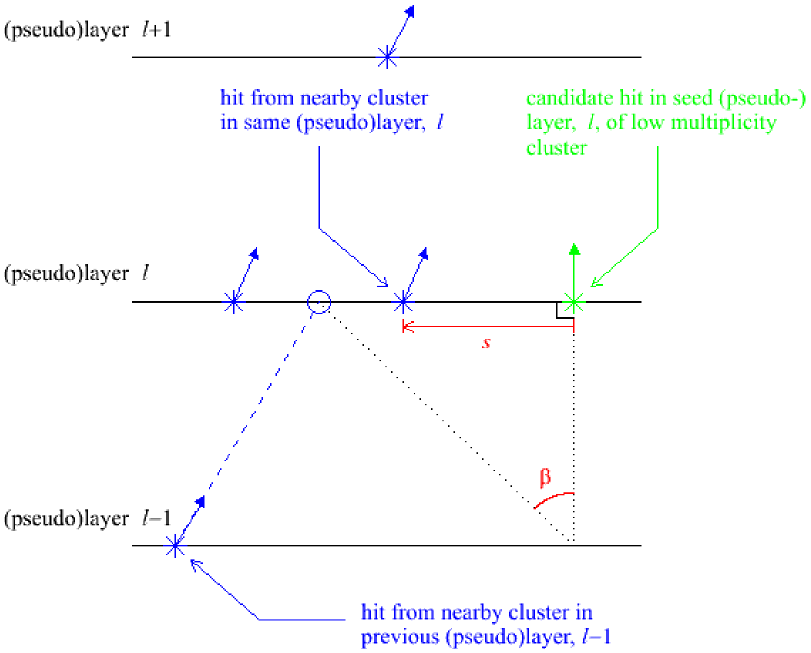
- Try to merge backward-spiralling track-like cluster-fragments with the forward propagating clusters to which they belong:
  - for each hit in the terminating layer, ***l***, of a candidate cluster fragment, calculate the distance, ***p***, to each hit in nearby clusters in the same layer, and the angle, γ, between their direction cosines;
  - loop over all pairs of hits;
  - if, for any pair, both:
    - ***p*** < **proxMergeMax** *and*
    - **cos γ < cosGammaMax**

    are satisfied, merge clusters together into one;
  - iterate over clusters.


(pseudo)layer l+1
γ
hit from nearby cluster in same (pseudo)layer, l
candidate hit in terminating (pseudo-) layer, l, of cluster fragment
(pseudo)layer l
p
(pseudo)layer l−1

Clustering with MAGIC: stage 4

- Try to merge low multiplicity cluster "halos" (hit multiplicity < **clusterSizeMin**) which just fail the stage 1 cluster-continuation cuts:
  - for the hit of highest density in the seed layer, l, of a low multiplicity cluster, minimise the angle, β, w.r.t all hits in layer l–1;
  - if **tan** β < **tanBetaMax** for minimum β, merge the clusters containing the repsective hits into one;
  - if not, repeat with all hits in layer l–2, then, if necessary, layer l–3, *etc.*, right through to layer l–**layersToTrackBack**;
  - if still not, repeat above steps with the candidate hit in the seed layer of the low multiplicity cluster of next highest density, *etc.* ;
  - if still not, merge the low multiplicity cluster into the nearest cluster with hits in the same layer as the low multiplicity cluster's seed layer, provided the two clusters contain hits separated by s < **proxMergeMax**;
  - iterate over clusters.


(pseudo)layer l+1
hit from nearby cluster in same (pseudo)layer, l
candidate hit in seed (pseudo-) layer, l, of low multiplicity cluster
(pseudo)layer l
s
β
(pseudo)layer l−1
hit from nearby cluster in previous (pseudo)layer, l−1

User-controlled steering with MARLIN (1)

- Code structured as a series of **MARLIN** "***processors***", together with a **steering file**: `cluster.steer` (read at ***run-time***).
- Reads hits collections from **LCIO** file, adds **LCIO** clusters collections (essentially pointers back to component hits) and writes everything to new **LCIO** output file.
- **Detector parameters** and **clustering cuts** set in `cluster.steer` (*e.g.* based on CALICE design):

```
ProcessorType CalorimeterConfigurer
     detectorType             full        # "full" => barrel+endcaps; "prototype" => layers perp'r to +z
     iPx                      0.          # x-coordinate of interaction point (in mm)
     iPy                      0.          # y-coordinate of interaction point (in mm)
     iPz                      0.          # z-coordinate of interaction point (in mm)
     ecalLayers               40          # number of Ecal layers
     hcalLayers               40          # number of Hcal layers
     barrelSymmetry           8           # degree of rotational symmetry of barrel
     phi_1                    90.0        # phi offset of barrel stave 1 w.r.t. x-axis (in deg)

ProcessorType CalorimeterHitSetter
     ecalMip                  0.000150    # Ecal MIP energy (in GeV)
     hcalMip                  0.0000004   # Hcal MIP energy (in GeV)
     ecalMipThreshold         0.3333333   # Ecal hit-energy threshold (in MIP units)
     hcalMipThreshold         0.3333333   # Hcal hit-energy threshold (in MIP units)
```

User-controlled steering with MARLIN (2)

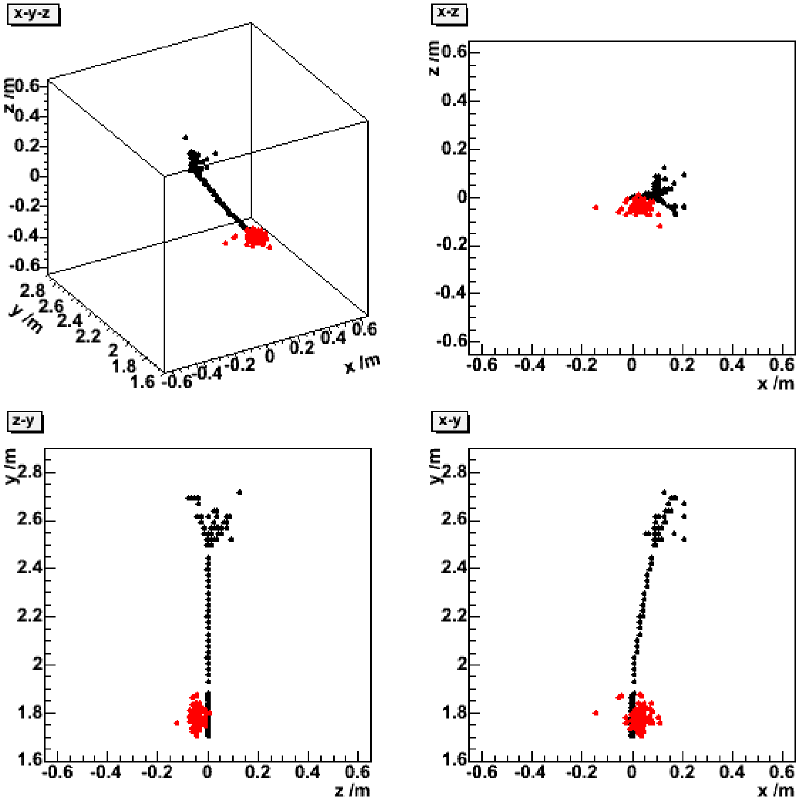
```
ProcessorType CalorimeterStage1Clusterer
     layersToTrackBack_ecal     3           # number of layers to track back in Ecal
     layersToTrackBack_hcal     3           # number of layers to track back in Hcal
     distMax_ecal               20.0        # distance cut in Ecal (in mm)
     distMax_hcal               30.0        # distance cut in Hcal (in mm)
     proxSeedMax_ecal           14.0        # maximum cluster-seed radius in Ecal (in mm)
     proxSeedMax_hcal           50.0        # maximum cluster-seed radius in Hcal (in mm)

ProcessorType CalorimeterStage2Clusterer
     clusterSizeMin             10          # minimum cluster size for consideration as a mip track
     hitsPerLayerMax            1.4         # maximum hit-number-per-layer for consideration as a mip track
     layersToTrackBack_ecal     5           # number of layers to track back in Ecal for merging
     layersToTrackBack_hcal     5           # number of layers to track back in Hcal for merging
     distMax_ecal               30.0        # Ecal distance cut for cluster merging (in mm)
     distMax_hcal               90.0        # Hcal distance cut for cluster merging (in mm)

ProcessorType CalorimeterStage3Clusterer
     proxMergeMax_ecal          20.0        # Ecal proximity cut for cluster merging (in mm)
     proxMergeMax_hcal          30.0        # Hcal proximity cut for cluster merging (in mm)
     cosGammaMax                0.5         # angular cut for cluster merging

ProcessorType CalorimeterStage4Clusterer
     clusterSizeMin             10          # minimum cluster size to avert potential merging
     layersToTrackBack_ecal     39          # number of layers to track back in Ecal for merging
     layersToTrackBack_hcal     79          # number of layers to track back in Hcal for merging
     tanBetaMax                 6.0         # angular cut for cluster merging
     proxSeedMax_ecal           400.0       # Ecal proximity cut for cluster merging (in mm)
     proxSeedMax_hcal           400.0       # Hcal proximity cut for cluster merging (in mm)
```

Charged/neutral shower separation studies

- Fire nearby charged/neutral particles into calorimeter.
- Perform standalone clustering on calorimeter hits with *MAGIC*.
- Extrapolate helix from charged track through calorimeters.
- Associate clusters/cluster fragments with charged particle if seeded within pad-size (= 1 cm) of projected helical trajectory.
- Remove corresponding calorimeter hits from further consideration; assume remainder to be the neutral shower.
- Apply energy calibration to leftover hits to reconstruct neutral particle energy.

