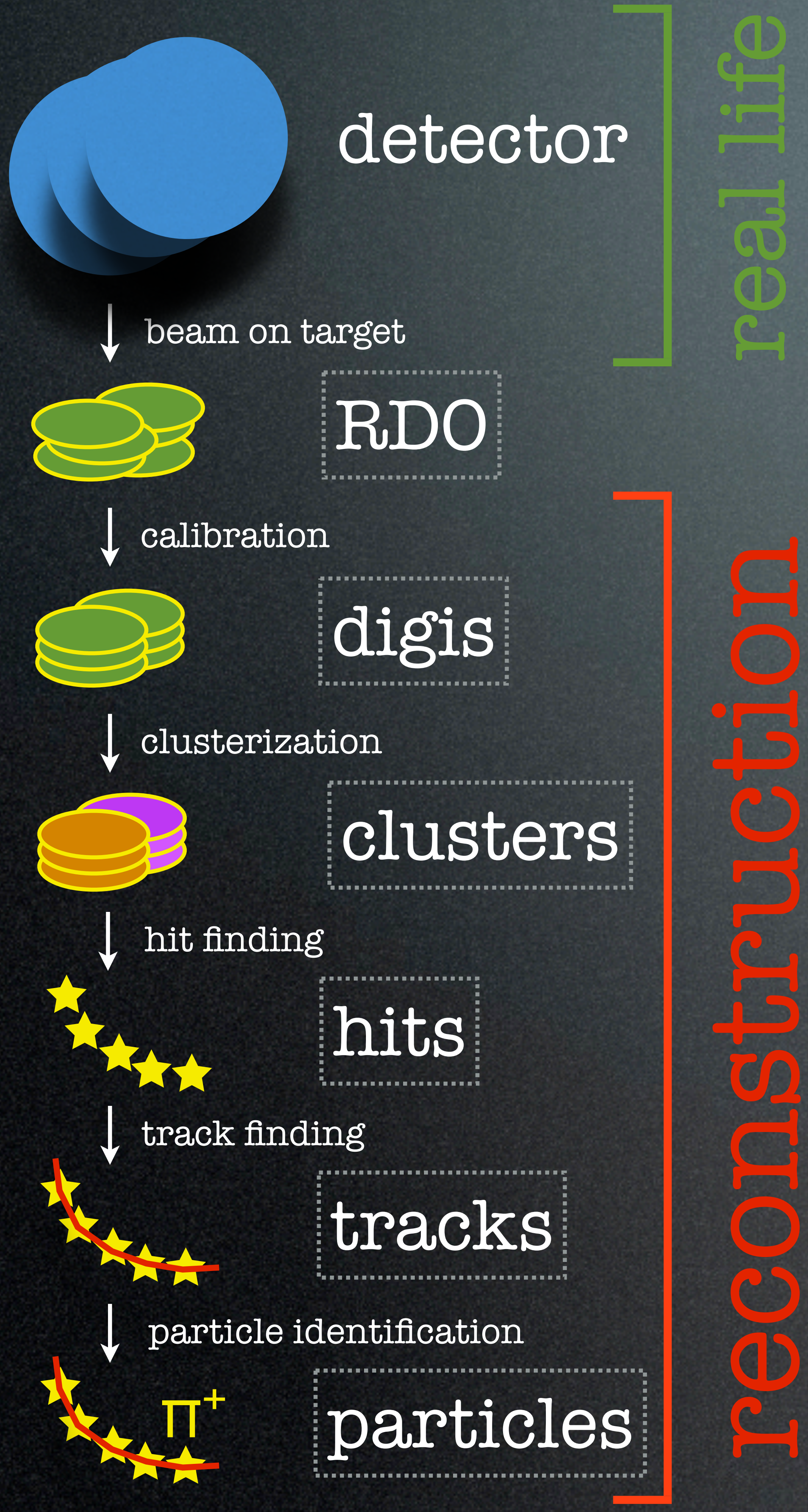
GEM Tracker

Radoslaw Karabowicz, GSI

INTRODUCTION



real life

reconstruction

detector

↓ beam on target

RDO

↓ calibration

digis

↓ clusterization

clusters

↓ hit finding

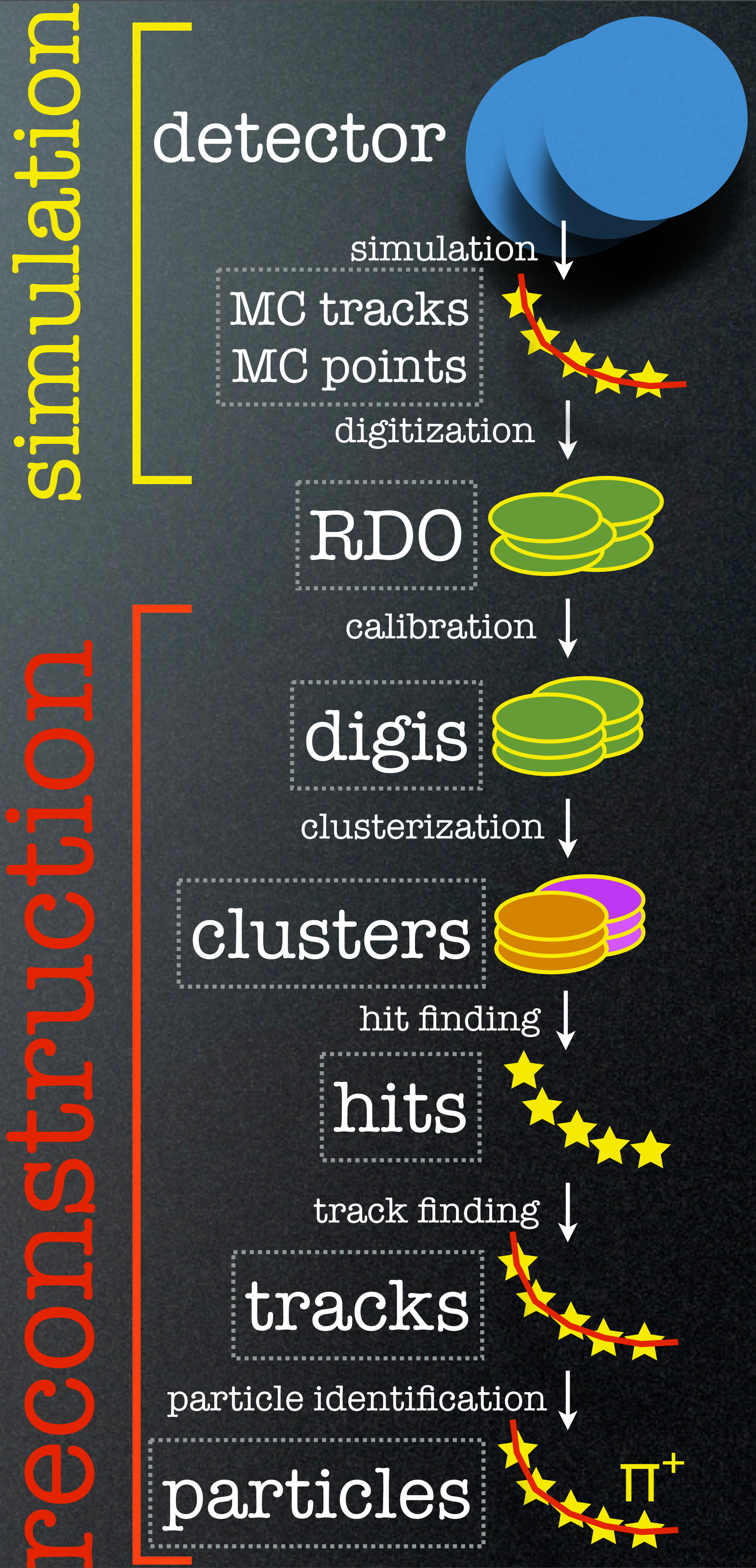
hits

↓ track finding

tracks

↓ particle identification

particles



simulation

reconstruction

