Updates on dRICH reconstruction software

Chandradoy Chatterjee
on behalf of the dRICH simulation team

Ouline

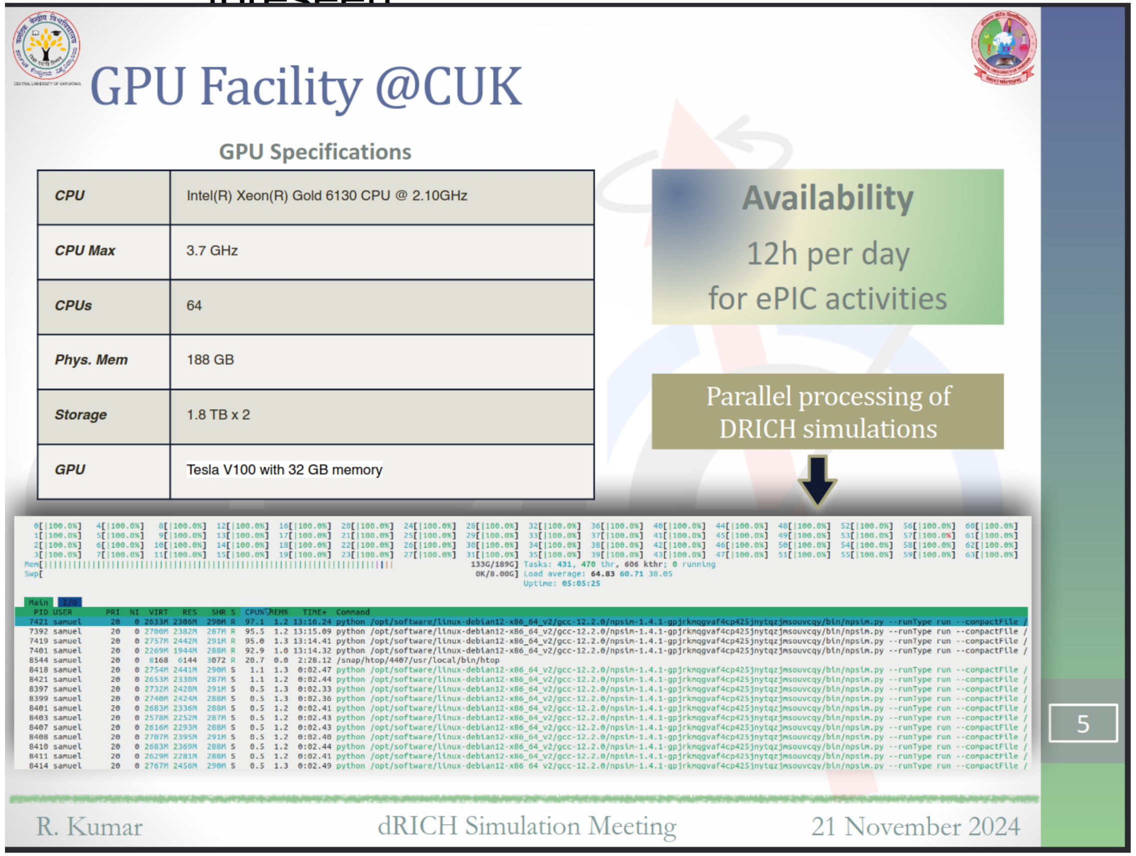
- 1) Team involved for the simulation studies.
- 2) Principle of dRICH reconstruction software.
- 3) IRT-EICRecon Interface.
- 4) Performance studies with dRICH software.
- 5) Synergy with pfRICH.
- 6) Future work-plan with pfRICH colleagues.
- 7) Conclusions.

1. Current dRICH simulation community

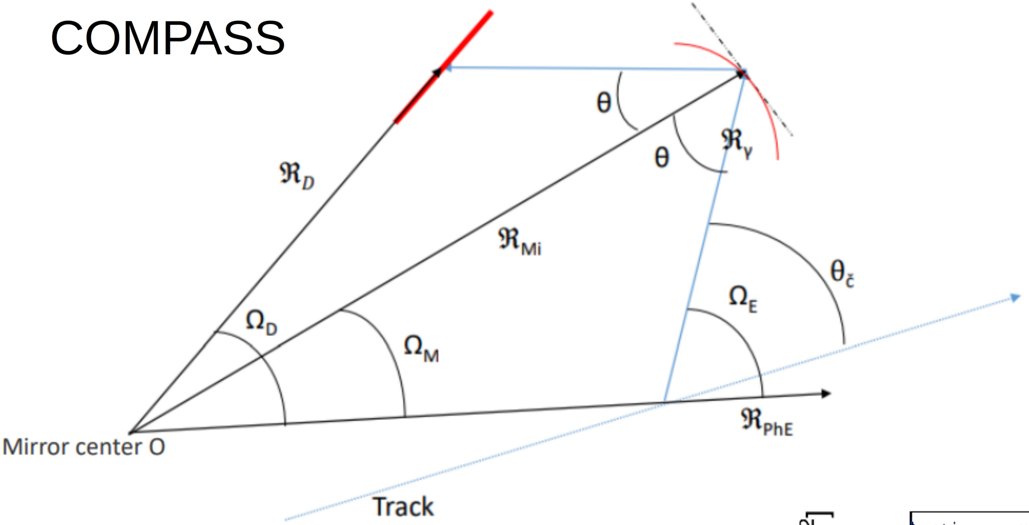
- Current team:
- Central University of Karnataka, India
  - Deepak Samuel
  - A. Rajan
  - N. George
- Central Universtiy of Haryana, India
  - Ramandeep Kumar
  - Meenu Thakur
  - R. Jangid
  - Taniya
  - T.Tanvi
  - G.Laishram
- Ramaiah University of applied sciences, India
  - Tapasi Ghosh
  - Rohit Sigh
- INFN Trieste
  - Jinky Agarwala
  - Chandradoy Chatterjee
- INFN Cosenza & University of Calabria
  - Luisa Occhiuto

Constant guidance and helps from Marco and Silvia.

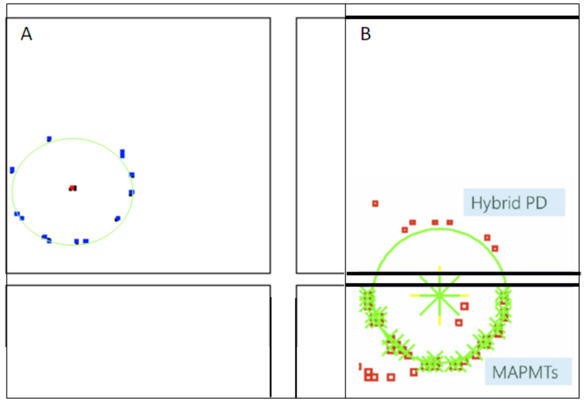
In future new collaborations are foreseen

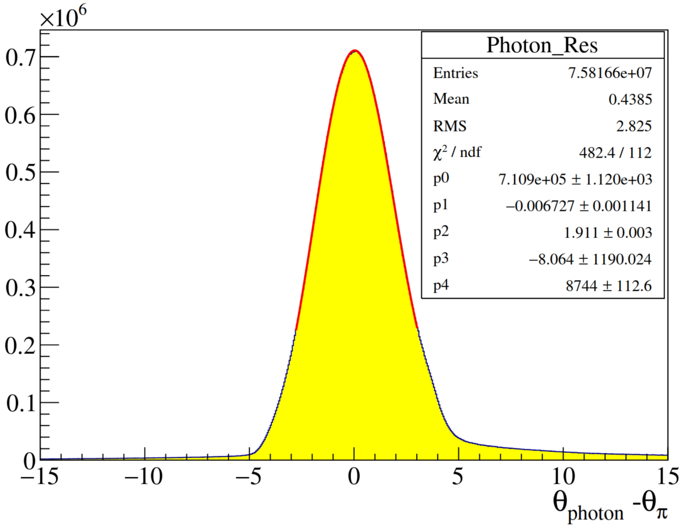


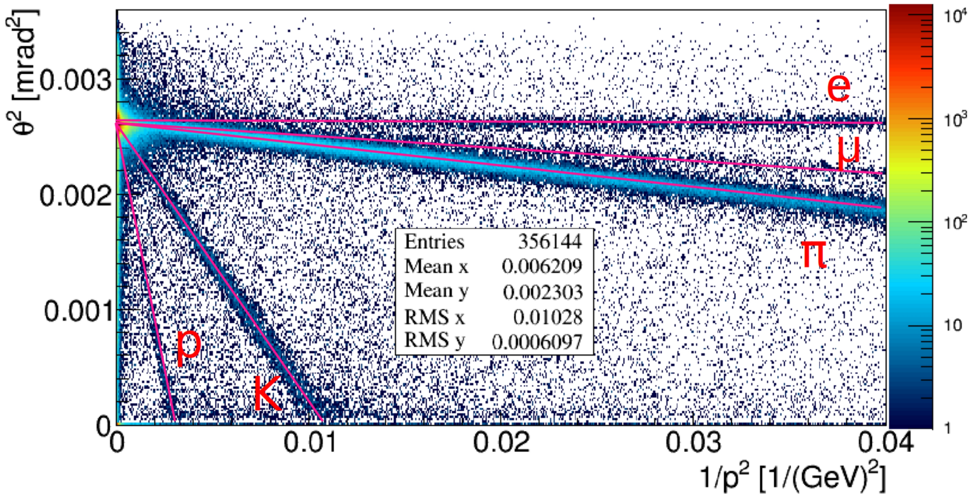
2) Inverse ray trace example in COMPASS RICH

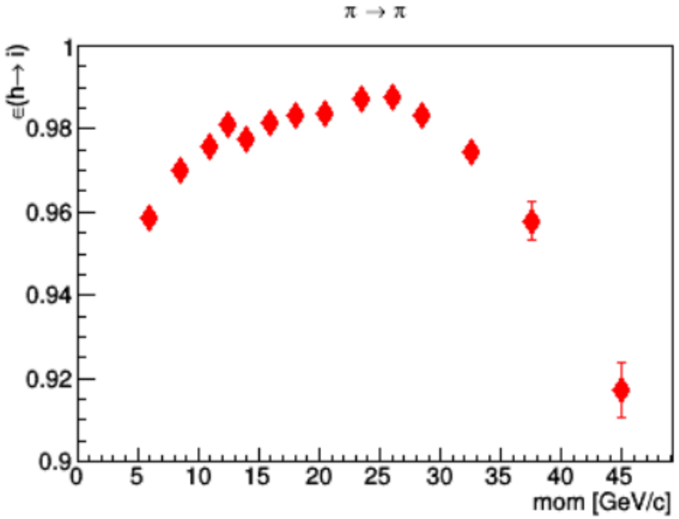


- We know the angles Ω_D and angles Ω_M.
- We know the lengths $\mathcal{R}_\mathcal{D}$, $\mathcal{R}_\mathcal{M}$ and $\mathcal{R}_{phE}$.
- We just have to apply a sine law for similar triangles to have an estimate of the reflection angle θ.
- Given that $\Omega_D - \Omega_M - \theta$ is a very small number, we can determine the angle Ω_E in an iterative method.
- Once we know Ω_E, we can determine the photon vector (ν).
- The projection of ν on track vector **P** is the **reconstructed Cherenkov angle**.