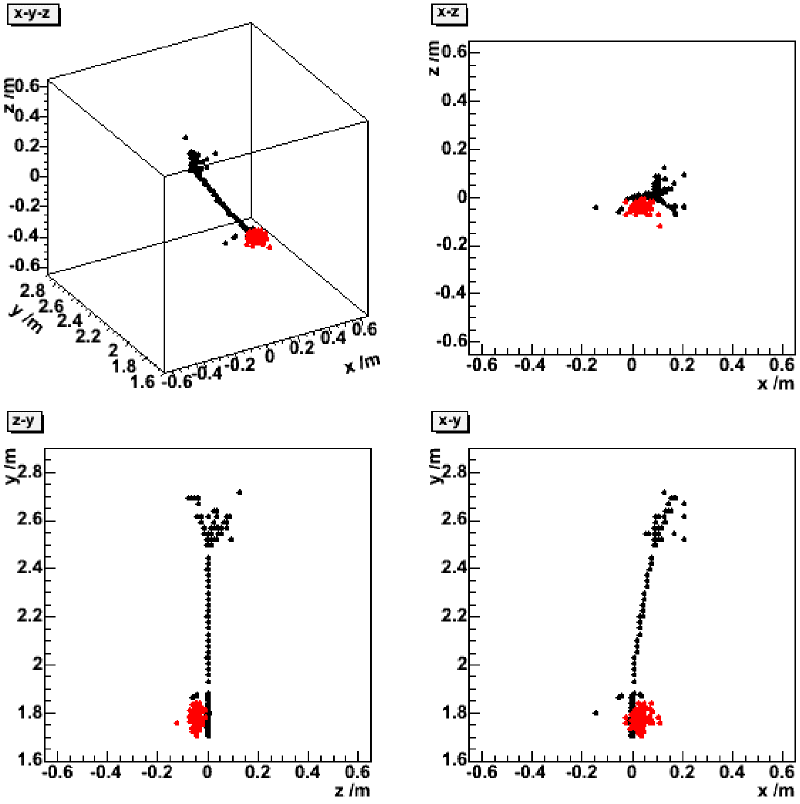
π^+/γ: Si/W Ecal + RPC/Fe DHcal (1)

True clusters



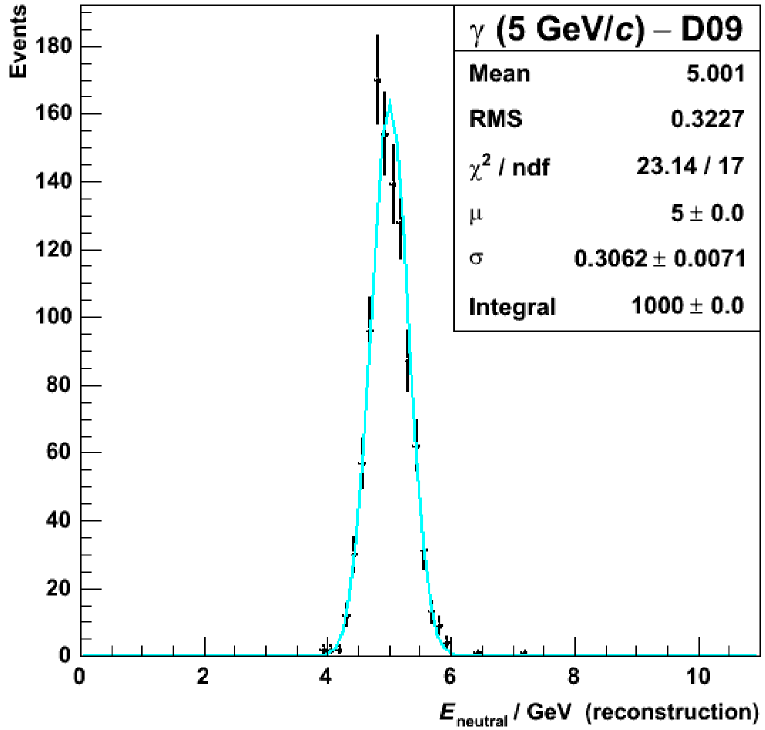
- **Black** cluster = 5 GeV/*c* π^+.
- **Red** cluster = 5 GeV/*c* γ.

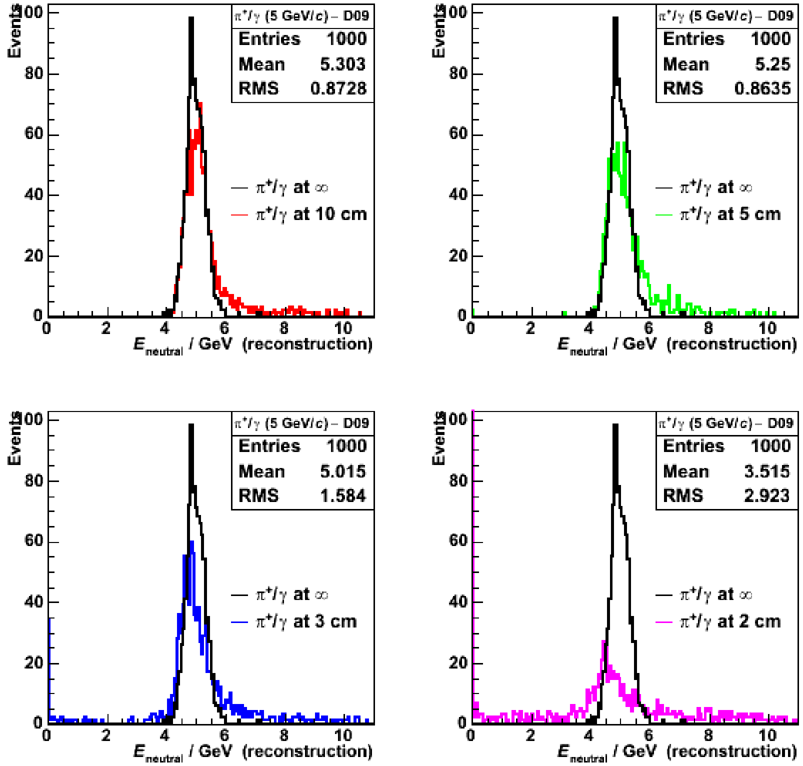
Reconstructed clusters



- **Black** cluster matched to charged track.
- **Red** cluster left over as neutral ⇒ γ energy well reconstructed.

π^+/γ: Si/W Ecal + RPC/Fe DHcal (2)



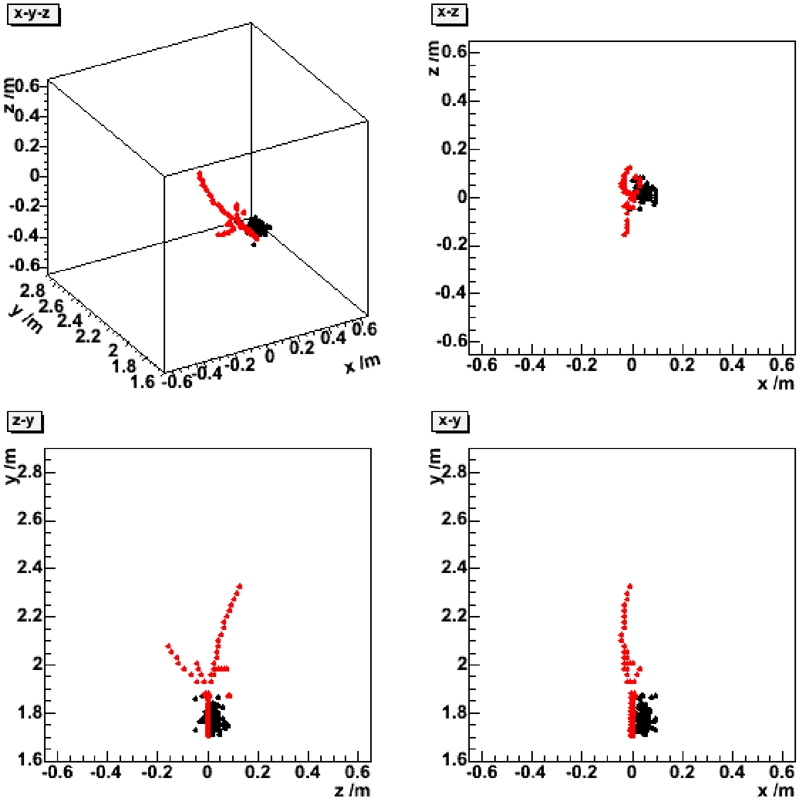


- **1k single γ at 5 GeV/*c*.**
- Fit Gaussian to energy distribution, calibrated according to:
  $E = \alpha[(E_{\text{Ecal; 1-30}} + 3E_{\text{Ecal; 31-40}})/E_{\text{Ecal mip}} + 20N_{\text{Hcal}}]$.
- Fix factors α, 20 by minimising χ^2/dof.
- $\sigma/\sqrt{\mu} \sim 14\% \sqrt{\text{GeV}}$.

- **1k γ with nearby π^+ (10, 5, 3, 2 cm from γ).**
- Peak of photon energy spectrum well reconstructed; improves with separation.
- Tail at higher *E* → inefficiency in π^+ reconstruction (next page…).
- Spike at *E* = 0 below 3 cm → clusters not distinguished.

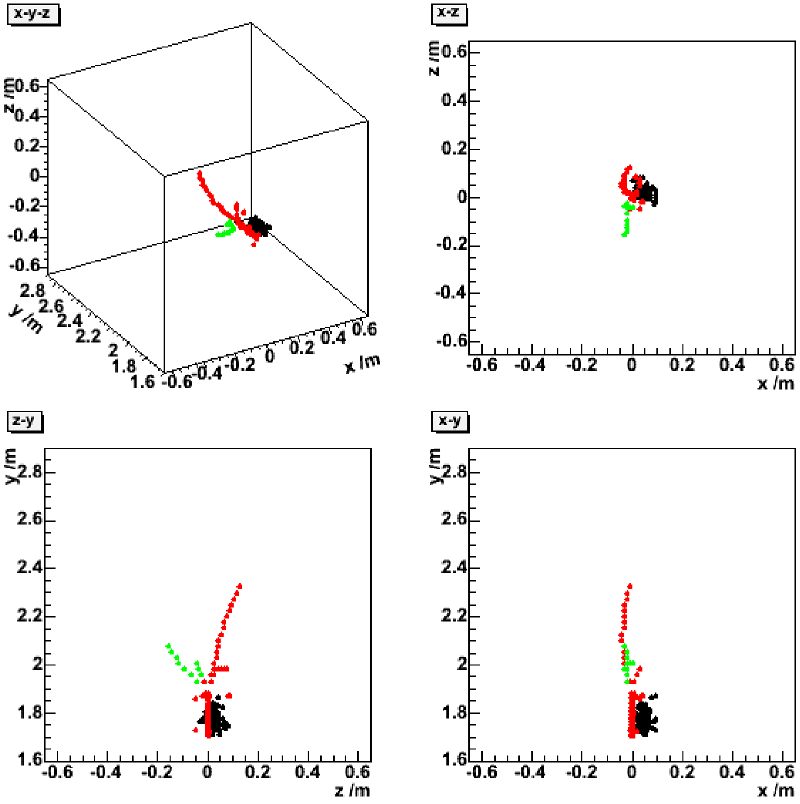
π^+/γ: Si/W Ecal + RPC/Fe DHcal (3)

True clusters



- **Red** cluster = 5 GeV/*c* π^+.
- **Black** cluster = 5 GeV/*c* γ.