detector

simulation ↓

MC tracks
MC points

digitization ↓

RDO

calibration ↓

digis

clusterization ↓

clusters

hit finding ↓

hits

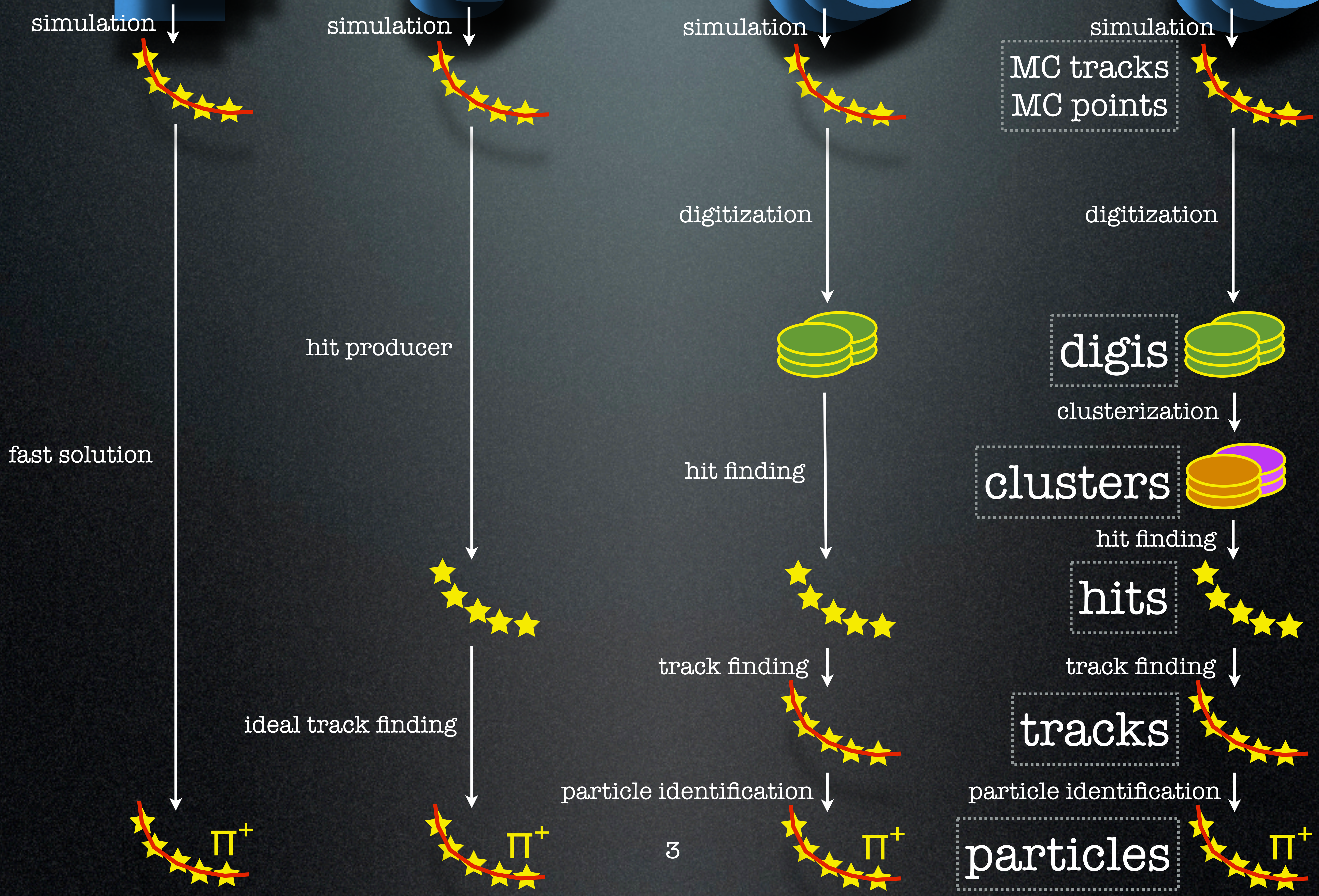
track finding ↓

tracks

particle identification ↓

particles

Implementation stages



simulation

fast solution

π^+

simulation

hit producer

ideal track finding

π^+

simulation