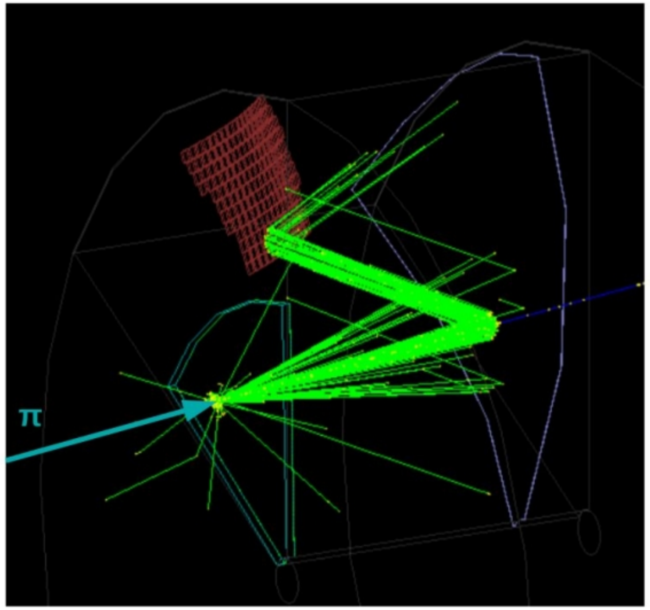


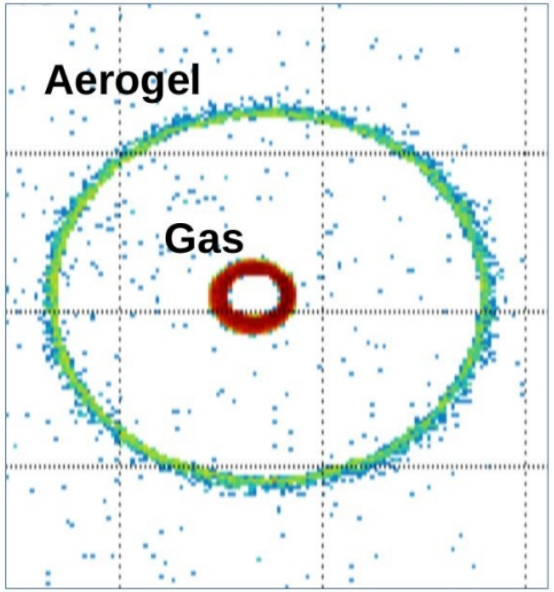


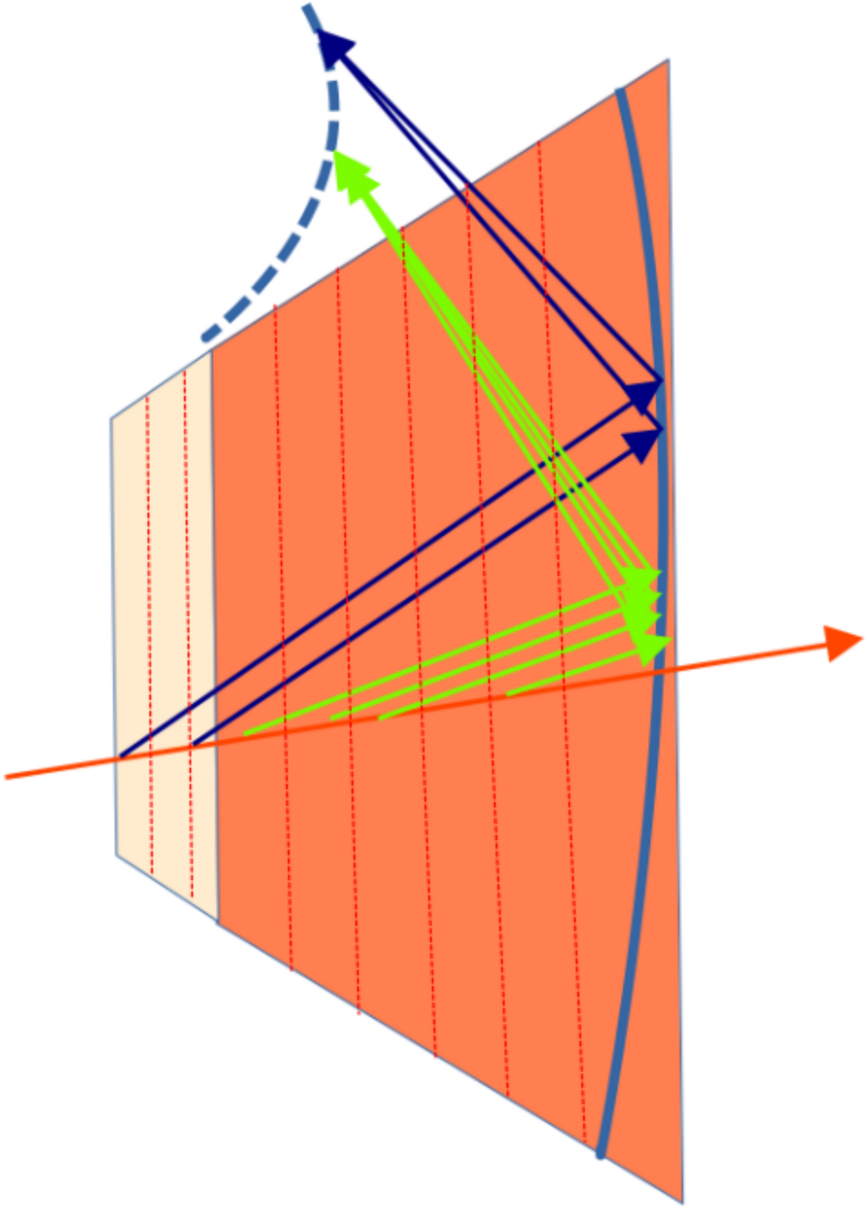




2) Scheme(optical) in dRICH



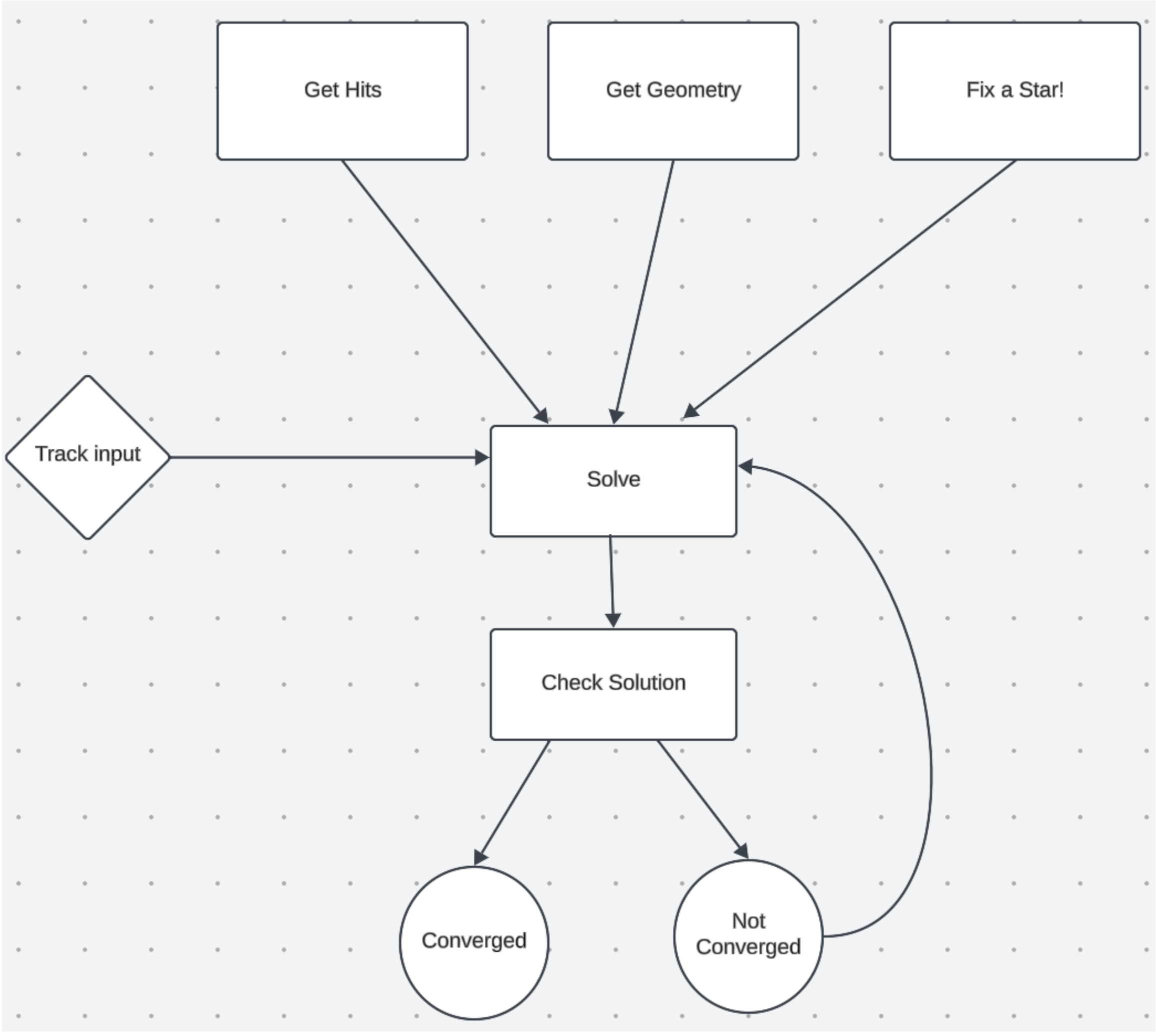




The track projection plane takes automatically into account the correction needed for the magnetic field.

Not relevant for aerogel

3) dRICH IRT logic flow



- Inside EICrecon Get the hits from the simulation: scales them with QE and MC hits are digitized.
- A geometry class takes account the right geometry:

  DD4hep: simulation geometry
  Readout: DD4hep readout pixel geometry
  ACTS: track-projection planes
  IRT: optical surfaces for Indirect Ray Tracing

- The nominal beam line (Z axis) is fixed as star for all coordinate definitions.
- EICrecon feeds the track reconstruction parameter for the dRICH radiator volume.
  These are projection planes at fixed intervals.
- All these information are then processed. And checked if the solution is converged.
- Two parameters (theta,phi): generalized Gauss-Newton Method.
- Based on the reconstructed Cherenkov angle's closeness to one hypothesis over the other a PID weight can be ascribed. Coarse PID.

3) IRT in EICRecon

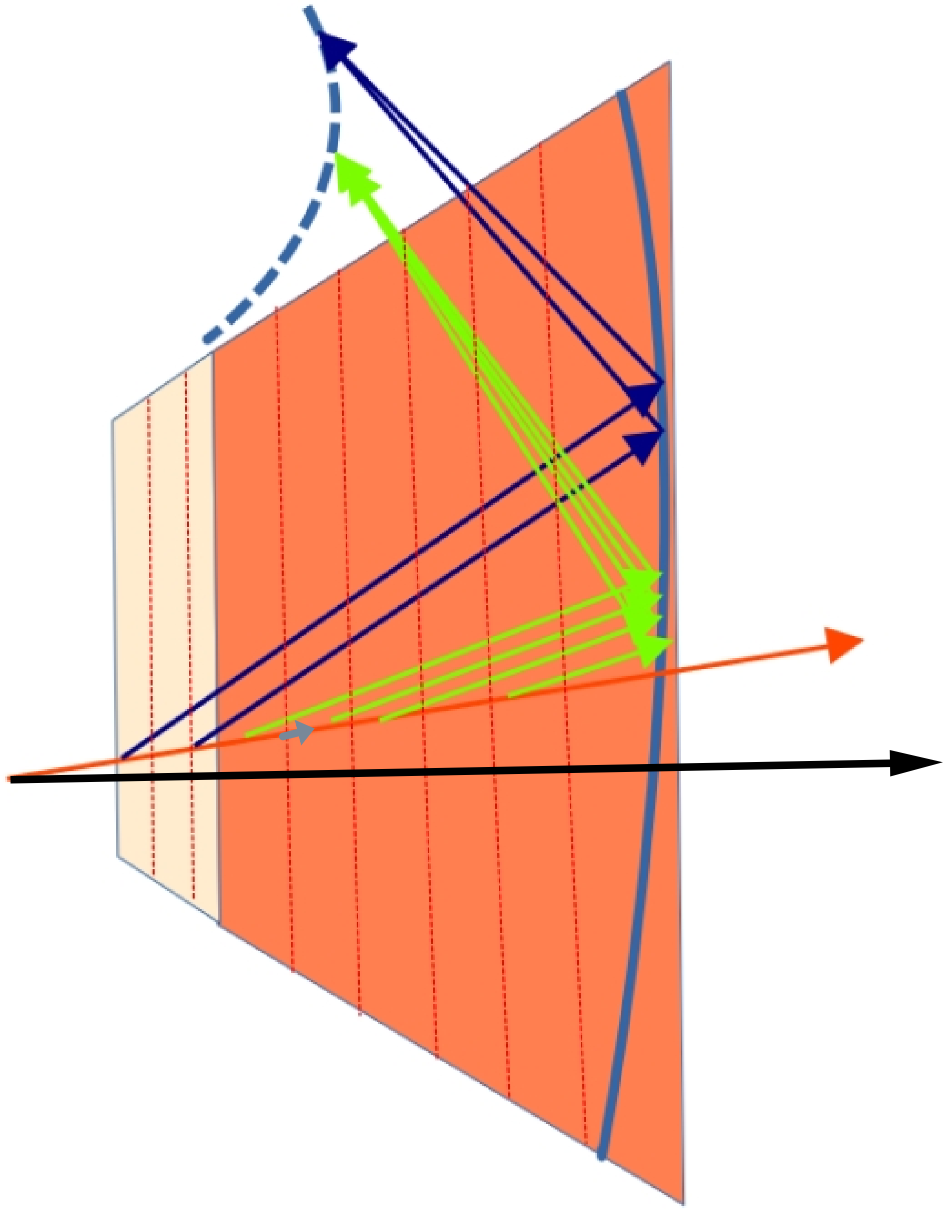
Current IRT in EICRecon

For each event
  For each track
    For each projection plane
      For each photon
        Runs to obtain a Polar and Azimuthal angle estimation.

solve(track_position, momentum_unit_vector,pixel_xy, beam dir,...)

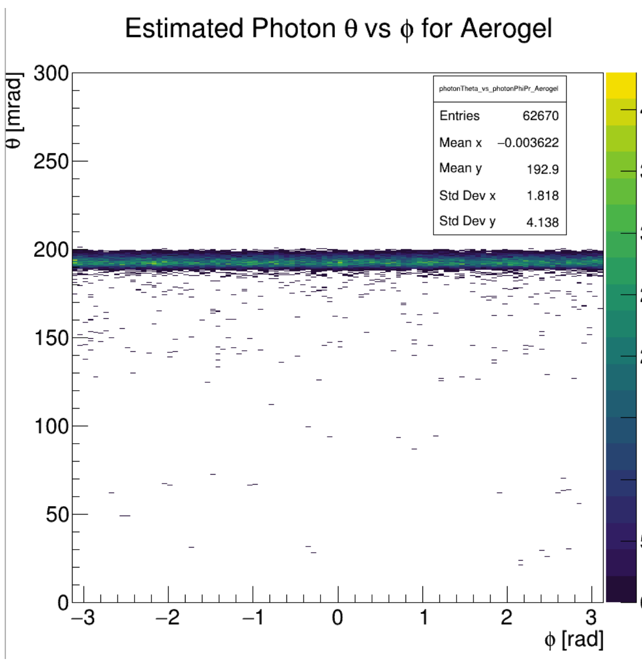
```
for(unsigned iq=0; iq<zdim+1; iq++) {
  auto &solution = solutions[iq] = irt->Solve(radiator->m_Locations[iq].first,
                 // FIXME: give beam line as a parameter;
                 radiator->m_Locations[iq].second.Unit(), phx, TVector3(0,0,1), false);
  if (!solution.Converged()) {
    all_converged = false;
    break;
  } //if
```