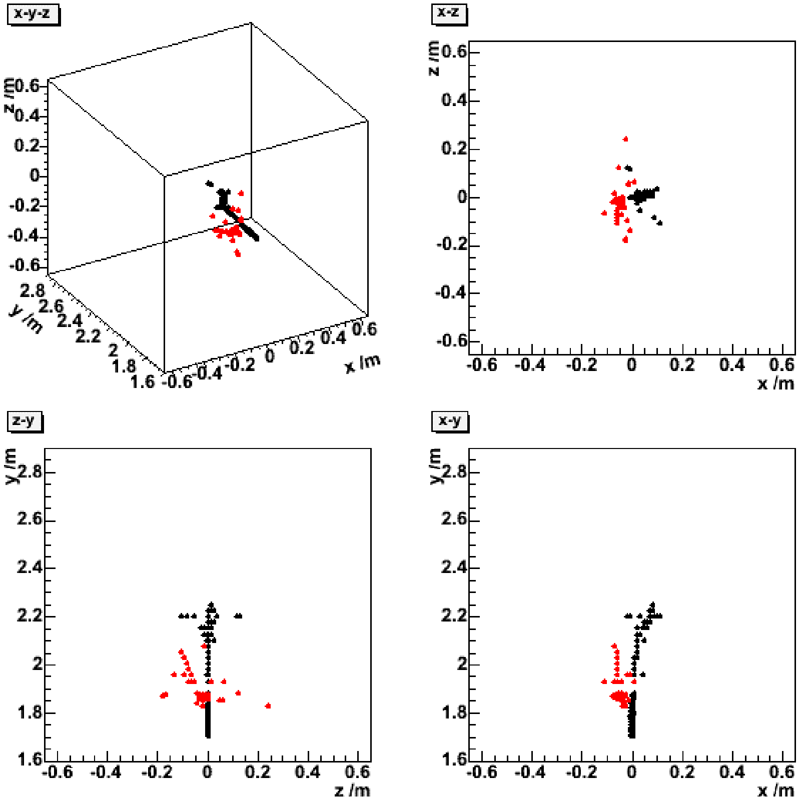
Reconstructed clusters



- **Red** cluster matched to charged track.
- **Black** ***and*** **green** clusters left over as neutral ⇒ γ energy overestimated (needs to be improved…).

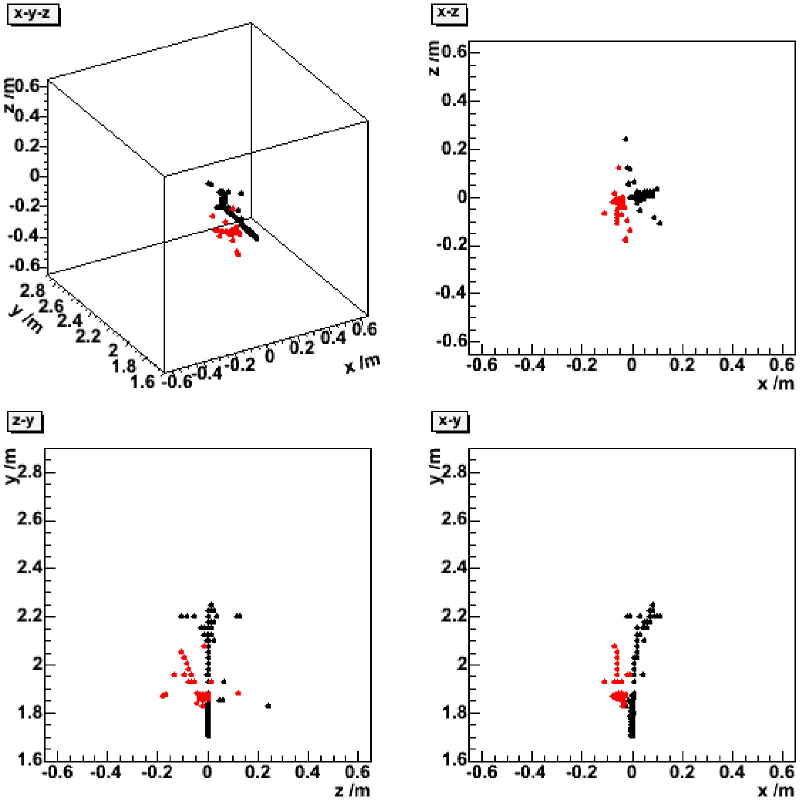
π^+/n: Si/W Ecal + RPC/Fe DHcal (1)

True clusters



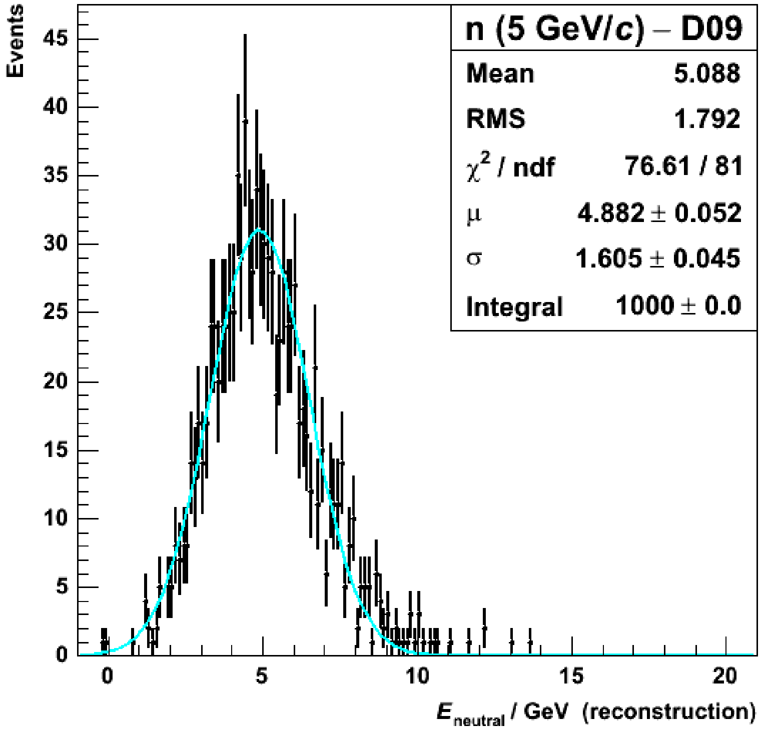
- **Black** cluster = 5 GeV/*c* π^+.
- **Red** cluster = 5 GeV/*c* **n**.

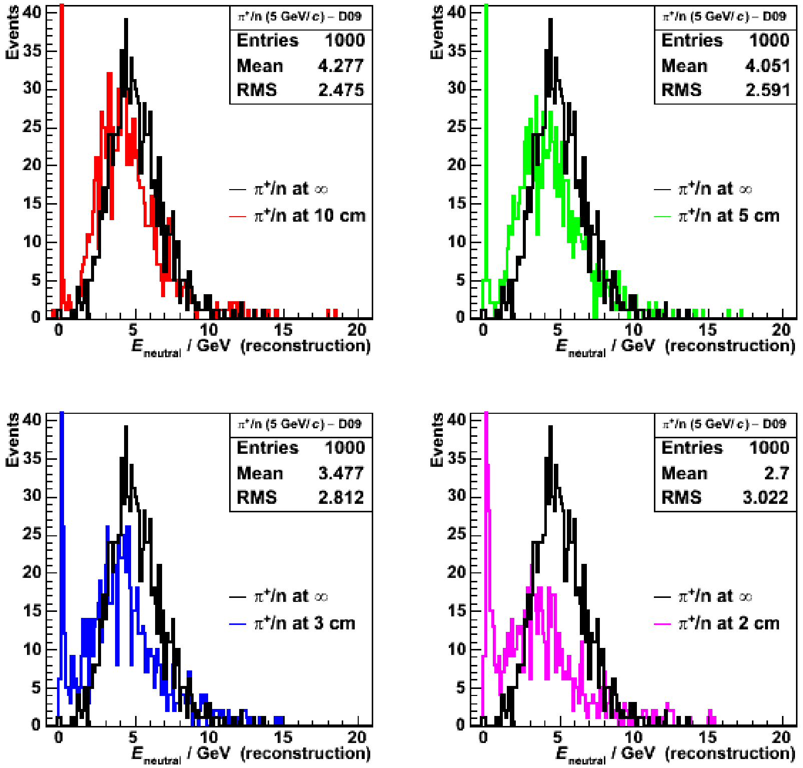
Reconstructed clusters



- **Black** cluster matched to charged track.
- **Red** cluster left over as neutral ⇒ n energy well reconstructed.

π^+/n: Si/W Ecal + RPC/Fe DHcal (2)



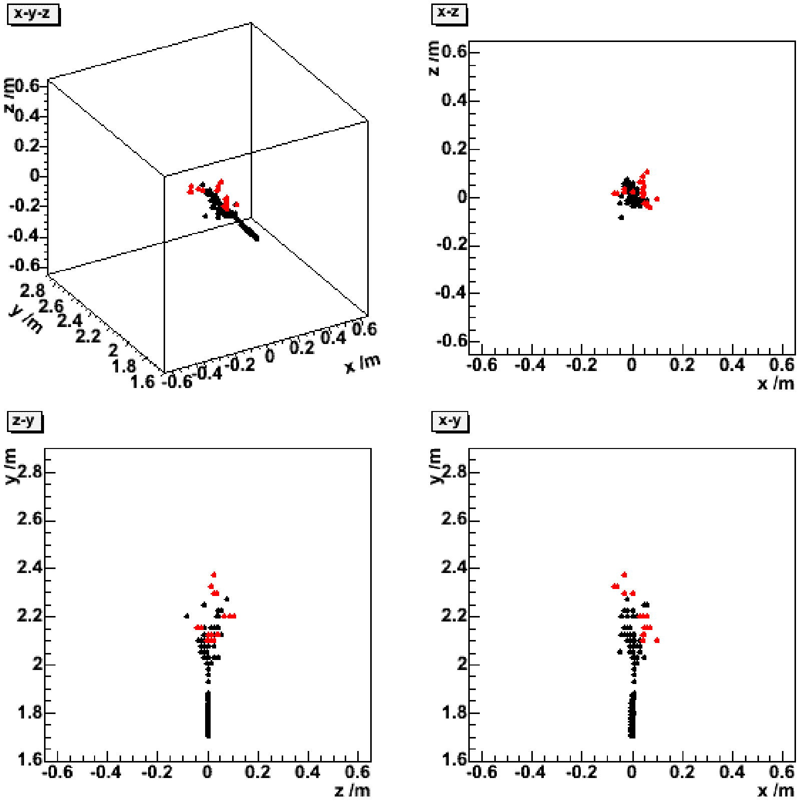


- **1k single n at 5 GeV/c.**
- Fit Gaussian to energy distribution, calibrated according to:
  $E = \alpha[(E_{\mathrm{Ecal;\,1\text{-}30}} + 3E_{\mathrm{Ecal;\,31\text{-}40}})/E_{\mathrm{Ecal\ mip}} + 20N_{\mathrm{Hcal}}]$.
- Fix factors α, 20 by minimising χ^2/dof.
- $\sigma/\surd\mu \sim 73\%\ \surd\mathrm{GeV}$.