digitization

hit finding

track finding

particle identification

π^+

simulation

MC tracks
MC points

digitization

digis

clusterization

clusters

hit finding

hits

track finding

tracks

particle identification

particles

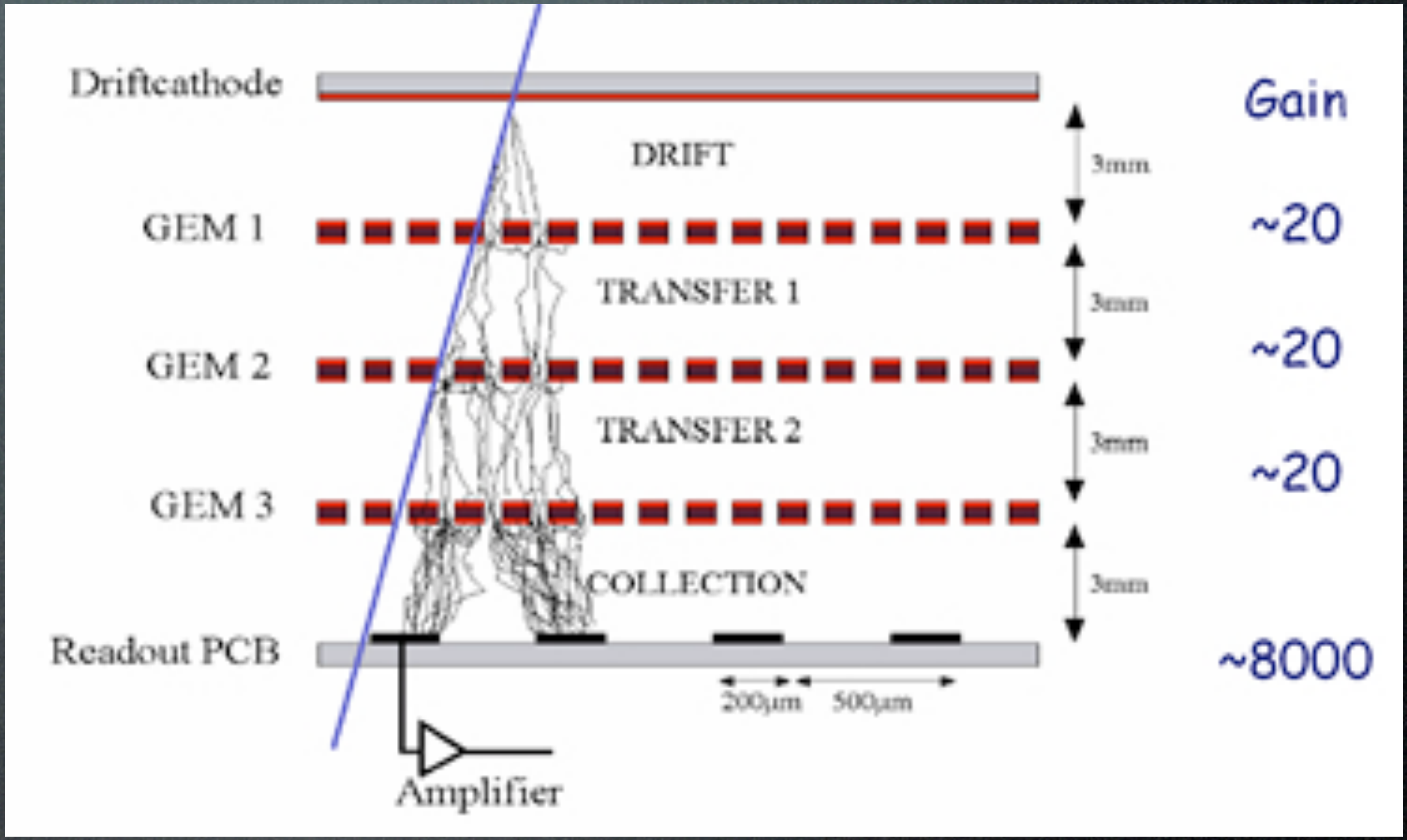
π^+

Plan

- Introduction
- Simulation
- Digitization
- Cluster finding
- Hit finding
- Tracking