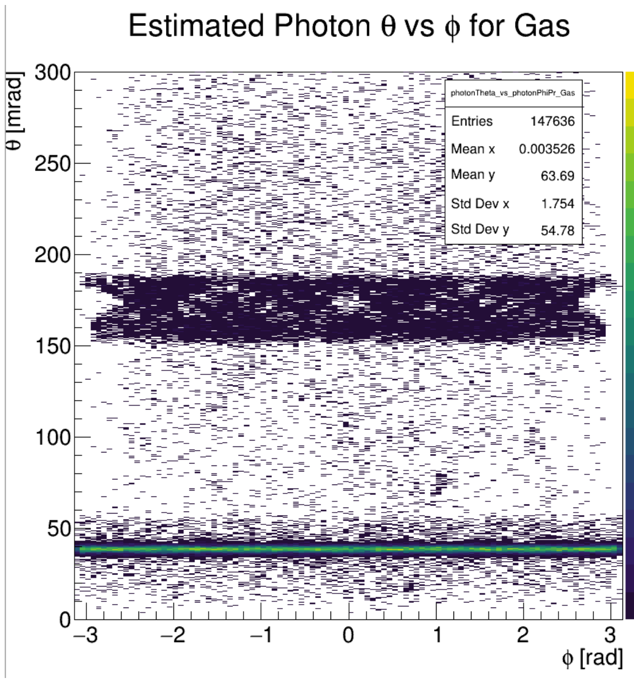
IRT

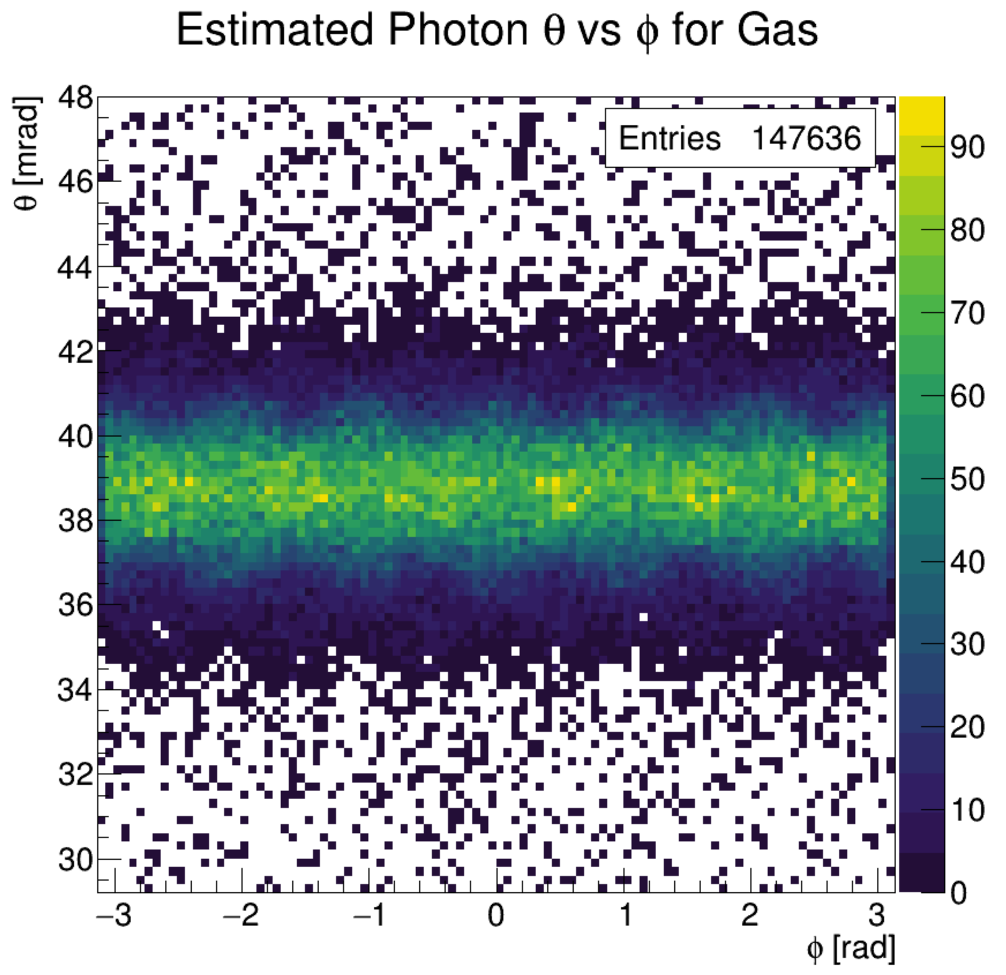


Analyse the reconstruction file!

3) IRT in EICRecon

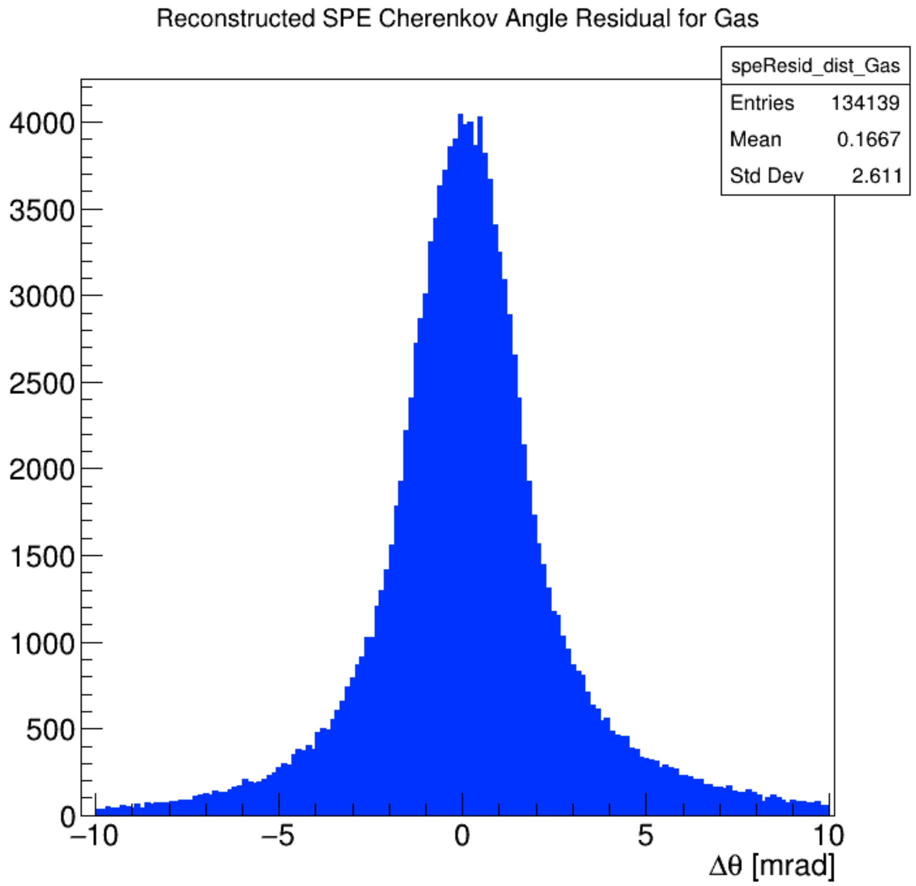


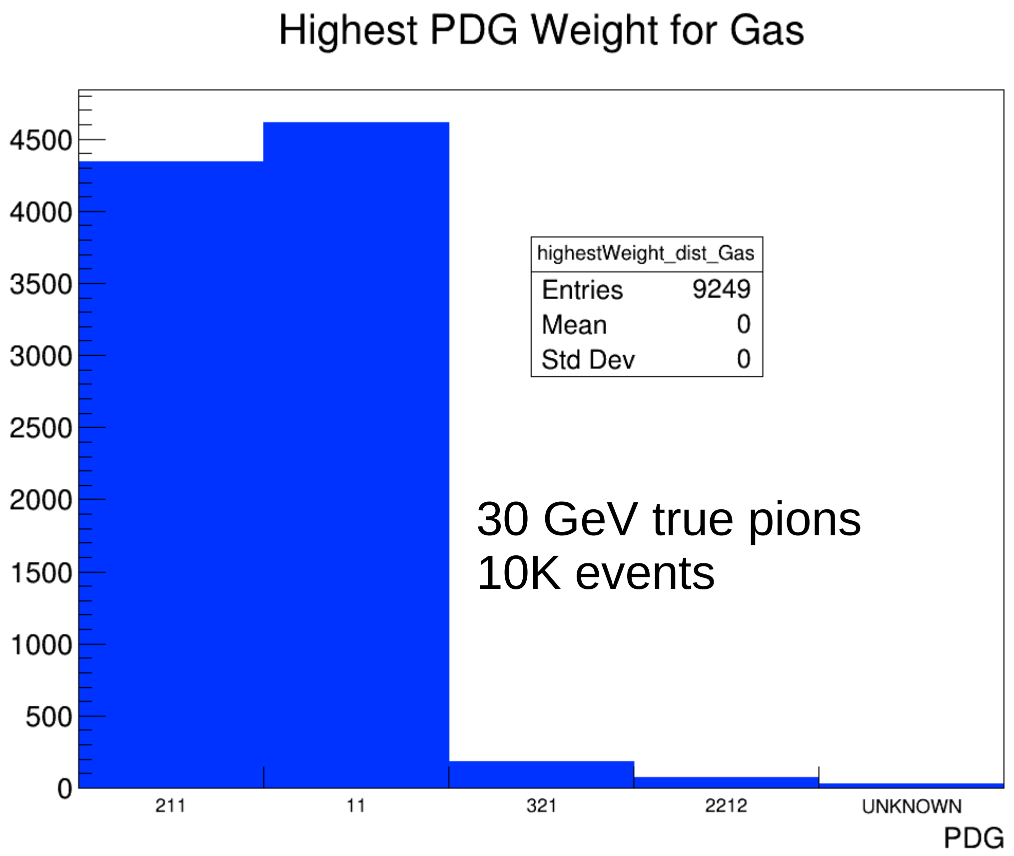




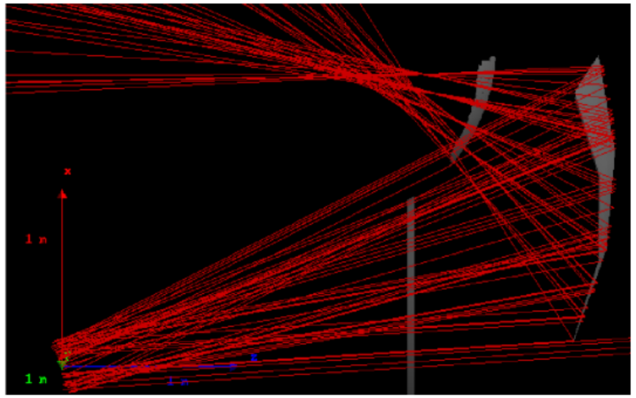
All photons are mixed. We see aerogel background for gas reconstruction → Good!
We see, aspherical aberration nodes in azimuth→ Excellent!