- **1k n with nearby π^+ (10, 5, 3, 2 cm from n).**
- Peak of neutron energy spectrum well reconstructed; improves with separation.
- Spike at $E = 0$ even at 10 cm → clusters not distinguished (next page…).

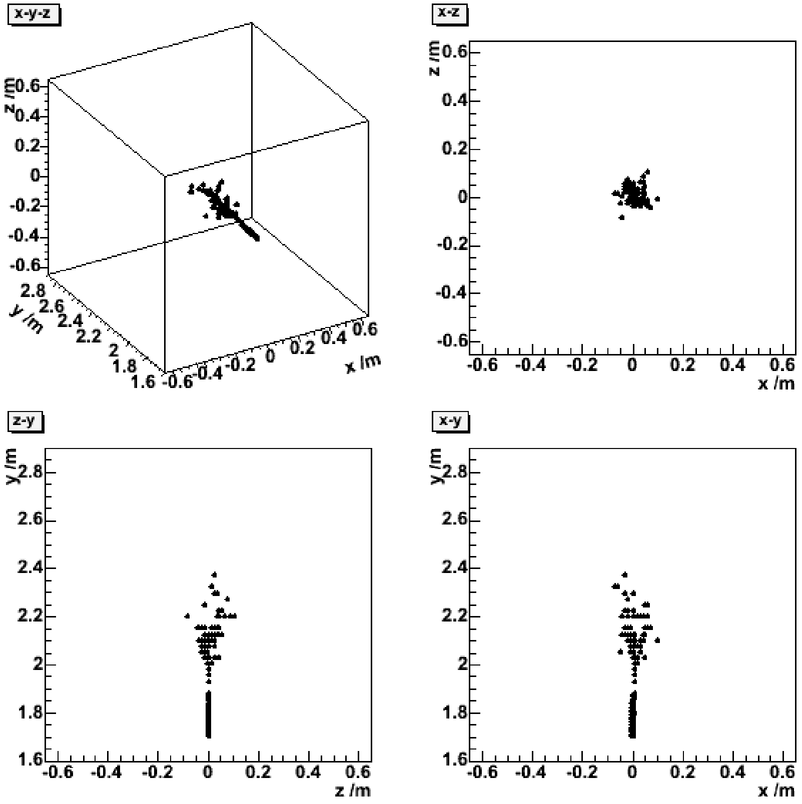
π^+/n: Si/W Ecal + RPC/Fe DHcal (3)

True clusters



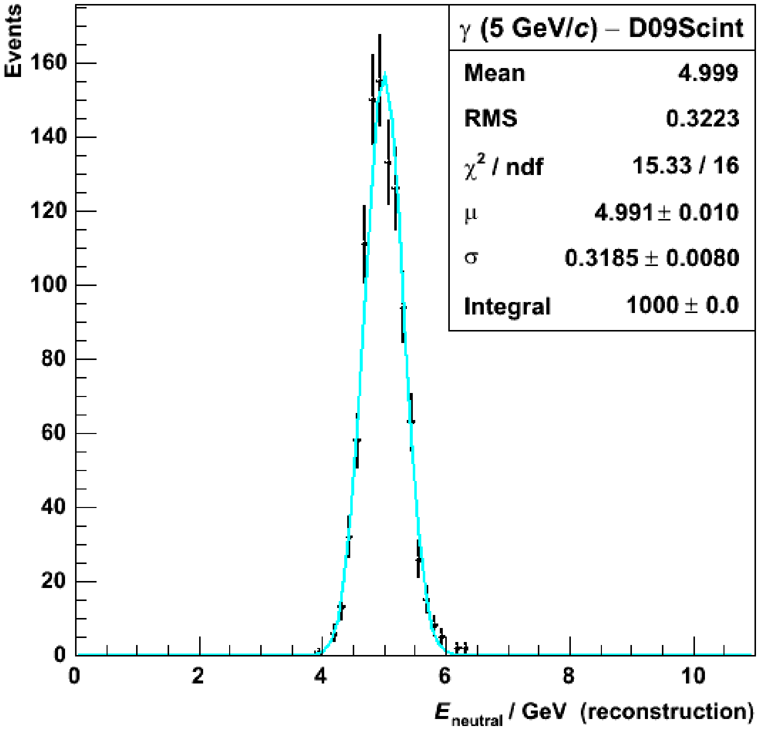
- **Black** cluster = 5 GeV/*c* π^+.
- **Red** cluster = 5 GeV/*c* **n**.

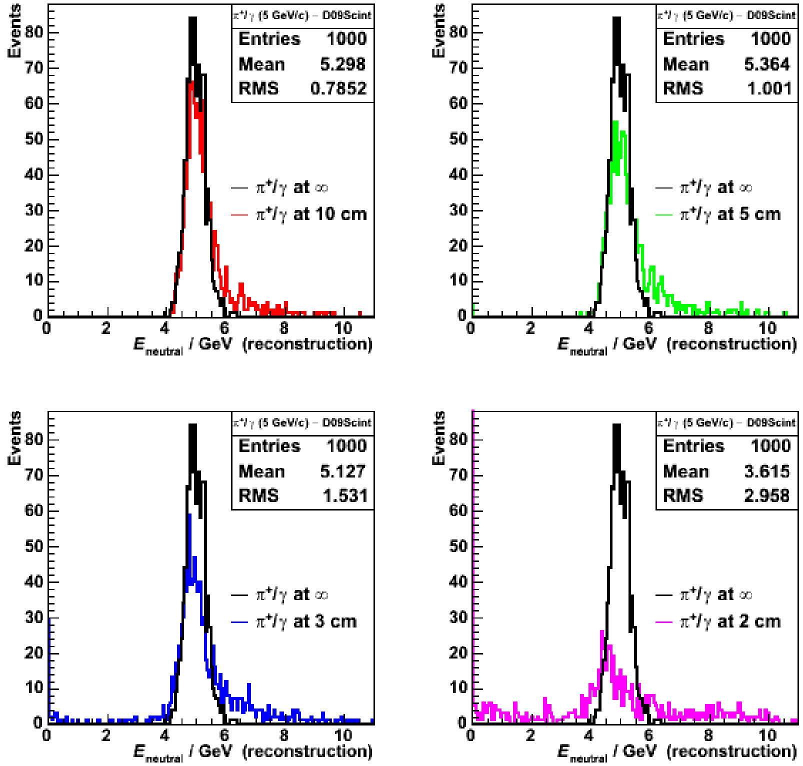
Reconstructed clusters



- **Black** cluster matched to charged track.
- Nothing left over as neutral ⇒ n not reconstructed at all (*i.e.* *E* = 0).