GEM - what is this

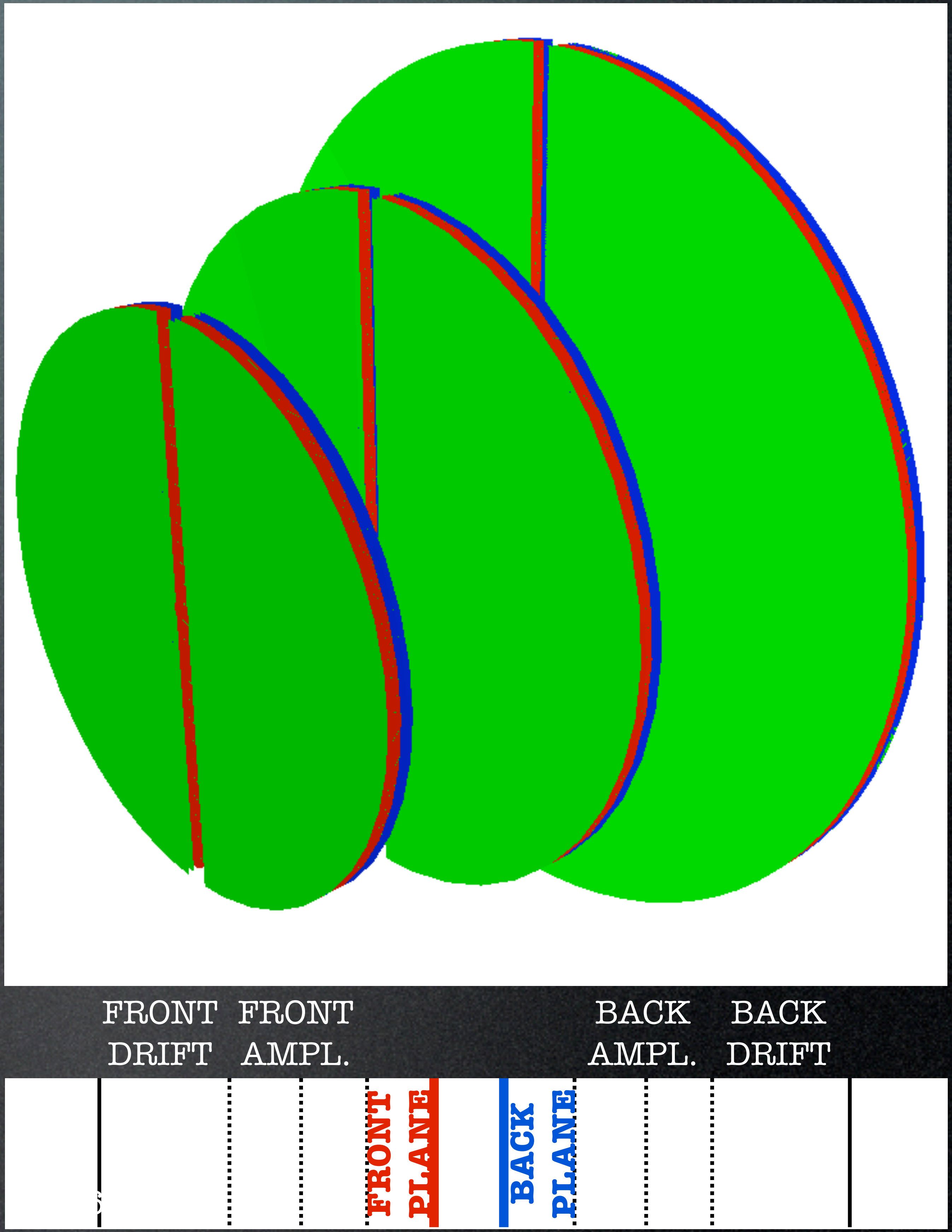
- GEM - Gas Electron Multiplier
- readout plane divided into strips (200 µm width)



- two different perpendicular strip orientations per readout plane implemented
- records trajectory position

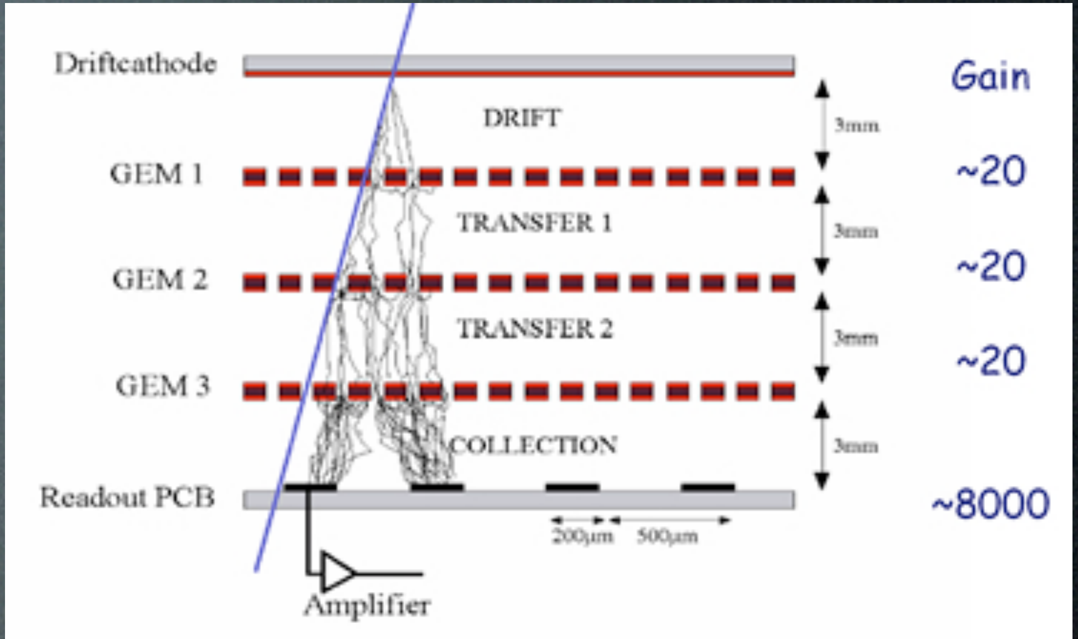
Tracker - what is this

- record track position at 3 stations at $z \approx 120,150,190$cm
- each station has two drift volumes and two sensitive planes: front and back (and therefore 4 different strip orientations)