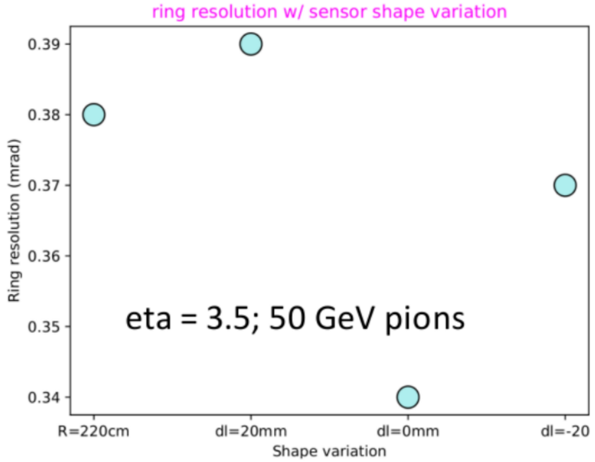
3) IRT in EICRecon

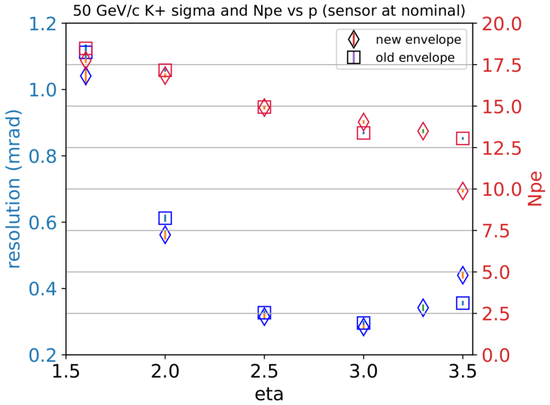


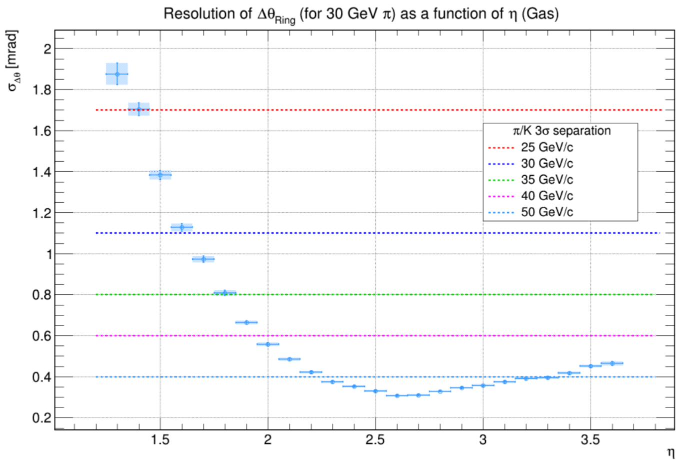


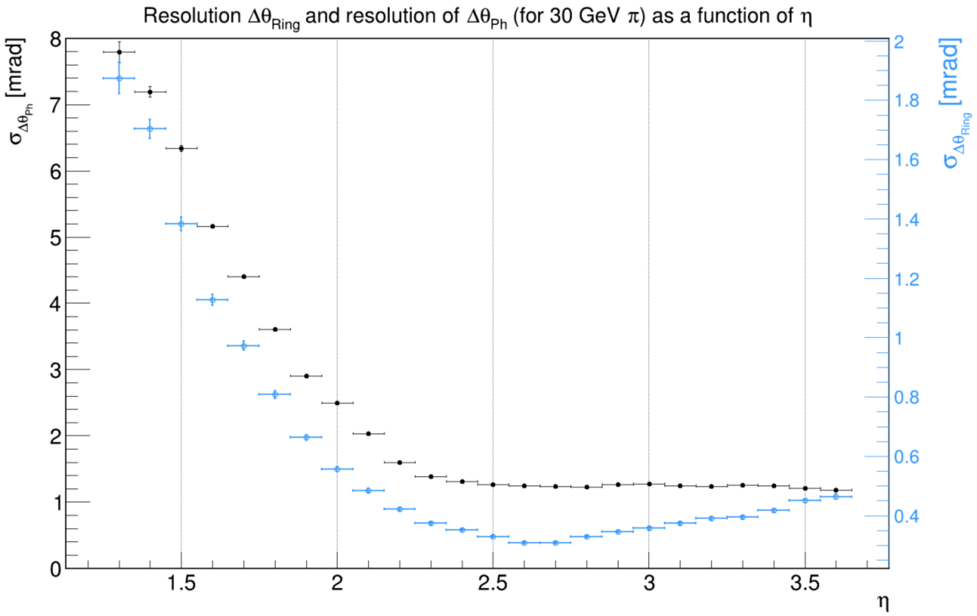
4) dRICH optimization using EICRecon

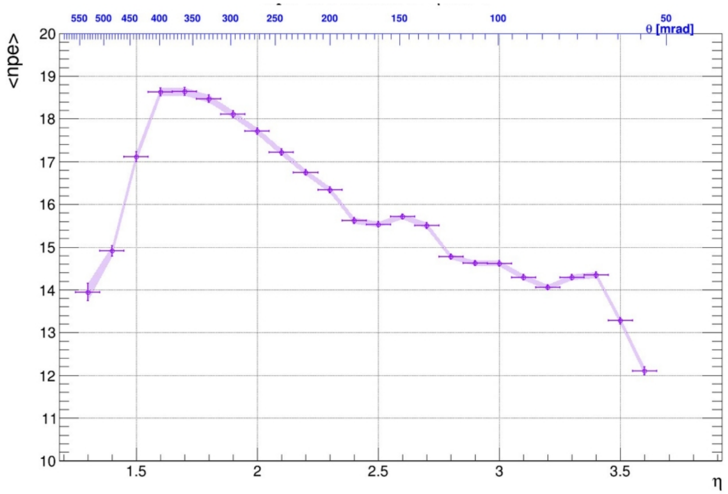


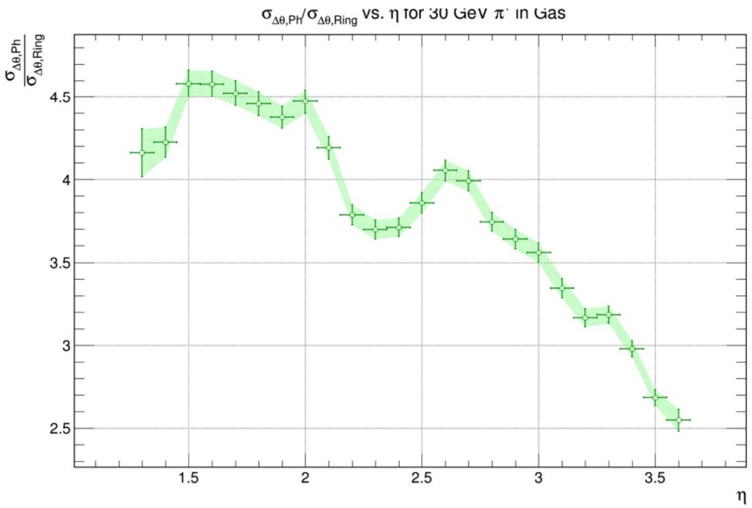




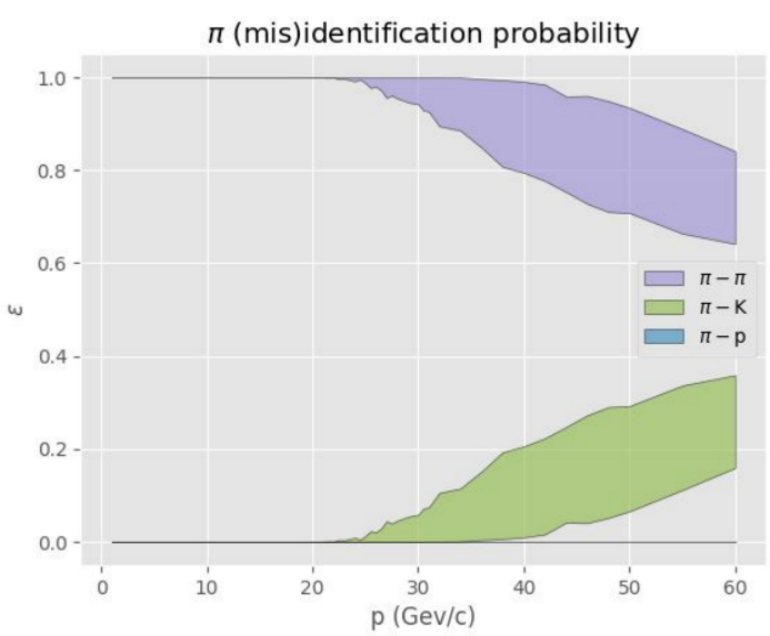


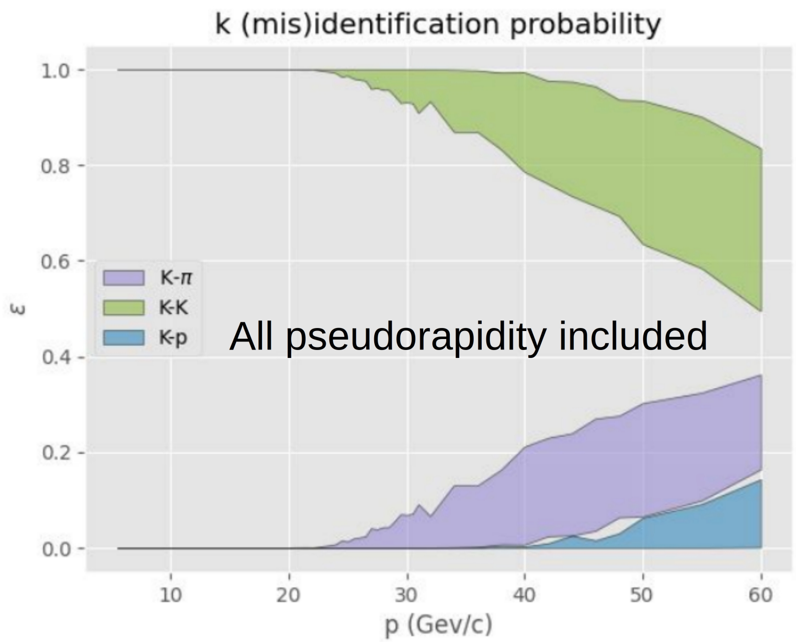




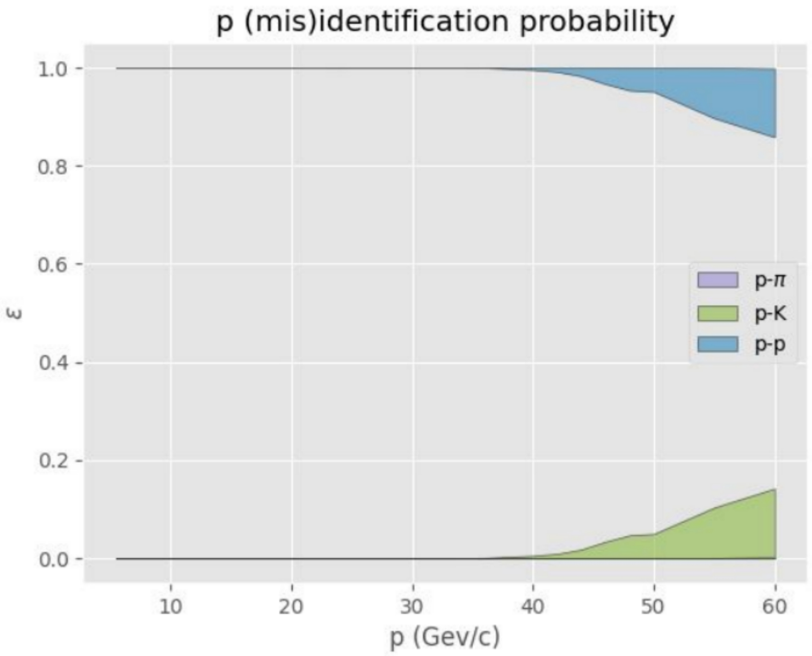


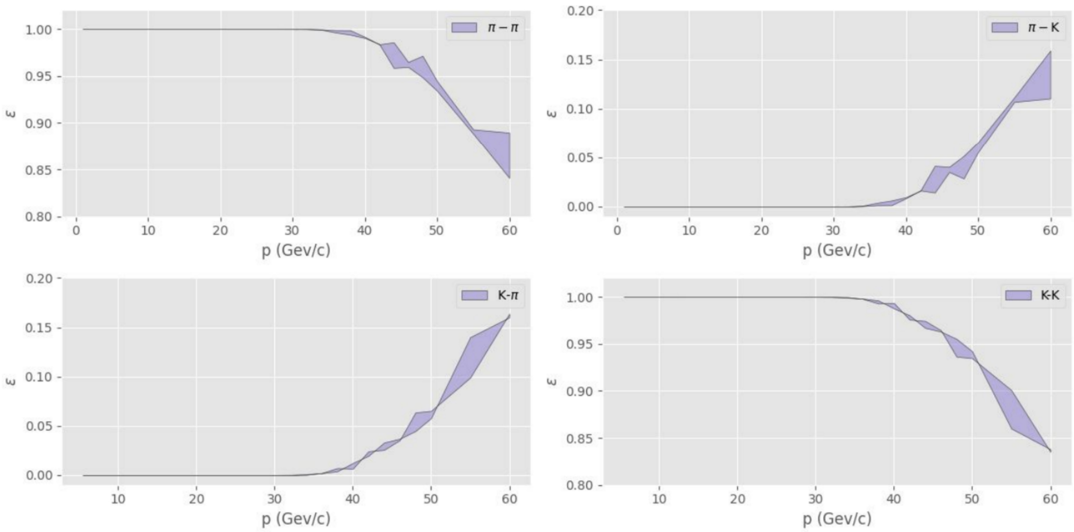
4) dRICH performance using EICRecon





All pseudorapidity included





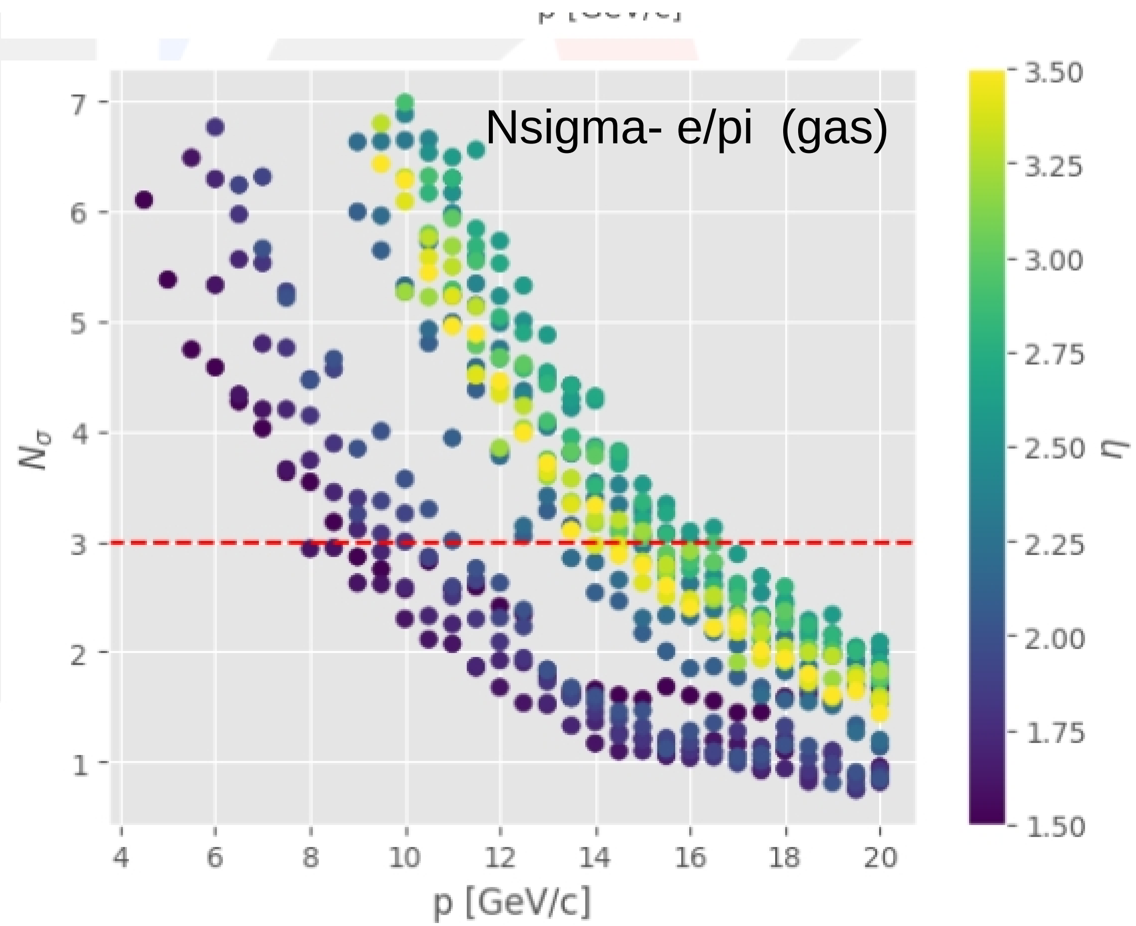
Only high pseudorapidity

Analysis details

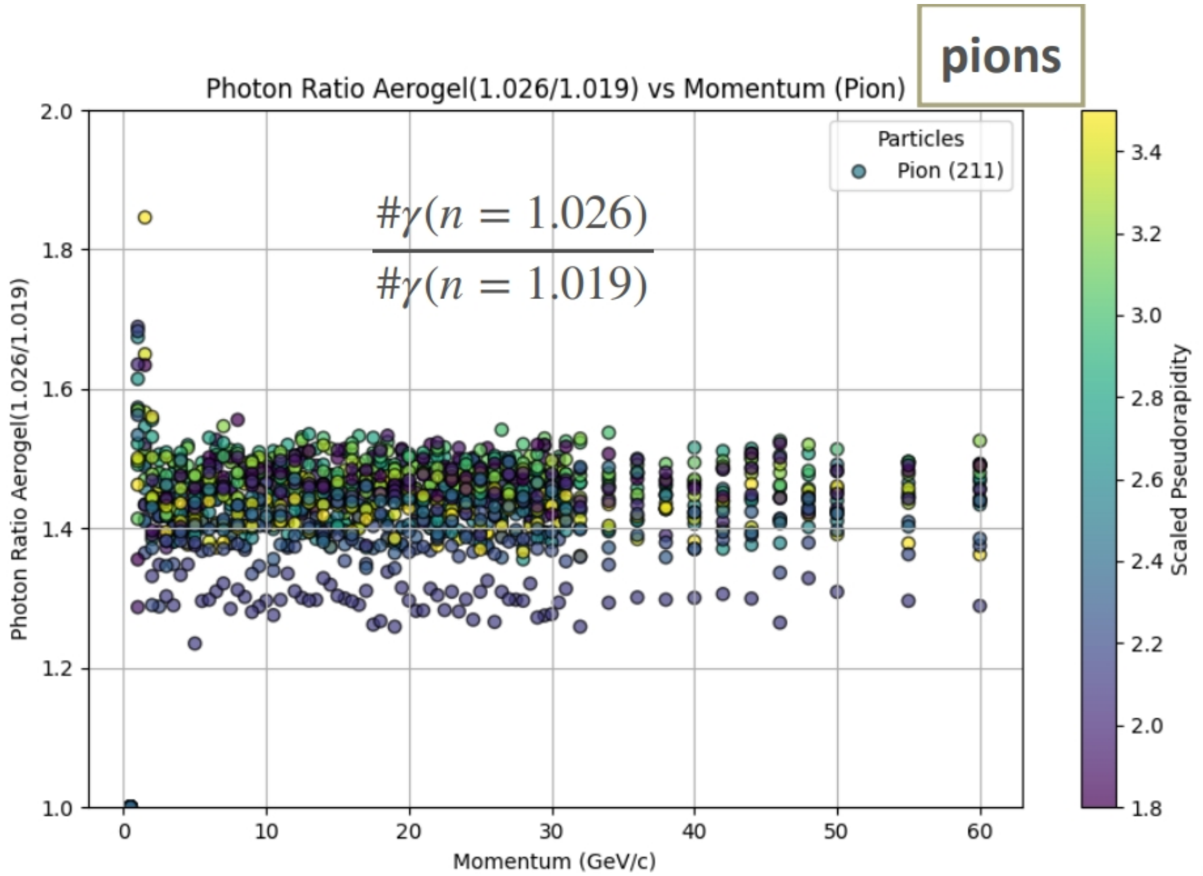
Two sets of simulations using **hepmc** files as input.

- HEPMC files (4662) located at:
  - https://github.com/deepaksamuel/hepmc-drich-eic/blob/main/README.md
- Each file (with fine kinematic bins) contains 2000 events but the simulation, due to time constraints, was done with the first 1000 events.
  - **PIDs:** [2212, 321 or 211]
  - **Momentum (74 bins):** [0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0, 5.5, 6.0, 6.5, 7.0, 7.5, 8.0, 8.5, 9.0, 9.5, 10.0, 10.5, 11.0, 11.5, 12.0, 12.5, 13.0, 13.5, 14.0, 14.5, 15.0, 15.5, 16.0, 16.5, 17.0, 17.5, 18.0, 18.5, 19.0, 19.5, 20.0, 20.5, 21.0, 21.5, 22.0, 22.5, 23.0, 23.5, 24.0, 24.5, 25.0, 25.5, 26.0, 26.5, 27.0, 27.5, 28.0, 28.5, 29.0, 29.5, 30.0, 30.5, 31.0, 32.0, 34.0, 36.0, 38.0, 40.0, 42.0, 44.0, 46.0, 48.0, 50.0, 55.0, 60.0]
  - **Pseudorapidity:** [1.5-3.5] in 0.1 steps