π^+/γ: Si/W Ecal + scintillator/Fe AHcal

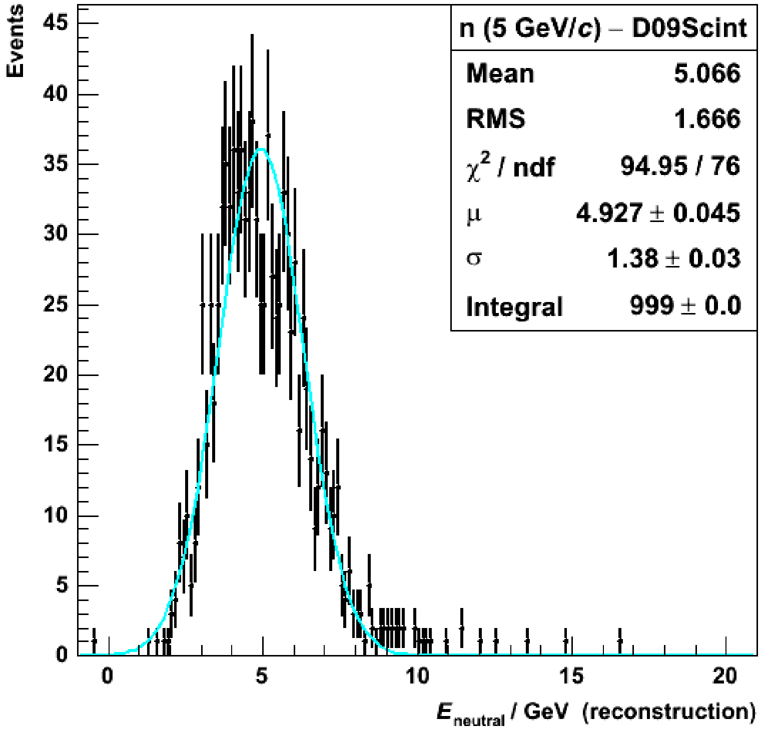


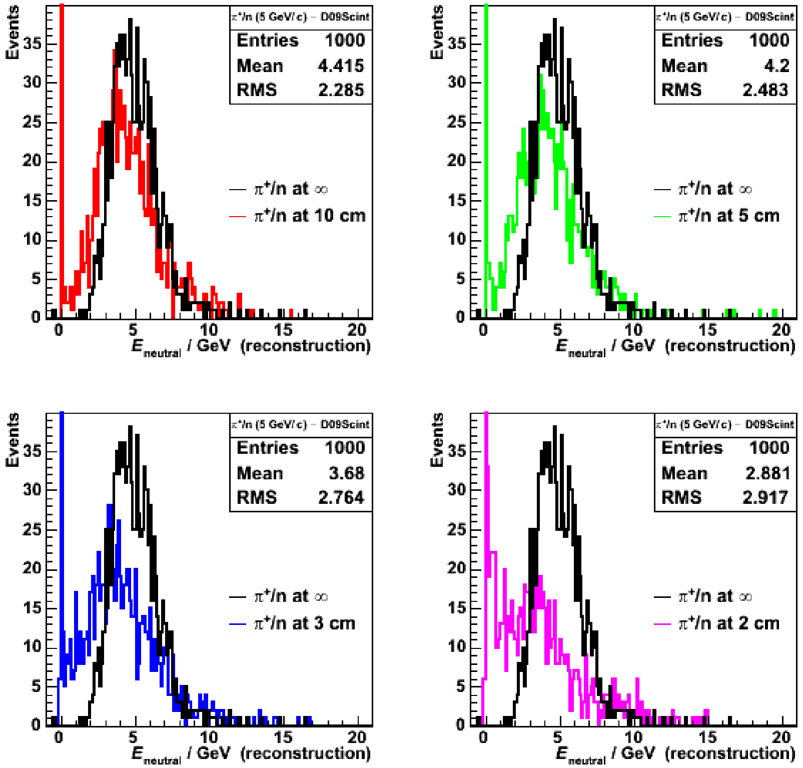


- **1k single γ at 5 GeV/*c*.**
- Fit Gaussian to energy distribution, calibrated according to:
  $E = \alpha[(E_{\mathrm{Ecal;\,1\text{-}30}} + 3E_{\mathrm{Ecal;\,31\text{-}40}})/E_{\mathrm{Ecal\ mip}} + 5E_{\mathrm{Hcal}}/E_{\mathrm{Hcal\ mip}}]$.
- Fix factors α, **5** by minimising χ^2/dof.
- $\sigma/\sqrt{\mu}$ ~ 14% √GeV (as for RPC DHcal).

- **1k γ with nearby π⁺ (10, 5, 3, 2 cm from γ).**
- General trends much as for RPC DHcal.

π^+/n: Si/W Ecal + scintillator/Fe AHcal



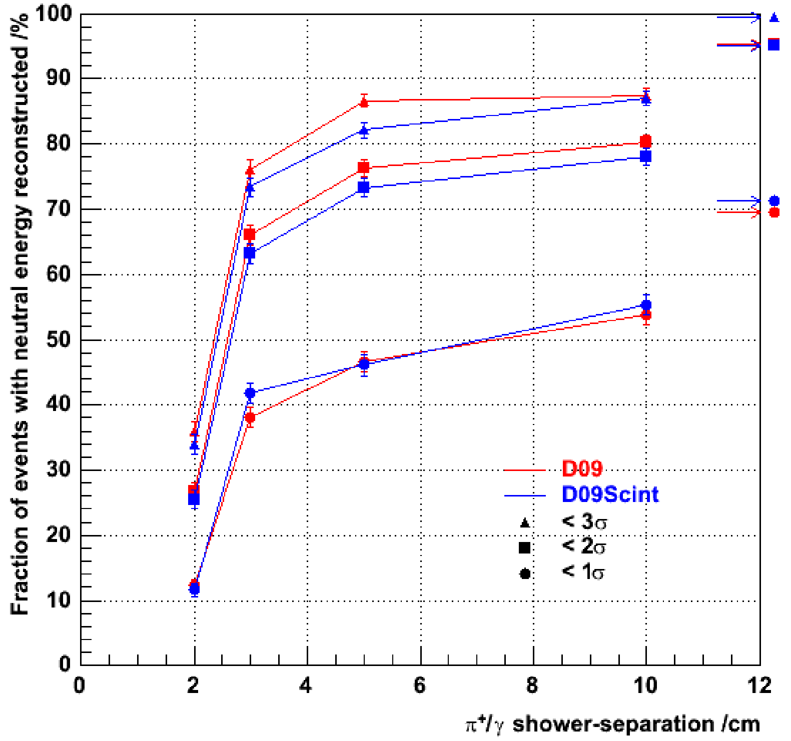


- **1k single n at 5 GeV/*c*.**
- Fit Gaussian to energy distribution, calibrated according to:
  $E = \alpha[(E_{\mathrm{Ecal;\,1\text{-}30}} + 3E_{\mathrm{Ecal;\,31\text{-}40}})/E_{\mathrm{Ecal\,mip}} + 5E_{\mathrm{Hcal}}/E_{\mathrm{Hcal\,mip}}]$.
- Fix factors α, **5** by minimising χ^2/dof.
- σ/√μ ~ 62% √GeV (cf. 73% √GeV for RPC DHcal).

- **1k n with nearby π^+ (10, 5, 3, 2 cm from n).**
- General trends much as for RPC DHcal.

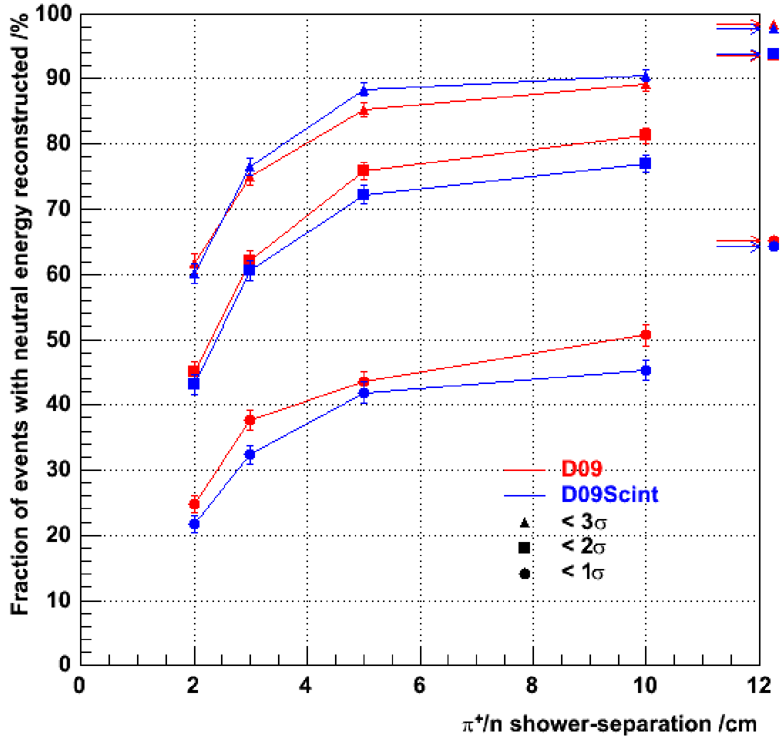
π^+/neutral cluster separability vs separation

5 GeV/c π^+/γ



- Fraction of events with photon energy reconstructed within 1,2,3σ generally higher for DHcal (“D09”) than for AHcal (“D09Scint”).

5 GeV/c π^+/n



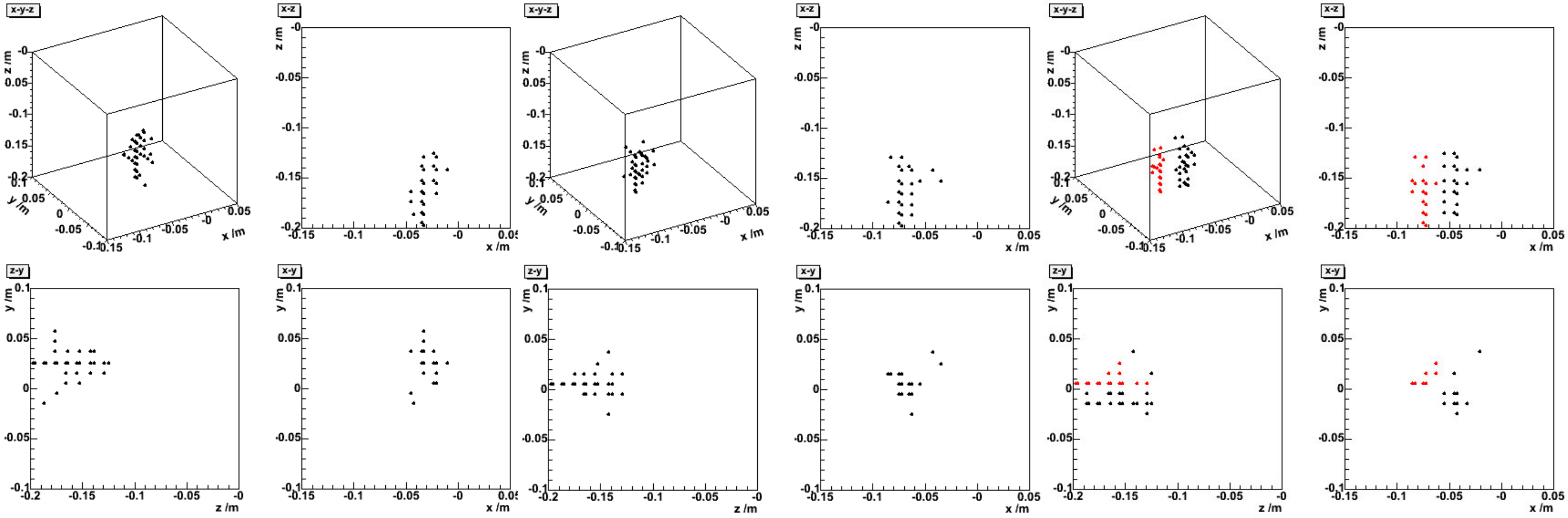
- Similar conclusion for neutrons.
- RPC DHcal favoured over scintillator AHcal?
- Needs further investigation… (reoptimisation of clustering cuts, *etc.*).

Prototype data (Run 100121) : e^- (1 GeV)

Event 803

Event 59992

Event 811



- DESY test beam (Jan. 2005): 14 layers (analogue) **Si/W Ecal**; > 50k 1 GeV e^- events.
- Default clustering cuts → events generally reconstruct as single clusters (no tracking info used).
- Events with > 1 reconstructed cluster generally look like event 811.
- On average, **98.93 ± 0.03 %** of event energy contained in highest energy reconstructed cluster (cluster energies calibrated according to: $E = \alpha(E_{\mathrm{Ecal;\,1\text{-}10}} + 2E_{\mathrm{Ecal;\,11\text{-}14}})$ GeV).