Simulation

PndGemDetector
ProcessHits



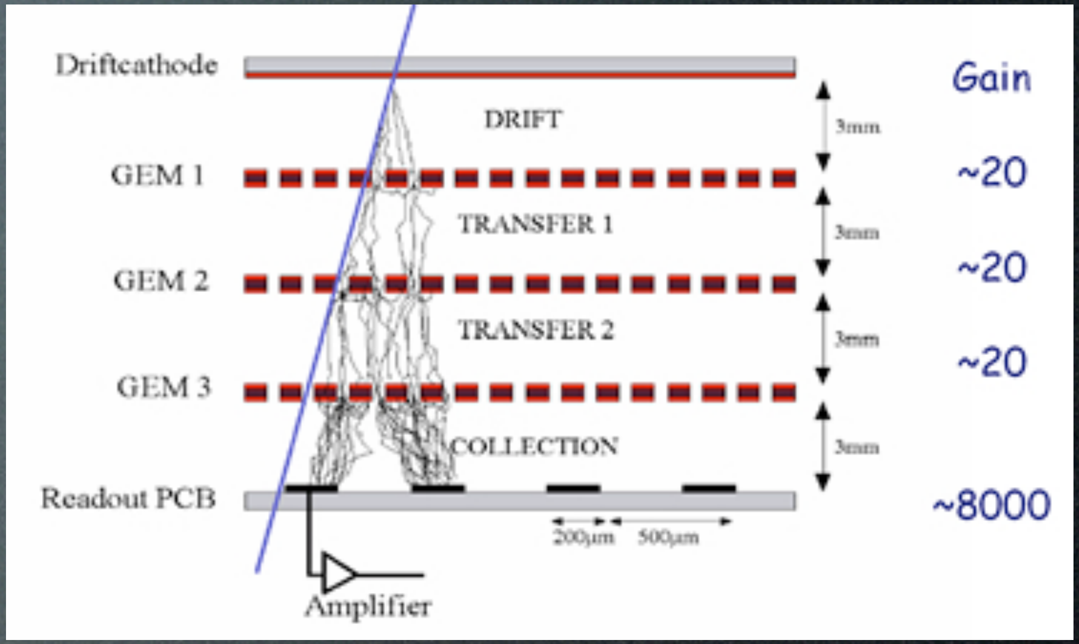
- Create MC points:
  - particle entrance point into the DRIFT volume
  - particle exit point from the DRIFT volume
  - particle energy loss in the DRIFT volume

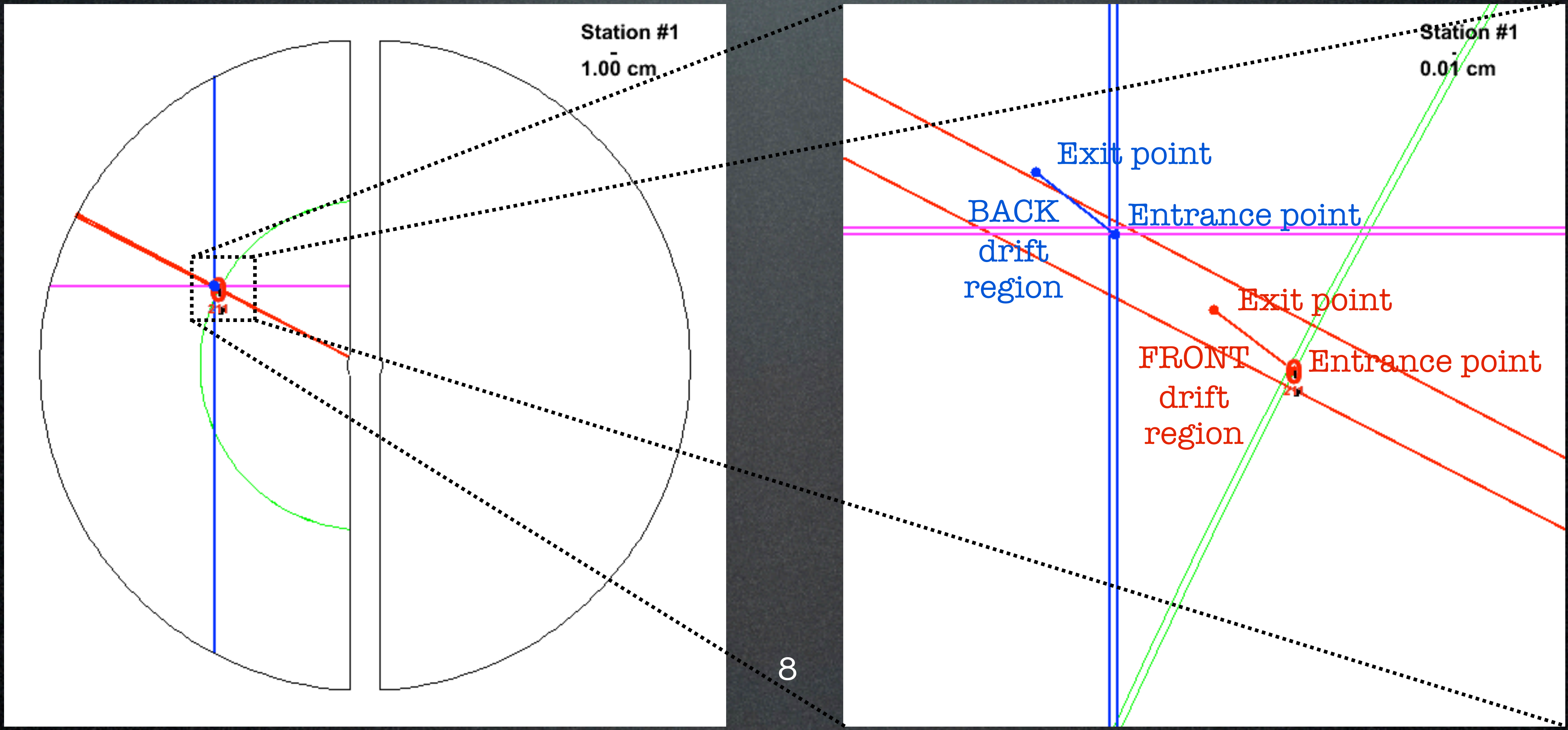
Digitization

PndGemDigitize

SetRealisticResponse(kFALSE)

- Ideal:
  - fire one strip per view (two strips for front and back drift regions)