    - Example 1.5 ⇒ bin of 1.499 to 1.501
  - Bin details in
    - https://docs.google.com/spreadsheets/d/1gNh3giwhRENJi25z45lgeh5d7QwIw5Uy3lepISHCv9Y/edit?gid=2032570815#gid=2032570815

4) dRICH performance using EICRecon

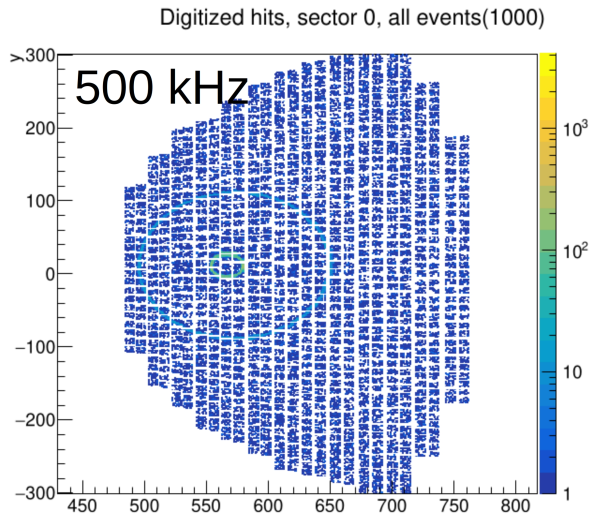


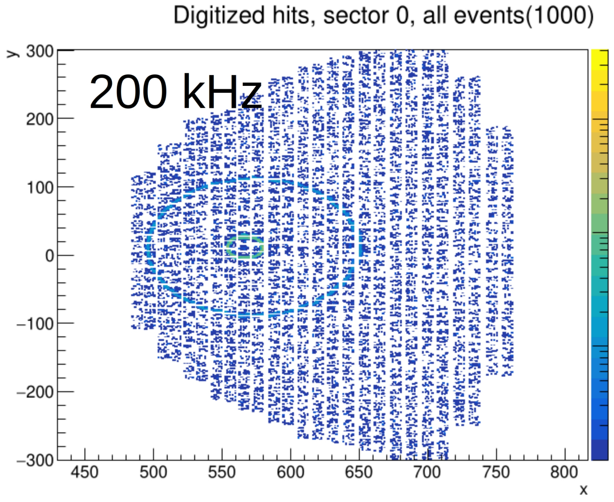
A potential case for a dual mirror implementation

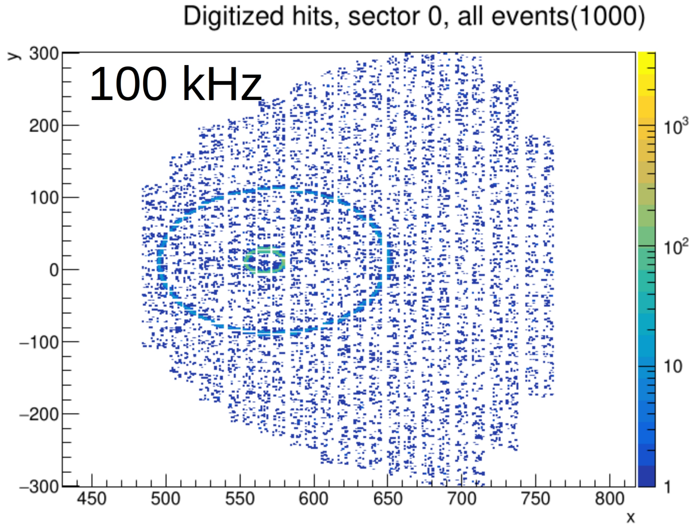


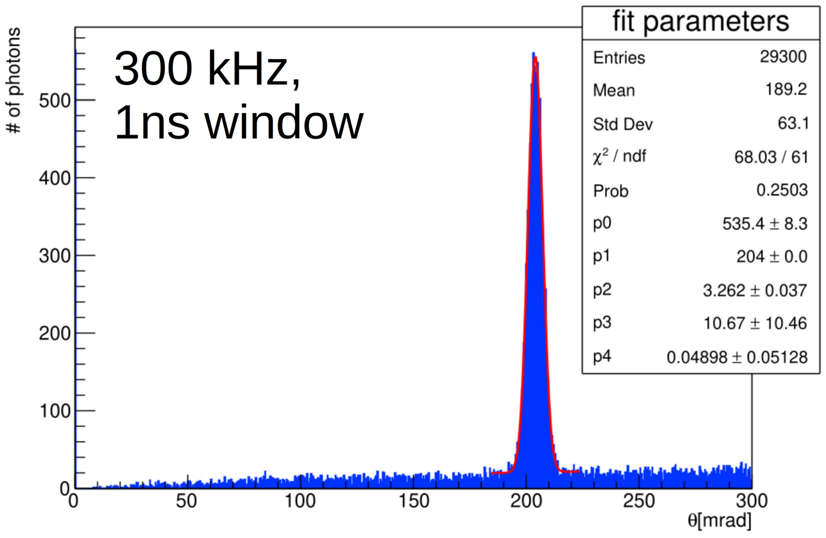
4) dRICH SiPM noise performance using EICRecon

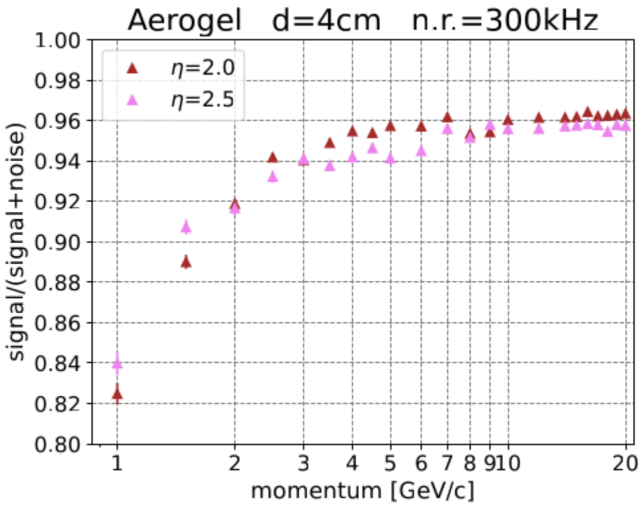
By default noise is OFF. We can turn it ON in EICRecon and need to make some tweaks to analyse

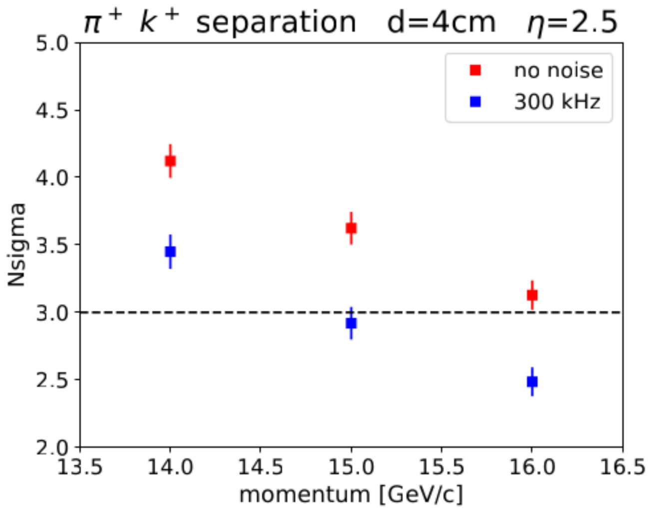












4) dRICH SiPM noise performance using EICRecon

By default noise is OFF. We can turn it ON in EICRecon and need to make some tweaks to analyse

dRICH: Data reduction Dataset

EPIC simulation pipeline:

HEPMC MC events → HEPMC_merger → FULL Geant4 → RECO EICRecon

PHYS signal e.g. DIS

PHYS bkg e - beamgas, p - beamgas

```
./SignalBackgroundMerger --signalFile dis.hepmc --signalFreq 0 \
        --bg1File hgas.hepmc --bg1Freq 31.9 \
        --bg2File egas.hepmc --bg2Freq 3177.25 \
```

Geant4 hits on detectors sensitive volume

root://dtn-eic.jlab.org//work/eic2/EPIC/FULL/24.05.0/epic_craterlake/BACKGROUNDS/MERGED/HEPMC_merger-1.0.2/10x100/RealisticSignalPerFrame/

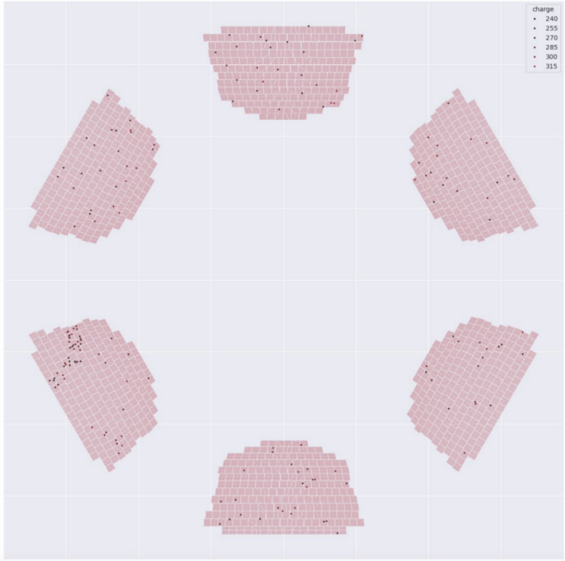
In dRICH case, applies digitization, quantum efficiency, safety factor. Possibly SiPM noise