Summary & outlook

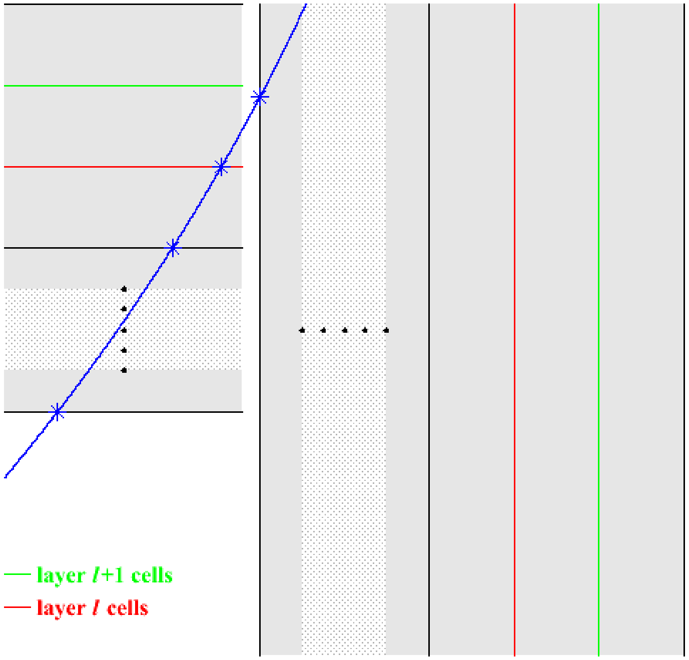
- Current version of *M*arlin-based *A*lgorithm for *G*eometry-*I*ndependent *C*lustering available from:

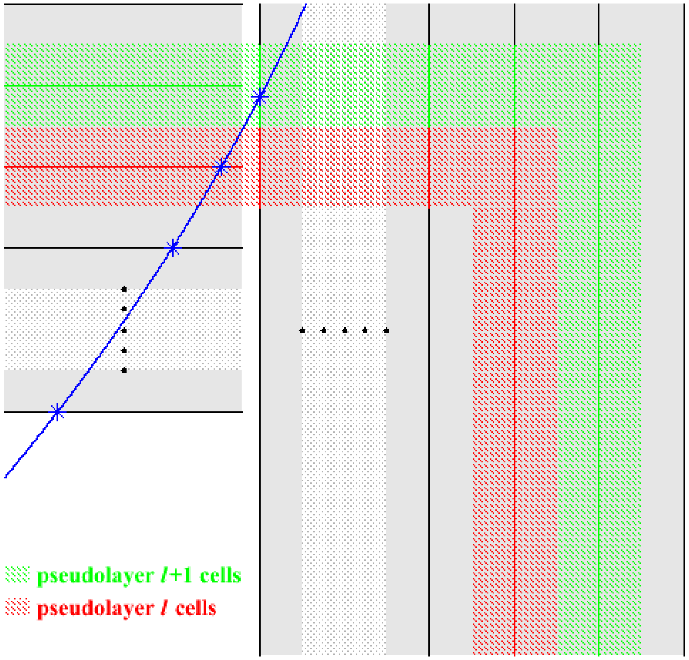
  http://www.hep.phy.cam.ac.uk/~ainsley/MAGIC/MAGIC-v01-03.tar.gz
- Also going into DESY CVS repository.
- Compliant with **LCIO** (≥ v01-05) / **MARLIN** (≥ v00-07) ⇒ input parameters (set at run-time) kept distinct from reconstruction (pre-compiled).
- Code straightforwardly **applicable to *any* detector geometry** comprising an ***n***-fold rotationally symmetric barrel closed by endcaps → just need to specify ***n***, barrel orientation, and layer positions as input (no time to demonstrate this today).
- User specifies **geometry** and **clustering cuts** at **run-time**.
- Comparisons of different calorimeter designs are straightforward (e.g. RPC DHcal vs scintillator AHcal, using Mokka models).
- Please try it out!

The end

That's all folks…

Generalising the calorimeter (1)

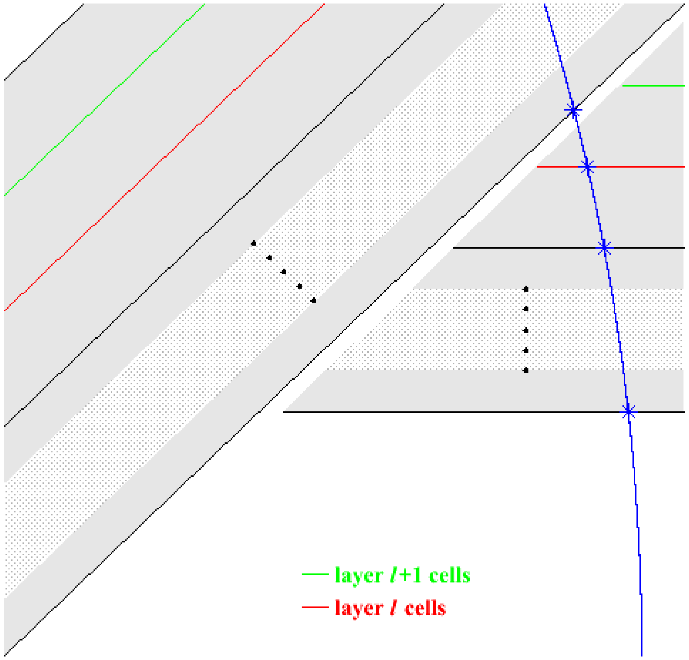


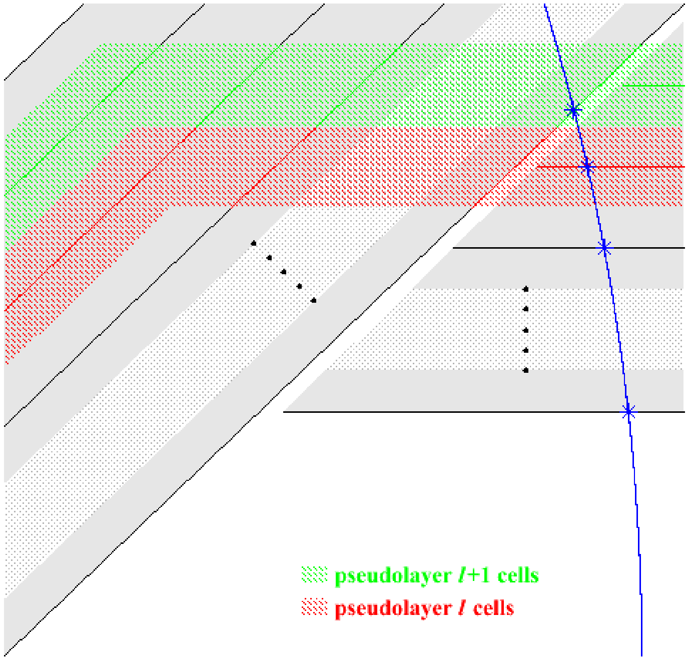


- Layer index changes discontinuously at barrel/endcap boundary.
- On crossing, jumps from l to 1 (first Ecal layer).

- Define a "***pseudolayer***" index based on projected intersections of physical layers.
- Index varies smoothly across boundary.
- Pseudolayer index = layer index, *except* in overlap region.

Generalising the calorimeter (2)

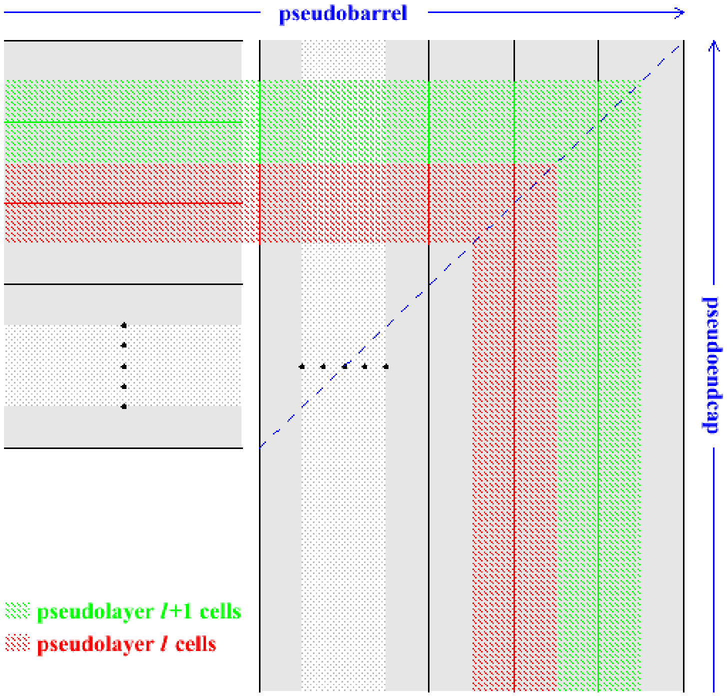


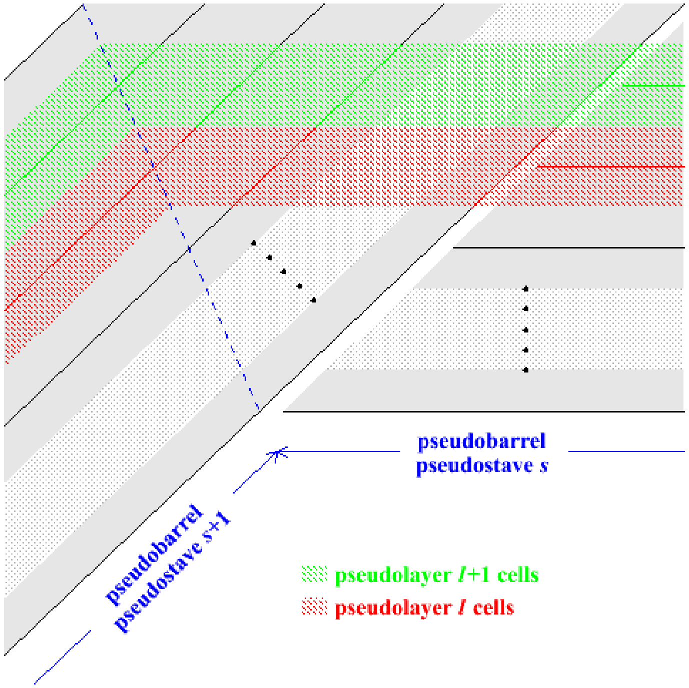


- Layer index changes discontinuously at boundary between overlapping barrel staves.
- On crossing, jumps from *l* to 1 (first Ecal layer).