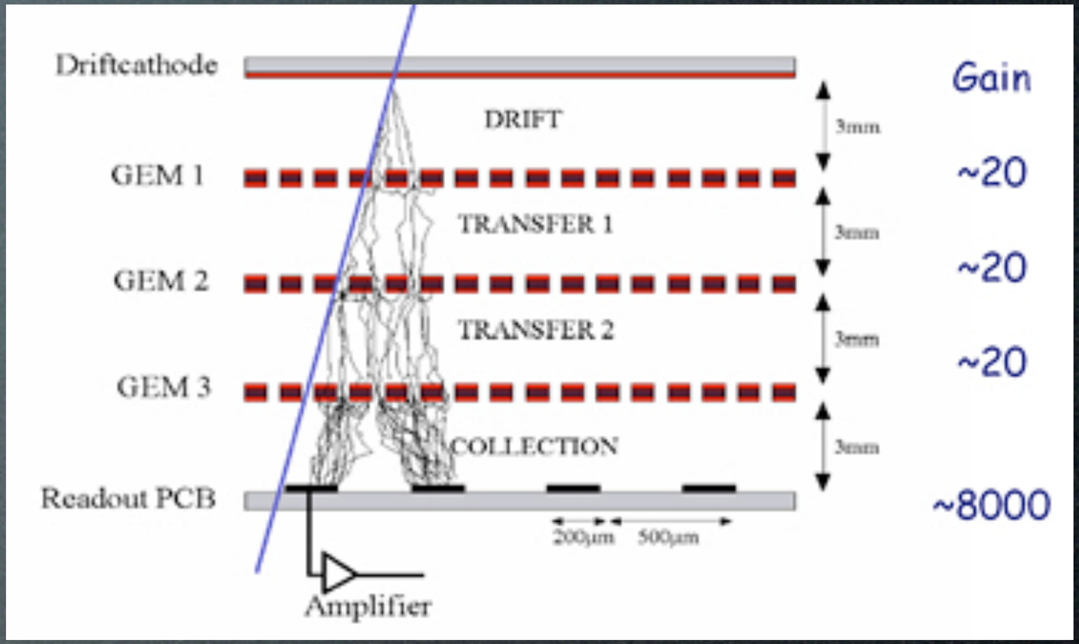




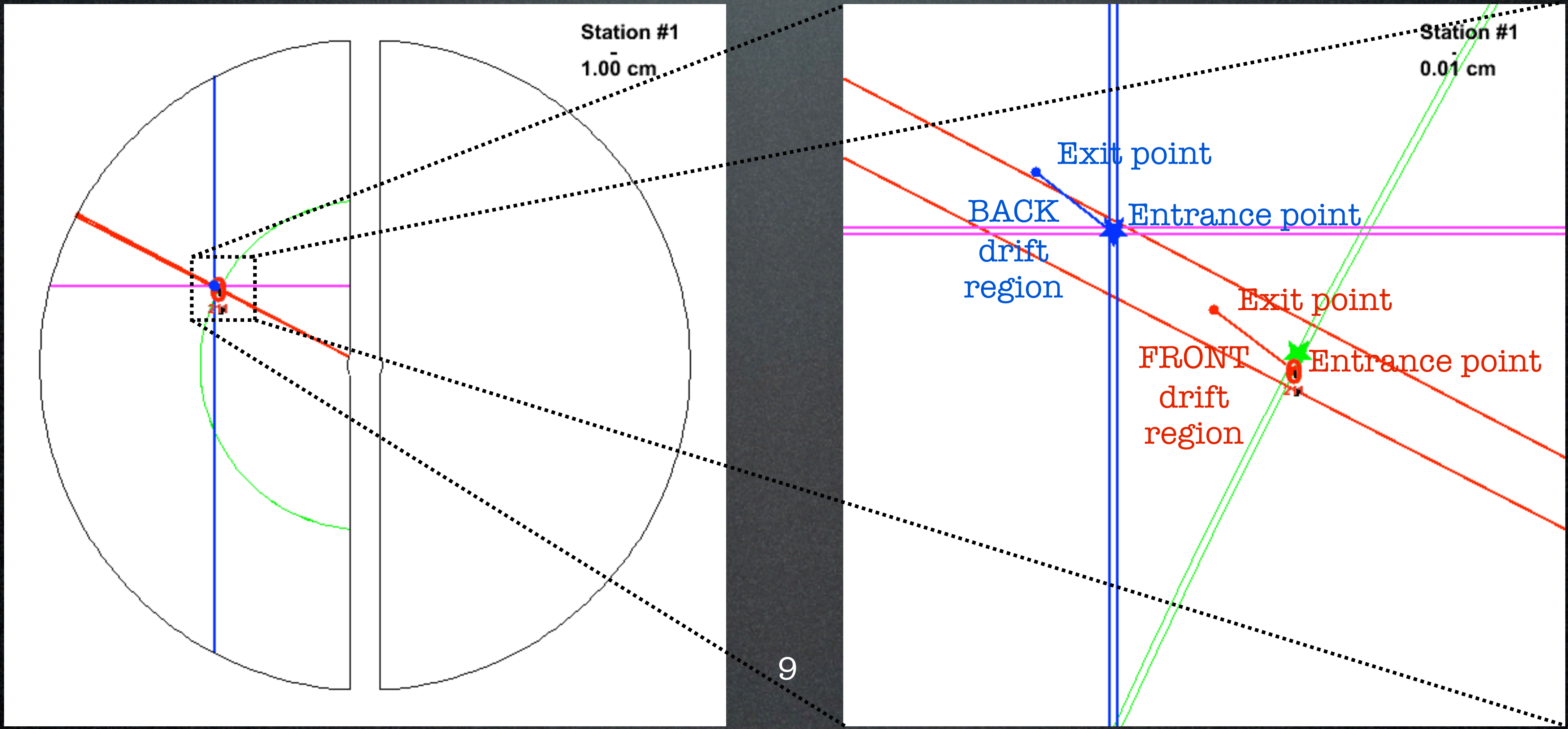
Hit finding

PndGemFindHits

SetUseClusters(kFALSE)