By default SiPM noise disabled and not present on root://dtn-eic.jlab.org

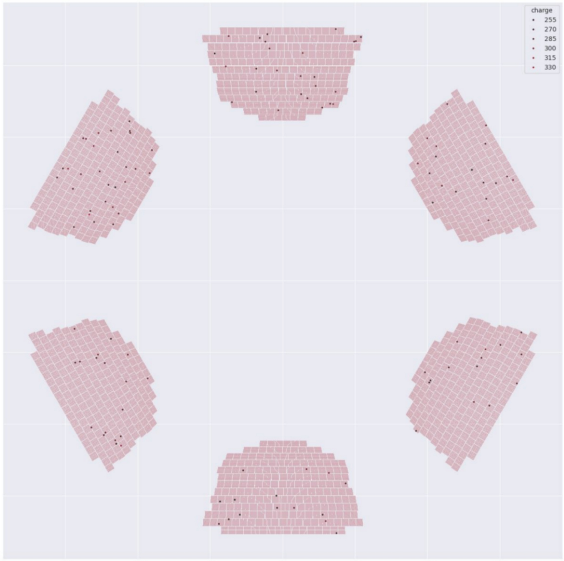
Options:

- Start from Merged FULL root files available on server and enable noise at RECO stage using *drich-dev/recon.rb* with configs (but only ~ 7k events)
- **Run the entire simulation pipeline ourselves, starting from HEPMC files.**
  - We now have around 200k events

Phys Signal+Phys Background+Noise



Noise Only

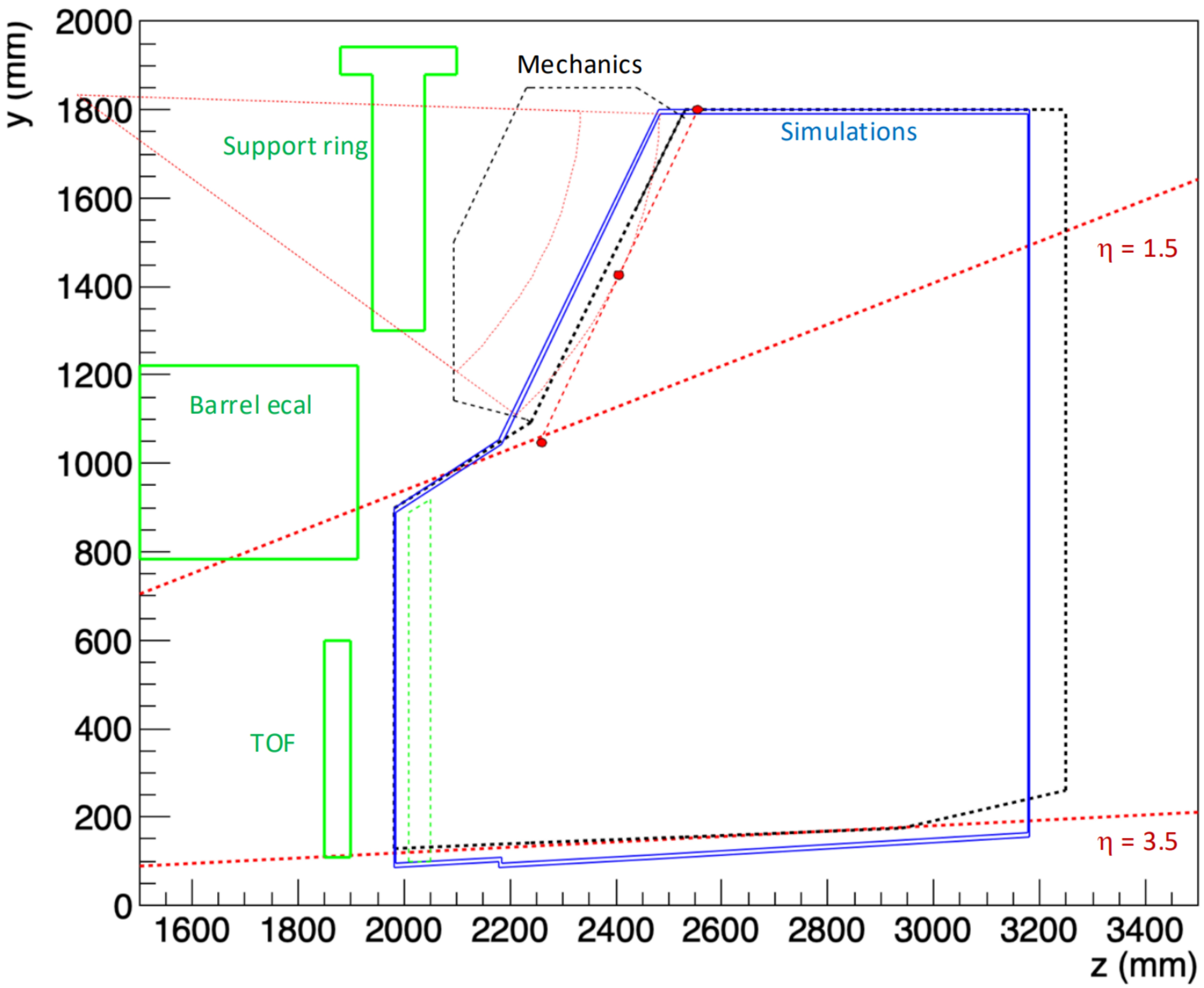


Data reduction at the DAM level (INFN Roma 1 and 2)

NOT USING IRT!!

Online filter can not be made with 1ns window.
Iterative process…
Large data to be reconstructed

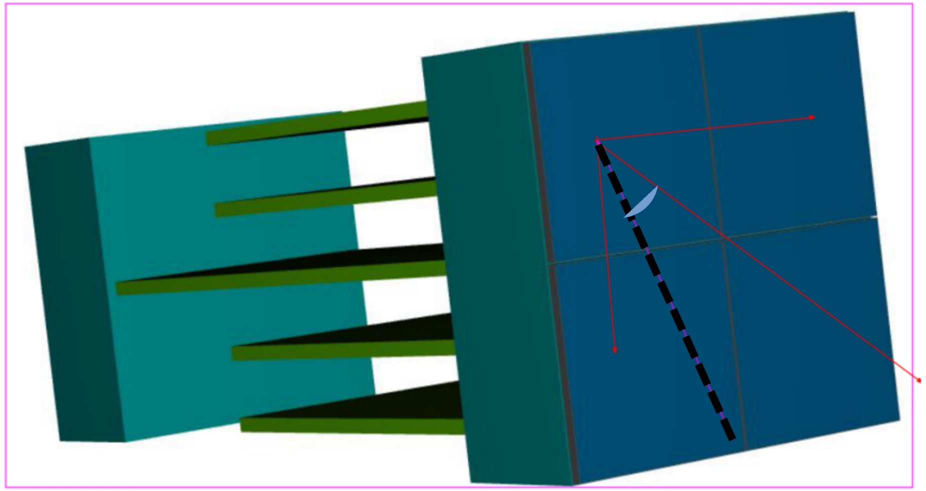
4) Further dRICH geometry optimization

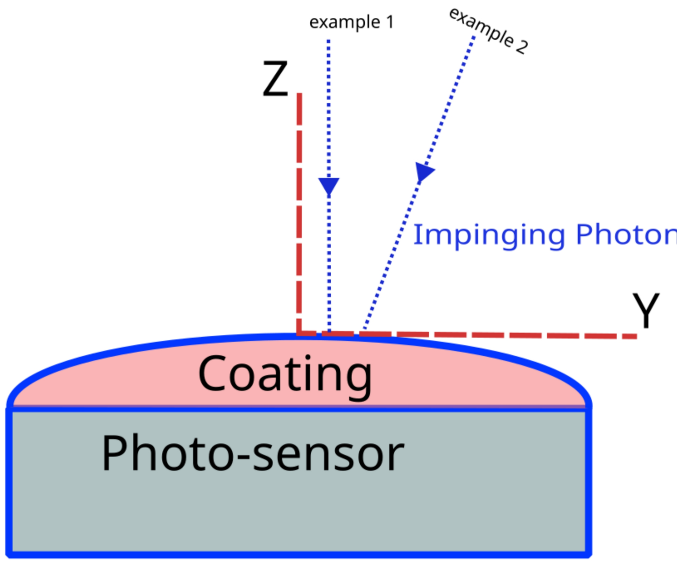


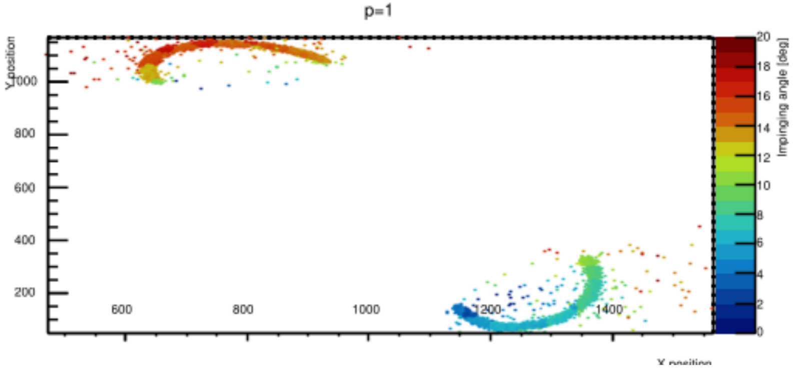
Critical changes in the dRICH geometry about to come!

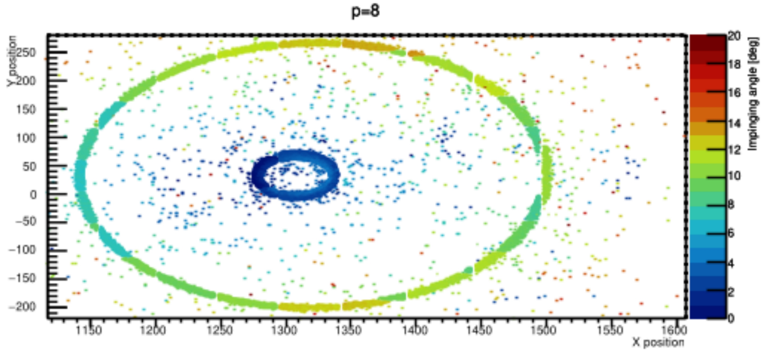
The idea is to optimize these changes with IRT outputs

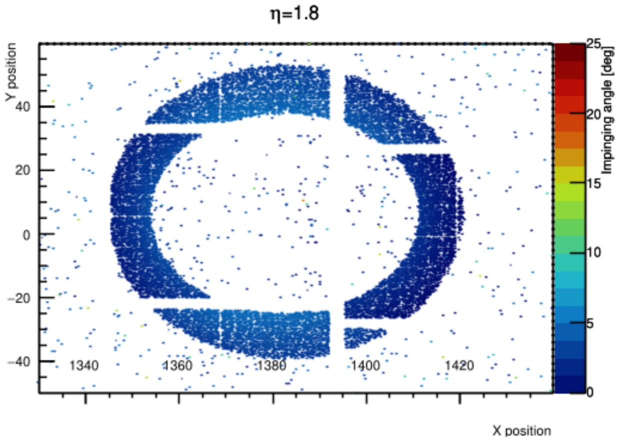
4) Microscopic studies of dRICH using EICRecon

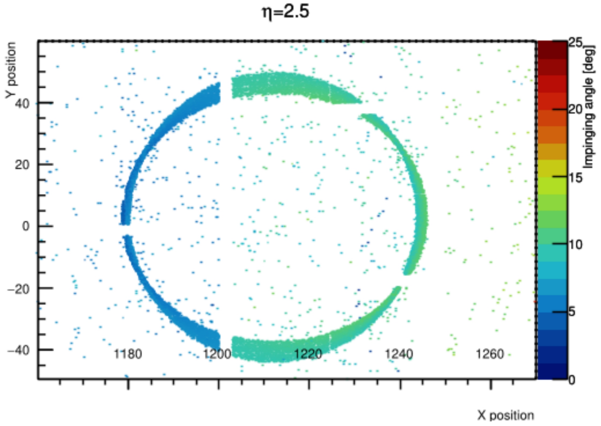


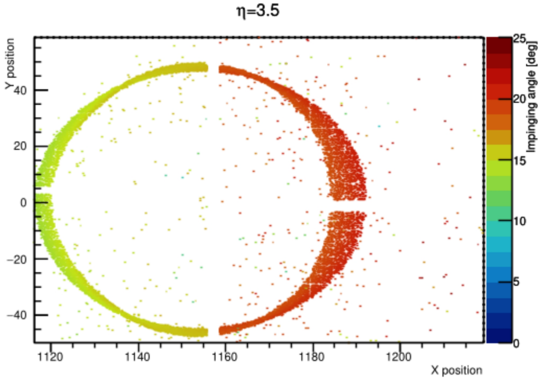










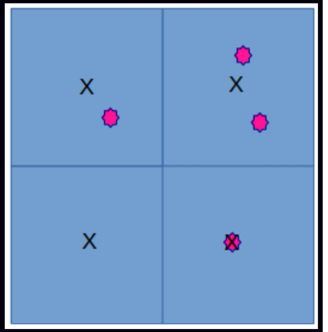


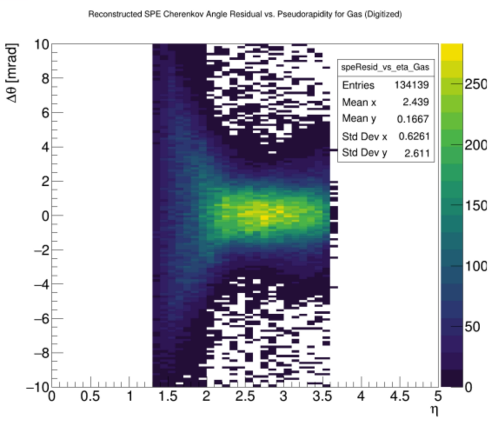
4) Microscopic studies of dRICH using EICRecon