- Again, define ***"pseudolayer"*** index from projected intersections of physical layers.
- Again, index varies smoothly across boundary.
- Again, pseudolayer index = layer index, *except* in overlap region.

Generalising the calorimeter (3)





- Define a "***pseudostave***" as a plane of parallel pseudolayers.
- "***Pseudobarrel***" pseudostaves meet boundaries with left- and right-hand "***pseudoendcap***" pseudostaves along 45° lines (if layer-spacings equal in barrel and endcaps).

- "***Pseudobarrel***" pseudostaves meet boundaries with other "***pseudobarrel***" pseudostaves along 360°/2***n*** lines (for an ***n***-fold rotationally symmetric barrel).
- Calorimeter divides naturally into ***n*+2** pseudostaves.

Generalising the calorimeter (4)

- Code recasts any layered calorimeter composed of a rotationally symmetric barrel closed by two endcaps into this standard, generalised form comprising layered shells of rotationally-symmetric ***n***-polygonal prisms, coaxial with *z*-axis.
- Layers and staves from which calorimeter is built translated into pseudolayers and pseudostaves with which algorithm works.
- Only required inputs as far as algorithm is concerned are:
  - **`barrelSymmetry`** = rotational symmetry of barrel (***n***);
  - **`phi_1`** = orientation of pseudobarrel pseudostave 1 w.r.t. *x*-axis;
  - **`distanceToBarrelLayers[ecalLayers+hcalLayers+2]`**

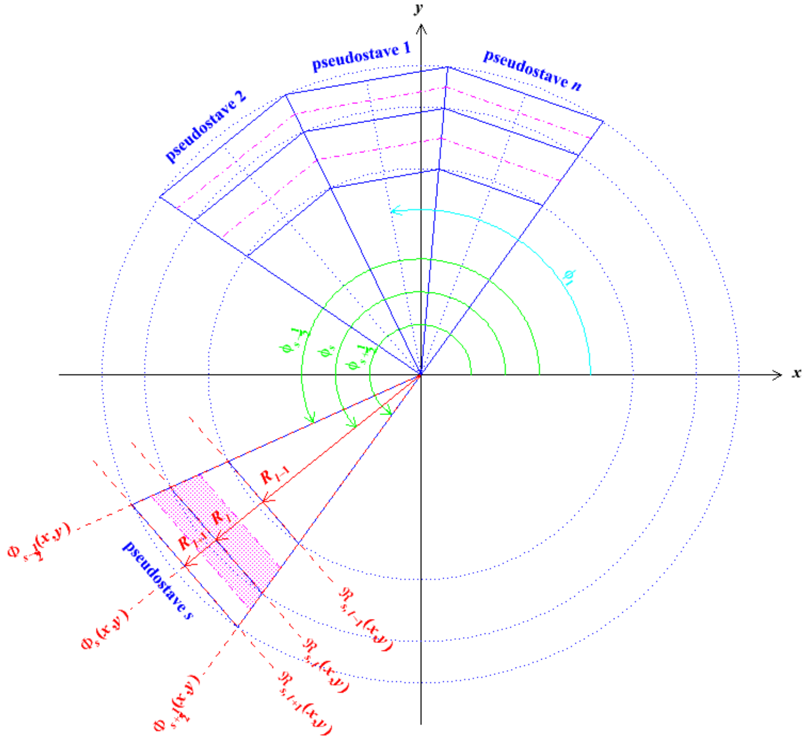
    = layer positions in barrel layers ("+2" to constrain inside edge of first pseudolayer and outside edge of last pseudolayer); and
  - **`distanceToEndcapLayers[ecalLayers+hcalLayers+2]`**

    = layer positions in endcap layers;

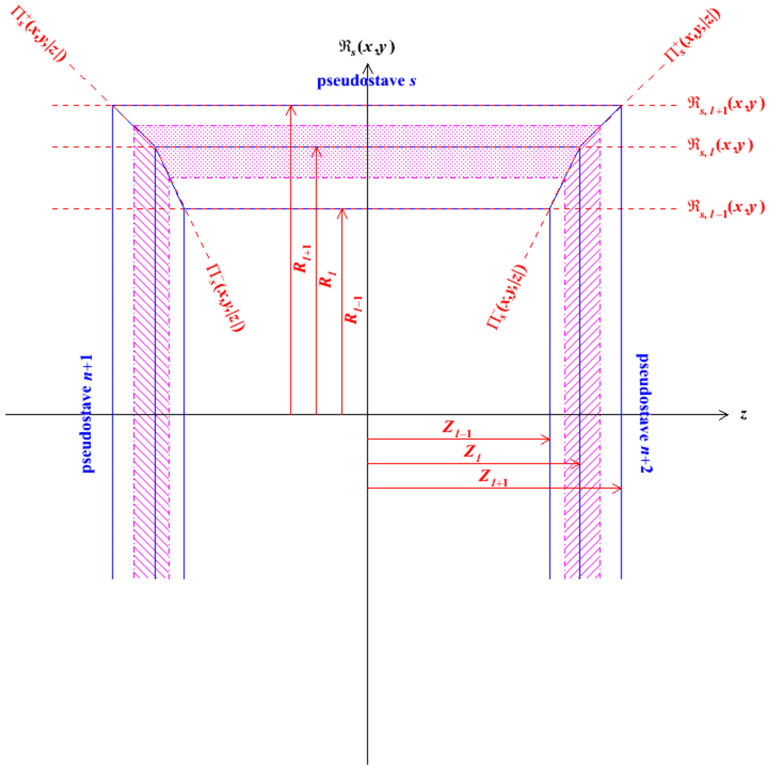
→ as geometry-independent as it's likely to get!

How the generalised detector shapes up

Transverse section



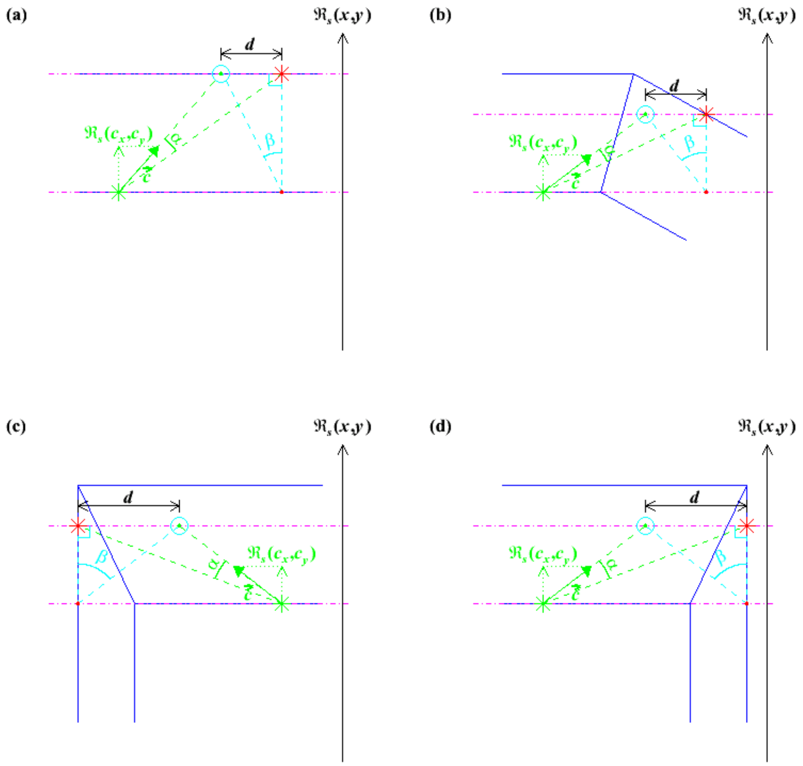
Longitudinal section



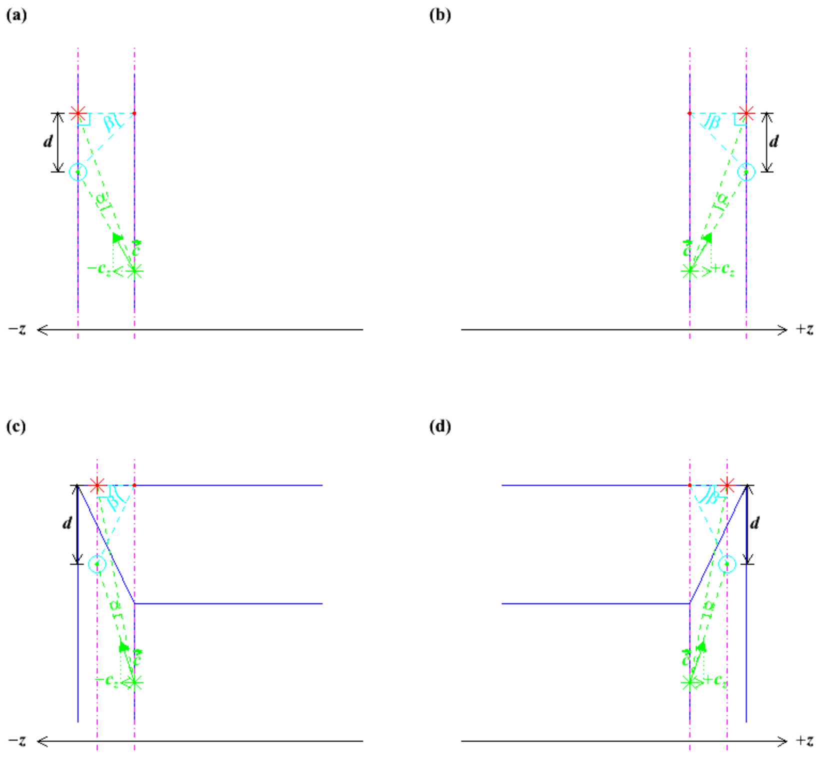
- Solid blue lines aligned along real, physical, sensitive layers.
- Dot-dashed magenta lines bound shell containing hits with same ***pseudolayer*** index, l.
- ***Pseudostaves*** automatically encoded by specifying n, ϕ_1 and R_l and Z_l ($\forall\ l$).