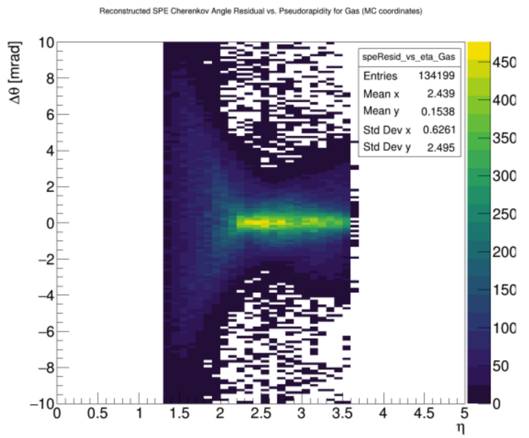
```
// get sensor and pixel info
// FIXME: signal and timing cuts (ADC, TDC, ToT
auto     cell_id   = raw_hit.getCellID();
uint64_t sensor_id = cell_id & m_cell_mask;
TVector3 pixel_pos = m_irt_det->m_ReadoutIDToPosition(cell_id);
//auto *irt_photon = new OpticalPhoton(); // new raw pointer; it will also be destroyed when `irt_particle` is destroyed

// trace logging
if(m_log->level() <= spdlog::level::trace) {
  m_log->trace("cell_id={:#X}  sensor_id={:#X}", cell_id, sensor_id);
  Tools::PrintTVector3(m_log, "pixel position", pixel_pos);
  if(mc_photon_found) {
    TVector3 mc_endpoint = Tools::PodioVector3_to_TVector3(mc_photon.getEndpoint());
    Tools::PrintTVector3(m_log, "photon endpoint", mc_endpoint);
    m_log->trace("{:>30} = {}", "dist( pixel,  photon )", (pixel_pos  - mc_endpoint).Mag());
  }
  else m_log->trace("  no MC photon found; probably a noise hit");
}

// start new IRT photon
auto *irt_sensor = m_irt_det->m_PhotonDetectors[0]; // NOTE: assumes one sensor type
auto *irt_photon = new OpticalPhoton(); // new raw pointer; it will also be destroyed when `irt_particle` is destroyed
irt_photon->SetVolumeCopy(sensor_id);
//irt_photon->SetDetectionPosition(Tools::PodioVector3_to_TVector3(mc_photon.getEndpoint()));
irt_photon->SetDetectionPosition(pixel_pos);
irt_photon->SetPhotonDetector(irt_sensor);
irt_photon->SetDetected(true);
```

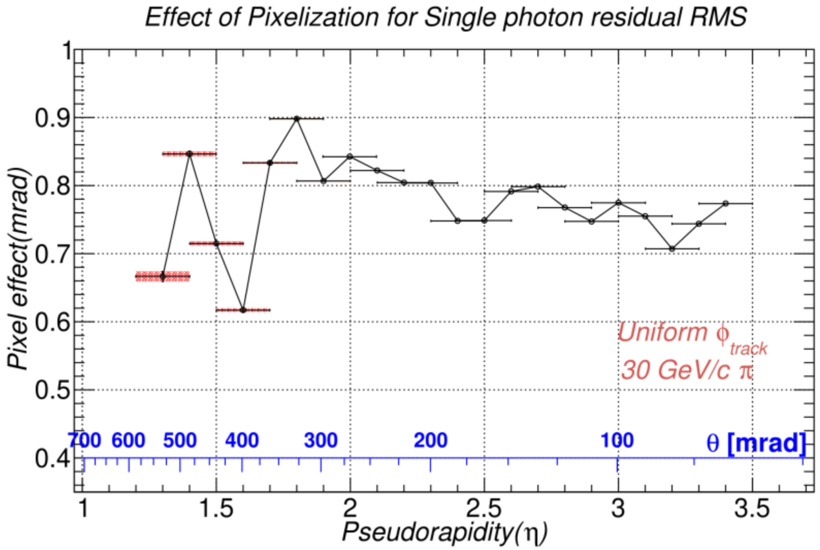
We know:

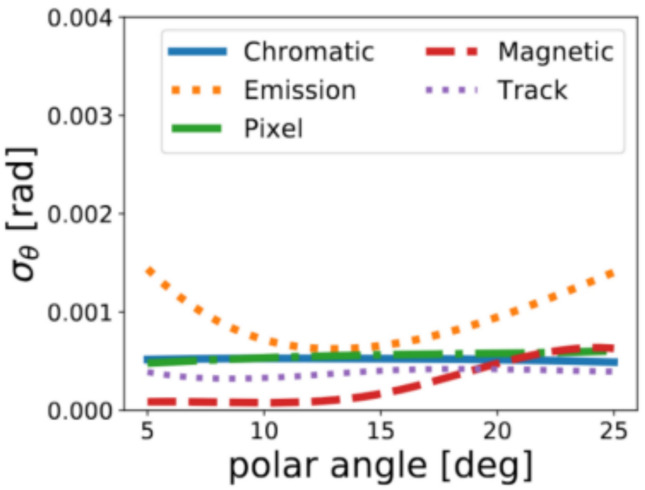
$$\sigma_{photon}^2 = \sigma_{cromatic}^2 + \sigma_{magnetic\text{-}field}^2 + \sigma_{Pixel}^2 + \sigma_{Emission\text{-}Point}^2 + \sigma_{track}^2$$







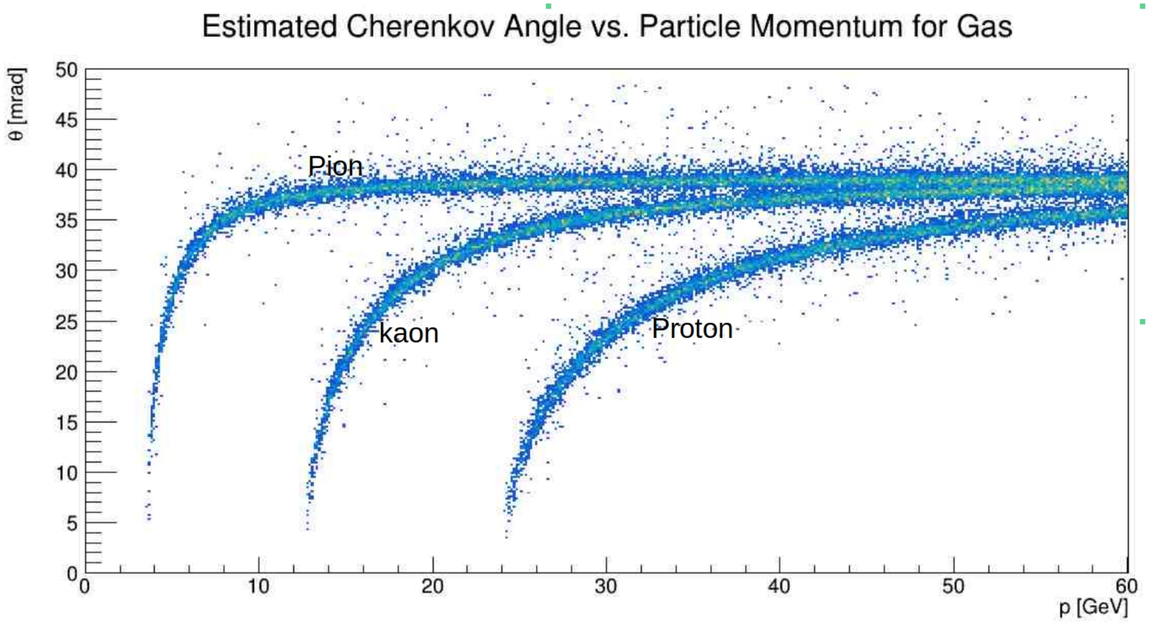


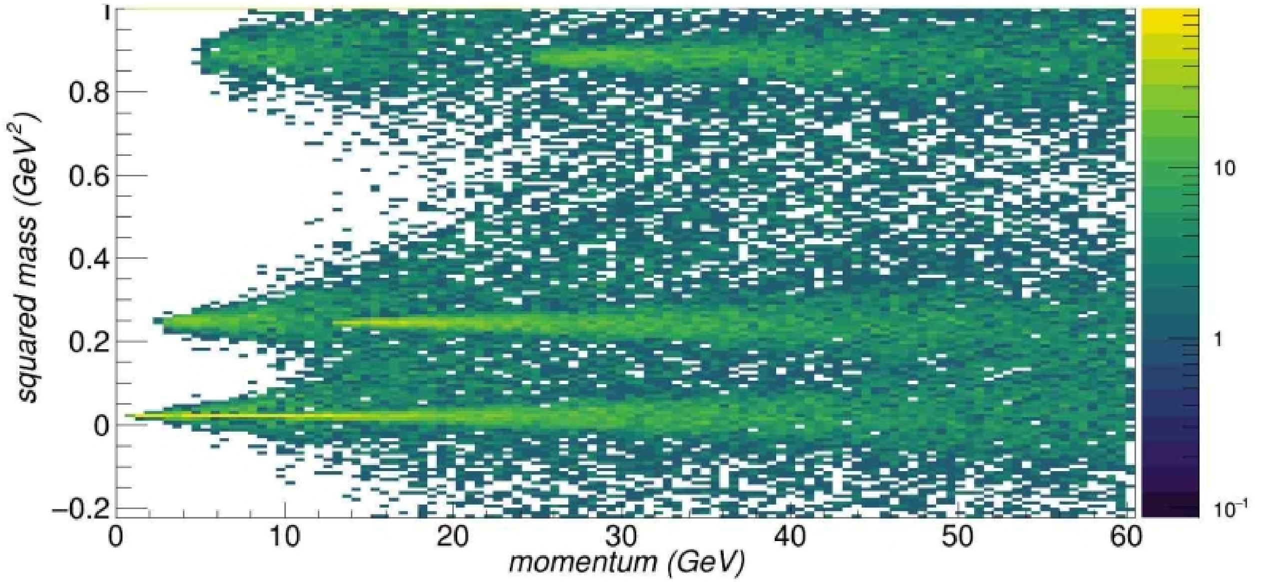


We don't know how can we study the effect of the magnetic field?

How can the effect from the track smearing be studied?

4) Current dRICH PID performance



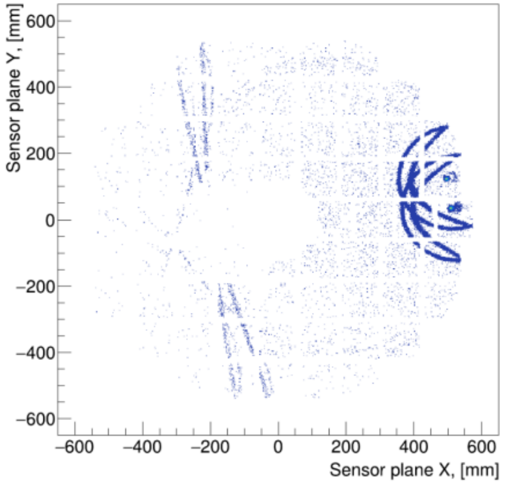


Jinky and myself are working on really understanding at which level current IRT stops working under multiparticle events.

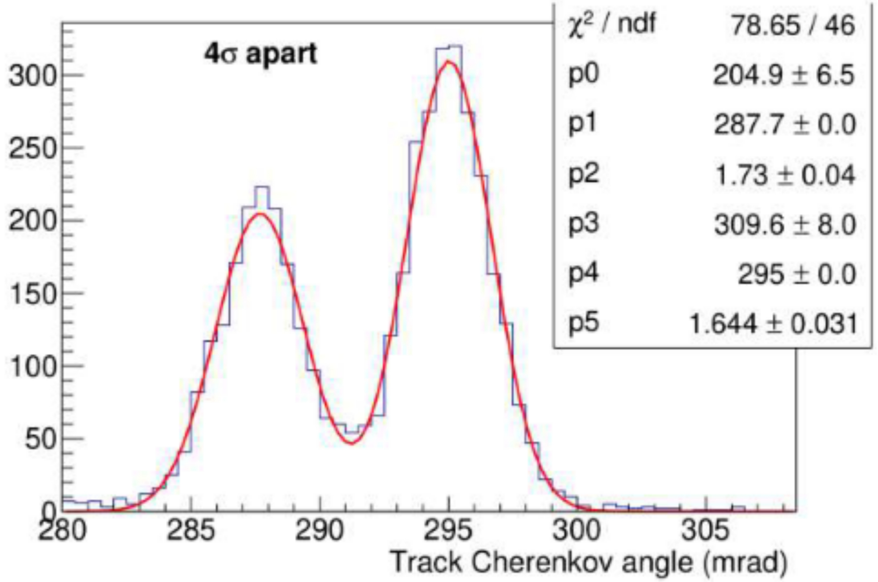
For physics cases, we may already start looking into single hadron cases.

5) Synergies between dRICH and pfRICH

- Both pfRICH and dRICH has IRT invloved to estimate single Cherenkov photon angle.
- Different mirrors are used in pfRICH, multiple optical paths. Allowing more complicated hit topology.
- Major change in the stand-alone IRT-v2 with respect to event based PID. dRICH definitely benefits from this.
- A dual mirror dRICH would come closer to pfRICH optics!
- A dual aerogel pfRICH would come closer to the dRICH radiator status.



In standalone studies it is demonstrated that IRT v-2 has clear advantages over IRT v-1.



5) Future Work plan

<u>dRICH+pfRICH Software Near Term Plans</u>

Based on Nov. 27 meeting (Chandra, Alexander, Brian, Gabor, others)

- Meeting Outcomes
  - Want closer coordination between dRICH and pfRICH software efforts (workforce and knowledge exchange)
  - Need short-to-medium term action plan to direct work

Software priorities and work plans

- Update pfRICH geometry in ePIC with necessary optical properties - Alexander
  - Continued maintenance of geometry handled by Bill Lee and Gabor (BNL)
- Validate existing IRT algorithm using pfRICH
  - Done after pfRICH geometry is updated
  - Ensure proper pfRICH information is being propagated to EICrecon
  - Can compare single particle results from EICrecon directly against the well understood standalone pfRICH model
- Implement and test IRT2 algorithm
  - Validate using both dRICH (simple reflection geometry) and pfRICH (complex reflection geometry)
- Catalogue needed changes to data model
  - While doing above, keep track of needed changes to the data model
  - Coordinate with S&C throughout this process
- Interface with S&C and Reconstruction groups on event reconstruction
  - Longer term goal
  - How do we integrate PID into holistic event reconstruction
  - Need POC from dRICH/pfRICH to interface with Reconstruction group
- Other activities
  - GPU acceleration for optical photon tracing (Gabor)
  - Modeling of thin anti-reflective coating for sensor windows

a) For dRICH we want to exploit the timing performance of the SiPM.
We are yet not sure how to include ADC and TDC information.

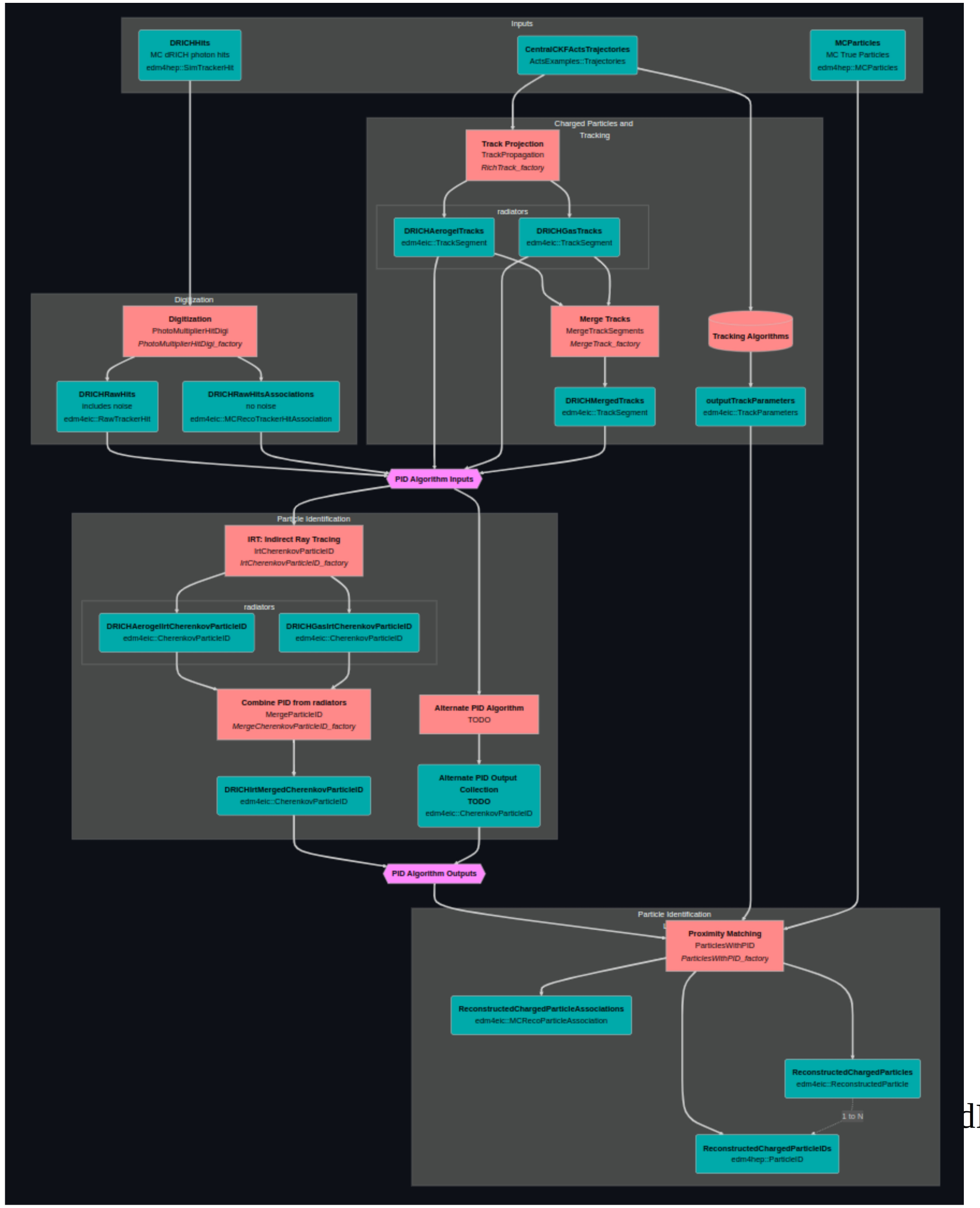
b) We would also like to model our noise according to the background level in different region of the SiPM panels.

Conclusions

- We are in a better shape in terms of workforce for the reconstruction side.
- We have made substantial dRICH characterization and understand the performance in many microscopic aspect.
- The baseline IRT has been extremely useful and We can also handle it within EICRecon.
- Sophisticated PID is available in IRT v-2 and we are teaming up with our pfRICH colleagues to work together.
  - We have quite a bit of understanding in the dRICH with existing framework. Essential for the first phase of validation.
  - We have several commonalities, addressing similar issues are easier.
- EICRecon digitization, QE scaling, noise framework is used by other INFN colleagues. Bulk data reconstruction, right definition of timing information is central for their studies to inject right amount of noise for filtering
- New changes in the dRICH geometry is foreseen, IRT-EICrecon will be used for the validation purpose.

Back ups

3) How is IRT interfaced with EICRecon



EICrecon / src / detectors / DRICH / **DRICH.cc**

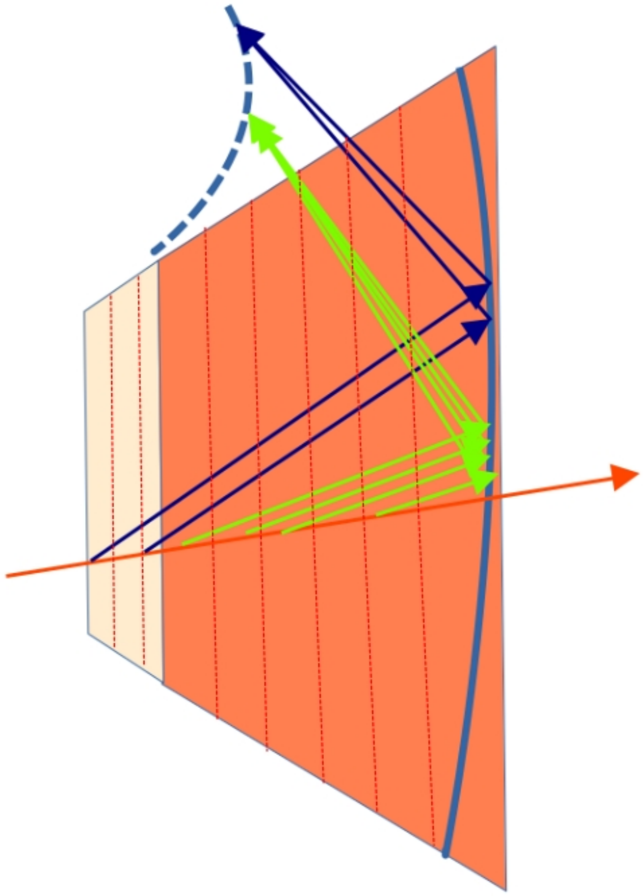
Code Blame 242 lines (222 loc) · 8.7 KB

```
void InitPlugin(JApplication *app) {
  // configuration parameters ///////////////////////////////////////////////

  // digitization
  PhotoMultiplierHitDigiConfig digi_cfg;
  digi_cfg.seed            = 5; // FIXME: set to 0 for a 'unique' seed, but
                                // that seems to delay the RNG from actually randomizing
  digi_cfg.hitTimeWindow   = 20.0; // [ns]
  digi_cfg.timeResolution  = 1/16.0; // [ns]
  digi_cfg.speMean         = 80.0;
  digi_cfg.speError        = 16.0;
  digi_cfg.pedMean         = 200.0;
  digi_cfg.pedError        = 3.0;
  digi_cfg.enablePixelGaps = true;
  digi_cfg.safetyFactor    = 0.7;
  digi_cfg.enableNoise     = false;
  digi_cfg.noiseRate       = 20000; // [Hz]
  digi_cfg.noiseTimeWindow = 20.0 * dd4hep::ns; // [ns]
  digi_cfg.quantumEfficiency.clear();
  digi_cfg.quantumEfficiency = { // wavelength units are [nm]
    {315,  0.00},
    {325,  0.04},
    {340,  0.10},
    {350,  0.20},
    {370,  0.30},
    {400,  0.35},
    {450,  0.40},
    {500,  0.38},
    {550,  0.35},
    {600,  0.27},
    {650,  0.20},
    {700,  0.15},
    {750,  0.12},
    {800,  0.08},
    {850,  0.06},
    {900,  0.04},
    {1000, 0.00}
  };
```

A couple of comments

1. We have **More than one** reconstructions involved.
a) The reconstruction of Cherenkov photon's polar and azimuthal angle.
b) The reconstruction of the event for the PID information.

2. The



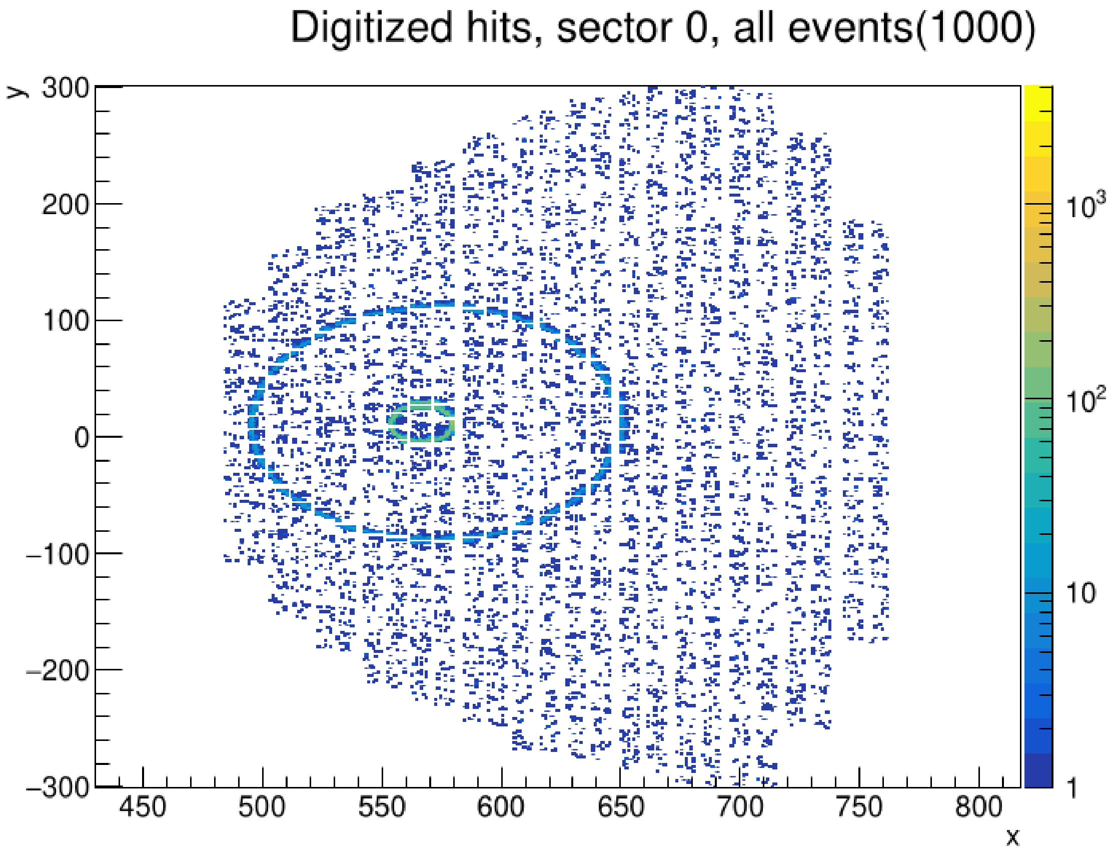
3) How is IRT interfaced with EICRecon

EICrecon / src / services / geometry / richgeo / IrtGeo.cc

Code | Blame | 95 lines (84 loc) · 3.73 KB

```cpp
// define the `cell ID -> pixel position` converter, correcting to sensor surface
void richgeo::IrtGeo::SetReadoutIDToPositionLambda() {

  m_irtDetector->m_ReadoutIDToPosition = [
    &m_log = this->m_log, // capture logger by reference
    // capture instance members by value, so those owned by `this` are not mutable here
    cell_mask   = this->m_irtDetector->GetReadoutCellMask(),
    converter   = this->m_converter,
    sensor_info = this->m_sensor_info
  ] (auto cell_id) {
    // decode cell ID to get the sensor ID and pixel volume centroid
    auto sensor_id = cell_id & cell_mask;
    auto pixel_volume_centroid = (1/dd4hep::mm) * converter->position(cell_id);
    // get sensor info
    auto sensor_info_it = sensor_info.find(sensor_id);
    if(sensor_info_it == sensor_info.end()) {
      m_log->warn("cannot find sensor ID {} in IrtGeo; using pixel volume centroid instead",sensor_id);
      return TVector3( pixel_volume_centroid.x(), pixel_volume_centroid.y(), pixel_volume_centroid.z());
    }
    auto sensor_obj = sensor_info_it->second;
    // get pixel surface centroid, given sensor surface offset w.r.t centroid
    auto pixel_surface_centroid  = pixel_volume_centroid + sensor_obj.surface_offset;
    // cross check: make sure pixel and sensor surface centroids are close enough
    auto dist = sqrt((pixel_surface_centroid - sensor_obj.surface_centroid).Mag2());
    if( dist > sensor_obj.size / sqrt(2) )
      m_log->warn("dist(pixel,sensor) is too large: {} mm",dist);
    return TVector3( pixel_surface_centroid.x(), pixel_surface_centroid.y(), pixel_surface_centroid.z());
  };
```

Cell-IDs are converted into Pixel Centroid coordinates



Hit map with added SiPM noise!