Cluster-tracking between pseudolayers

From the pseudobarrel

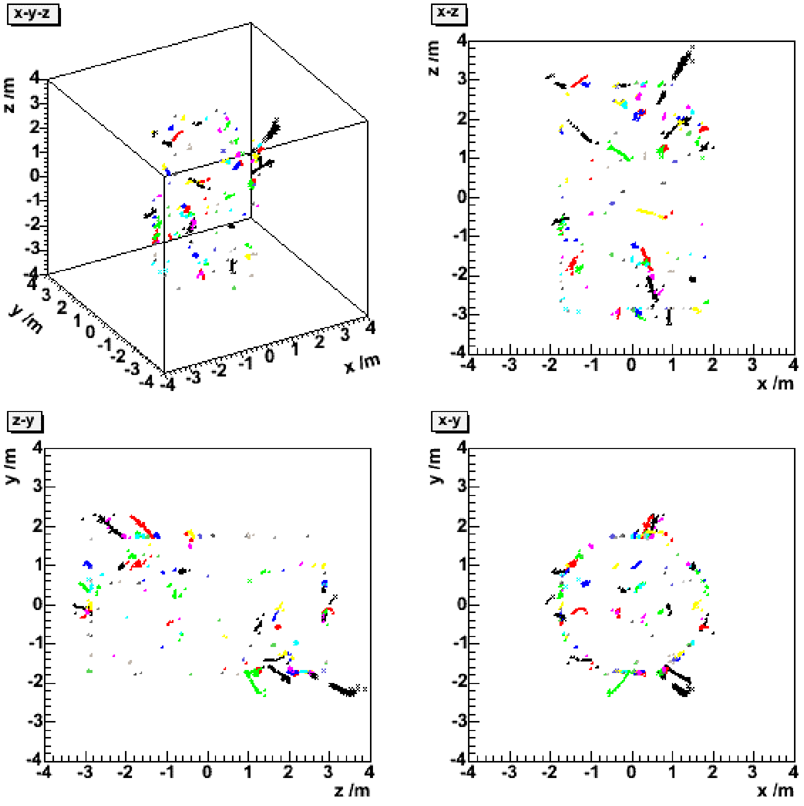


From the pseudoendcap

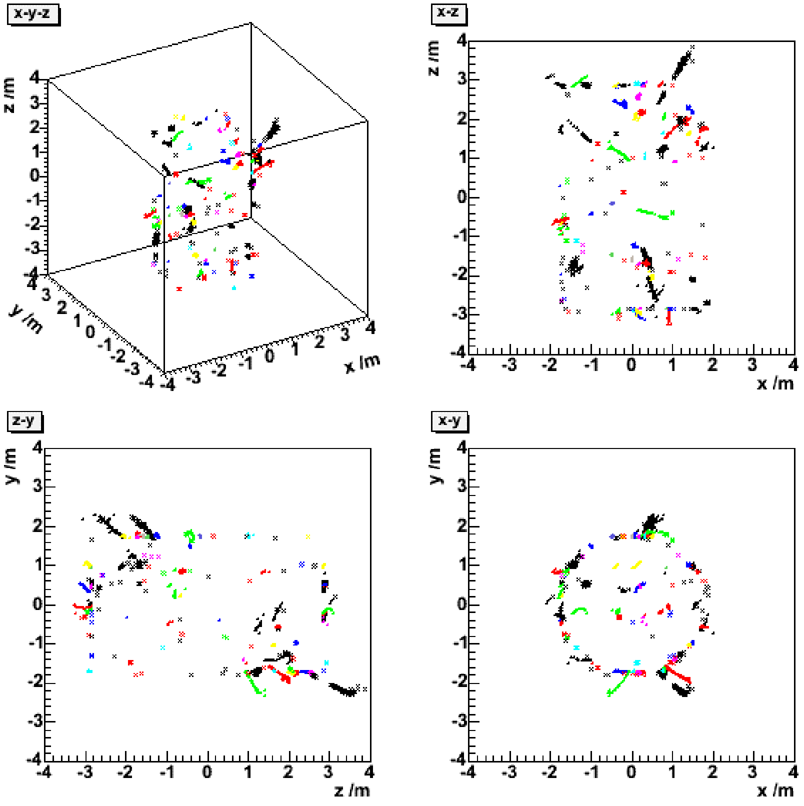


Example event: $Z \rightarrow u,d,s$ jets at 91 GeV

Reconstructed clusters



True clusters



- Reconstruction works successfully not only for ***intra***-stave, but also for ***inter***-stave clusters (*e.g.* ***black*** truth cluster spanning barrel staves 5+6 and the RH endcap correctly reconstructed).

Code organisation within LCIO/MARLIN

- Code structured as a series of 6+1 **MARLIN** "***processors***", together with a **steering file**: `cluster.steer` (read at ***run-time***).
- Reads hits collections from **LCIO** file, adds **LCIO** clusters collections (essentially pointers back to component hits) and writes everything to new **LCIO** output file.
- Processors to do the reconstruction:
  - `CalorimeterConfigurer`
    → allows user to define geometrical layout of calorimeter;
  - `CalorimeterHitSetter`
    → applies hit-energy threshold and adds pseudolayer and pseudostave indices to hits collection (encoded in CellID1 akin to encoding of layer and stave indices in CellID0) as well as hit weights (= local hit density);
  - `CalorimeterStage1Clusterer`
    → performs coarse cluster reconstruction;
  - `CalorimeterStage2Clusterer`
    → joins broken mip-track-like cluster fragments;
  - `CalorimeterStage3Clusterer`
    → recovers backward-spiralling track-like cluster fragments;
  - `CalorimeterStage4Clusterer`
    → recovers low multiplicity cluster fragments.
- Additional processor to access MC truth (if simulation):
  - `CalorimeterTrueClusterer`
    → constructs true clusters, where a true cluster is considered to comprise all hits attributable to either:
    (i) the same generator primary or any of its non-backscattered progeny, or
    (ii) the same backscattered daughter or any of its non-backscattered progeny.

Code organisation within LCIO/MARLIN

- Layer positions set (for convenience) in `CalorimeterConfigurer.cc`:

```
// Create collections to store the barrel and endcap layer positions
LCCollectionVec* distanceToBarrelLayersVec = new LCCollectionVec(LCIO::LCFLOATVEC);
LCCollectionVec* distanceToEndcapLayersVec = new LCCollectionVec(LCIO::LCFLOATVEC);

// Fill the collections with their positions (in mm)
for(int l=0; l<=ecalLayers+hcalLayers+1; l++) {
  LCFloatVec* distanceToBarrelLayers = new LCFloatVec;
  LCFloatVec* distanceToEndcapLayers = new LCFloatVec;
  if(detectorType=="full") {  // full detector
    if(l<=30) {  // first 30 Ecal layers at a pitch of 3.9 mm (+ layer 0)              ← edit
      distanceToBarrelLayers->push_back(1698.85+(3.9*l));                                ← edit
      distanceToEndcapLayers->push_back(2831.10+(3.9*l));                                ← edit
    }                                                                                    ← edit
    else if(l>30 && l<=ecalLayers) {  // last 10 Ecal layers at a pitch of 6.7 mm        ← edit
      distanceToBarrelLayers->push_back(1815.85+(6.7*(l-30)));                           ← edit
      distanceToEndcapLayers->push_back(2948.10+(6.7*(l-30)));                           ← edit
    }                                                                                    ← edit
    else {  // 40 Hcal layers at a pitch of 24.5 mm (+ layer 81)                         ← edit
      distanceToBarrelLayers->push_back(1931.25+(24.5*(l-41)));                          ← edit
      distanceToEndcapLayers->push_back(3039.25+(24.5*(l-41)));                          ← edit
    }                                                                                    ← edit
  }
  else if(detectorType=="prototype") { …some more code… }  // prototype detector
  distanceToBarrelLayersVec->push_back(distanceToBarrelLayers);
  distanceToEndcapLayersVec->push_back(distanceToEndcapLayers);
}

// Save the collections
evt->addCollection(distanceToBarrelLayersVec, "distance_barrellayers");
evt->addCollection(distanceToEndcapLayersVec, "distance_endcaplayers");
```

Getting started with MAGIC (1)

- Install **LCIO** (≥ v01-05) and **MARLIN** (≥ v00-07).
- Download *MAGIC* tar-ball from

  `http://www.hep.phy.cam.ac.uk/~ainsley/MAGIC/MAGIC-v01-03.tar.gz`
- **Two directories** and a **README** file (read this first!).
- The `clustering` **directory** contains the cluster-reconstruction (and cluster-truth) code (i.e. all processors and steering file mentioned earlier).
- Takes **.slcio input** files containing `CalorimeterHits` (data) or `SimCalorimeterHits` (MC):
  - must be generated with hit-positions stored, *i.e.* `RCHBIT_LONG=1` (data) or `CHBIT_LONG=1` (MC);
  - collection names must contain the string "`ecal`" or "`hcal`" (in upper or lower case, or in some combination of these) to identify the type of hit (for energy-threshold application).
- Produces **.slcio output** file with cluster-related collections added:
  - `CalorimeterHits` ⇒ hits above energy threshold;
  - `CalorimeterHitRelationsToSimCalorimeterHits` (**MC only**) ⇒ pointers to original simulated hits;
  - `CalorimeterStage1Clusters` ⇒ clusters after stage 1 of algorithm;
  - `CalorimeterStage2Clusters` ⇒ clusters after stage 2 of algorithm;
  - `CalorimeterStage3Clusters` ⇒ clusters after stage 3 of algorithm;
  - `CalorimeterStage4Clusters` ⇒ clusters after stage 4 of algorithm;
  - `CalorimeterTrueClusters` (**MC only**) ⇒ true clusters;
  - `CalorimeterTrueClusterRelationsToMCParticles` (**MC only**) ⇒ pointers to original MC particles.
- The `examples` **directory** contains example analysis code which performs simple manipulations with the clusters (e.g. processors which add calibrated energies to clusters, produce the plots shown earlier, calculate the reconstruction quality… and an accompanying steering file).

Getting started with MAGIC (2)

- For new LCIO `CalorimeterHits` collection can:
  - `getCellID0()`;
  - `getCellID1()` ⇒ pseudolayer/stave id encoded like layer/stave id in CellID0;
  - `getEnergy()`;
  - `getPosition()`;
  - `getType()` ⇒ “0”=ecal hit; “1”=hcal hit.
- For all new LCIO `Calorimeter*Clusters` collections, can:
  - `getCalorimeterHits()`;
  - `getHitContributions()`; and
  - `getClusters()`

  (no energy/position/shape attributes set—user can set these in own private processors as desired).
- If simulation, can also use `LCRelationNavigator` to:
  - `simHitRel->getRelatedToObjects(hit)`, and
  - `mCParticleRel->getRelatedToObjects(trueCluster)`.