- Ideal:
  - create hits on front/back sensors separately where two strips cross

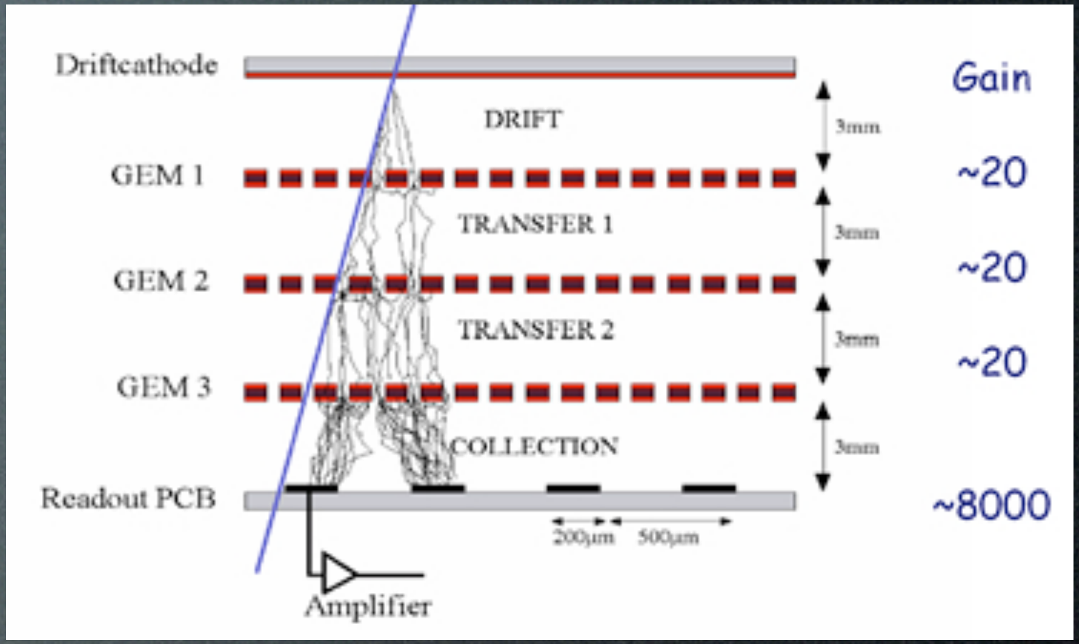


Digitization

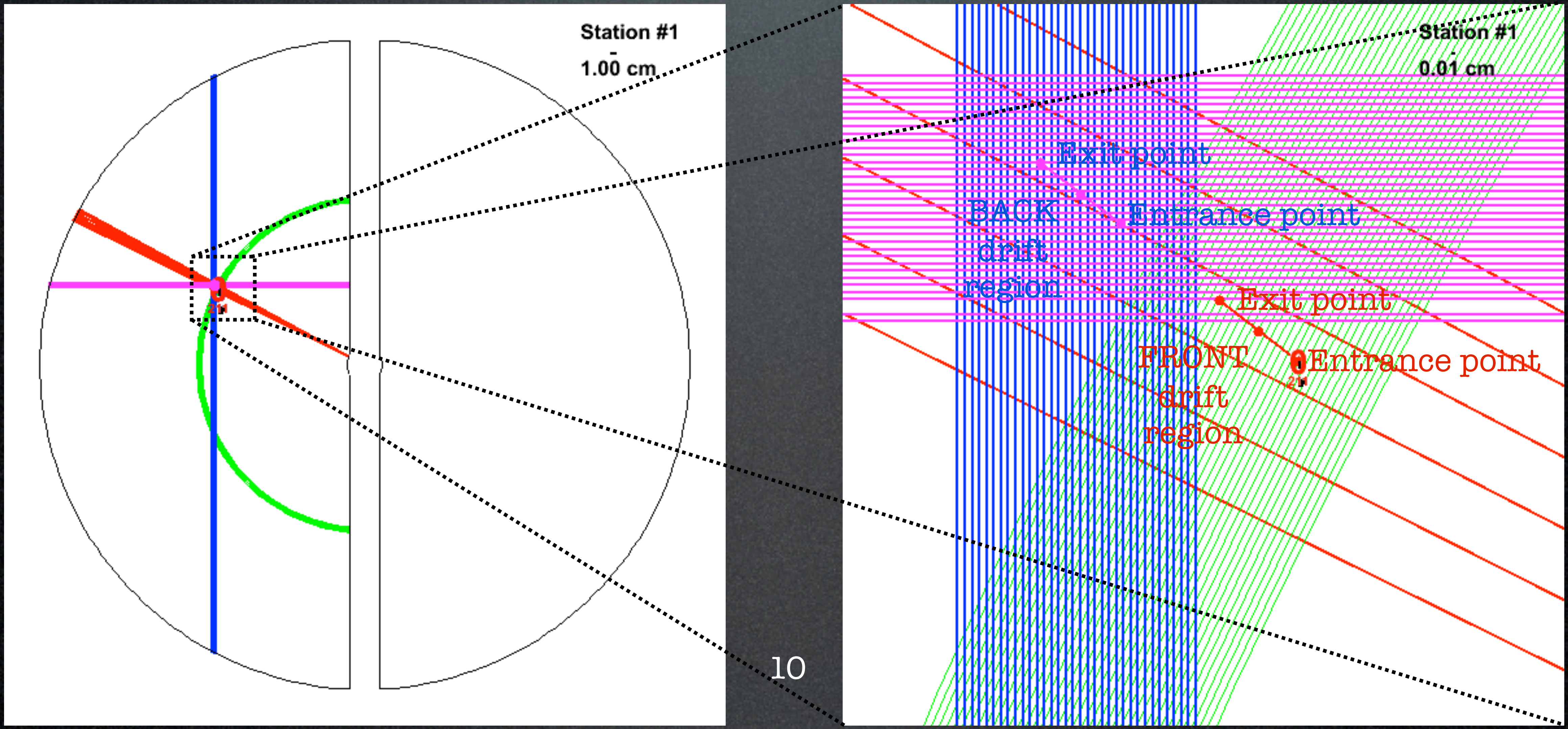
`PndGemDigitize`

`SetRealisticResponse(kTRUE)`

- Realistic:

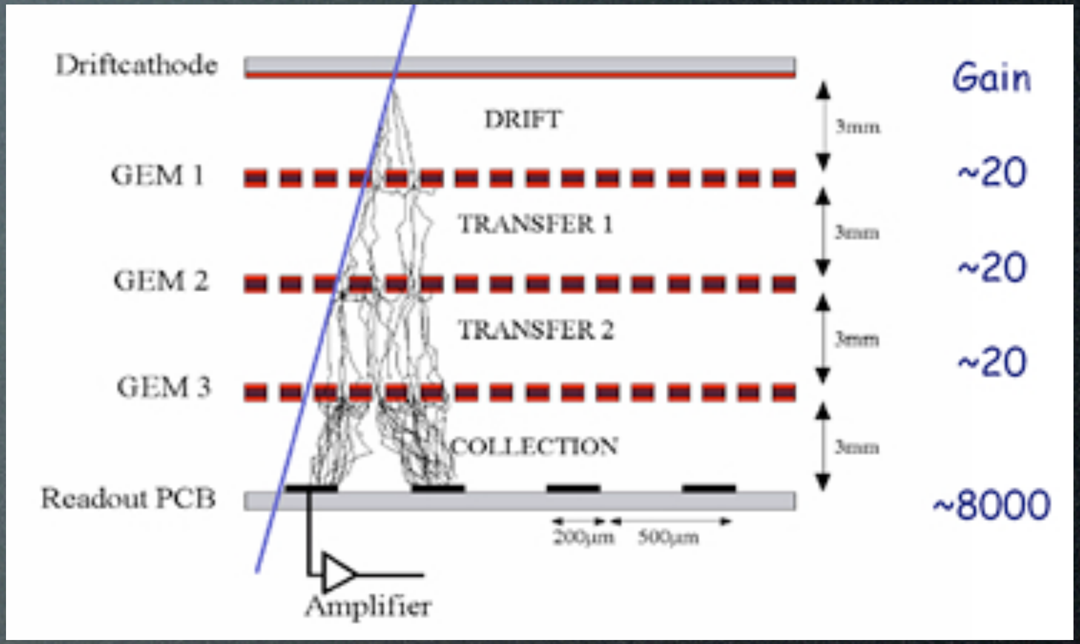


  - fire strips along the trajectory combined with charge diffusion

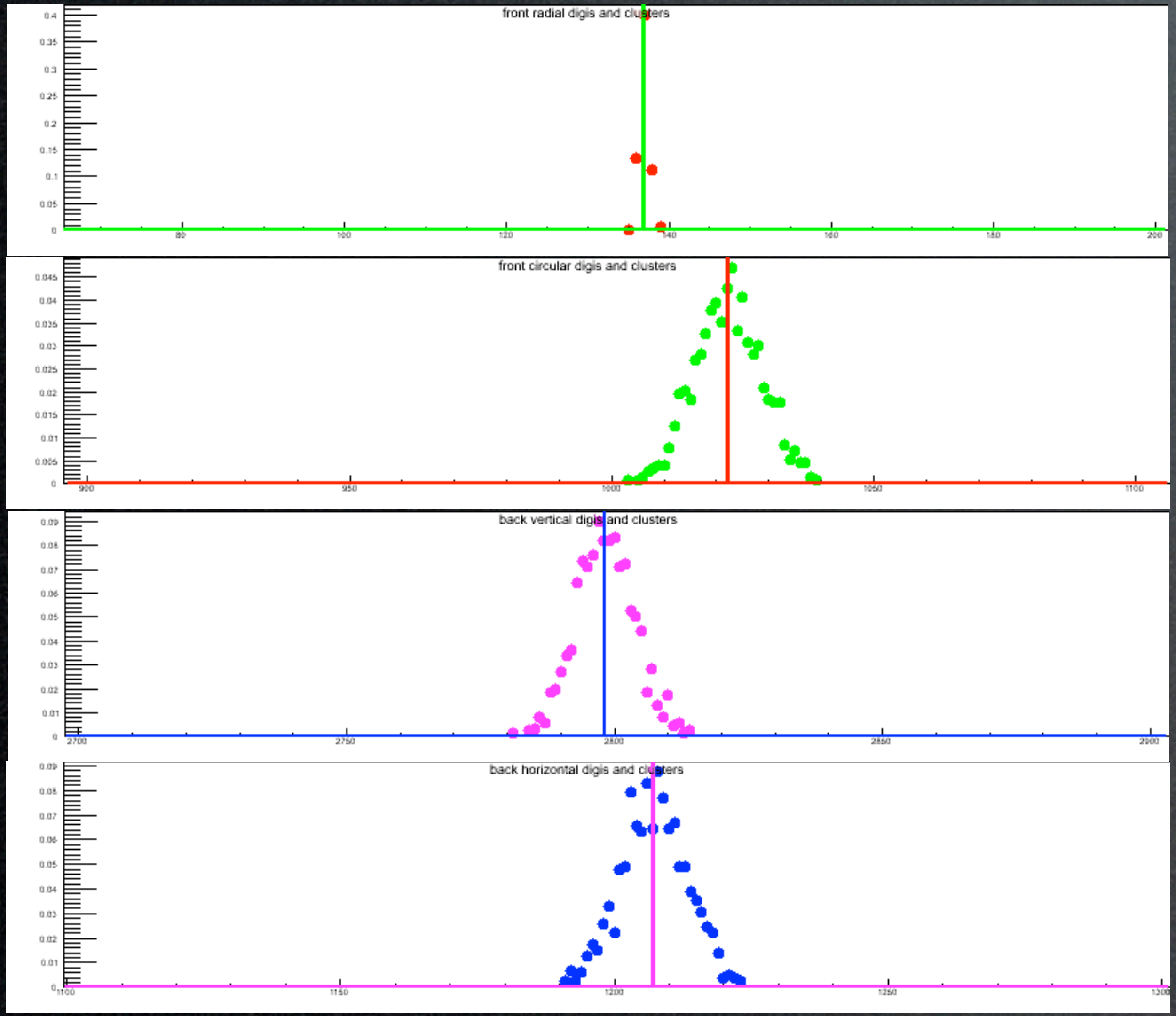


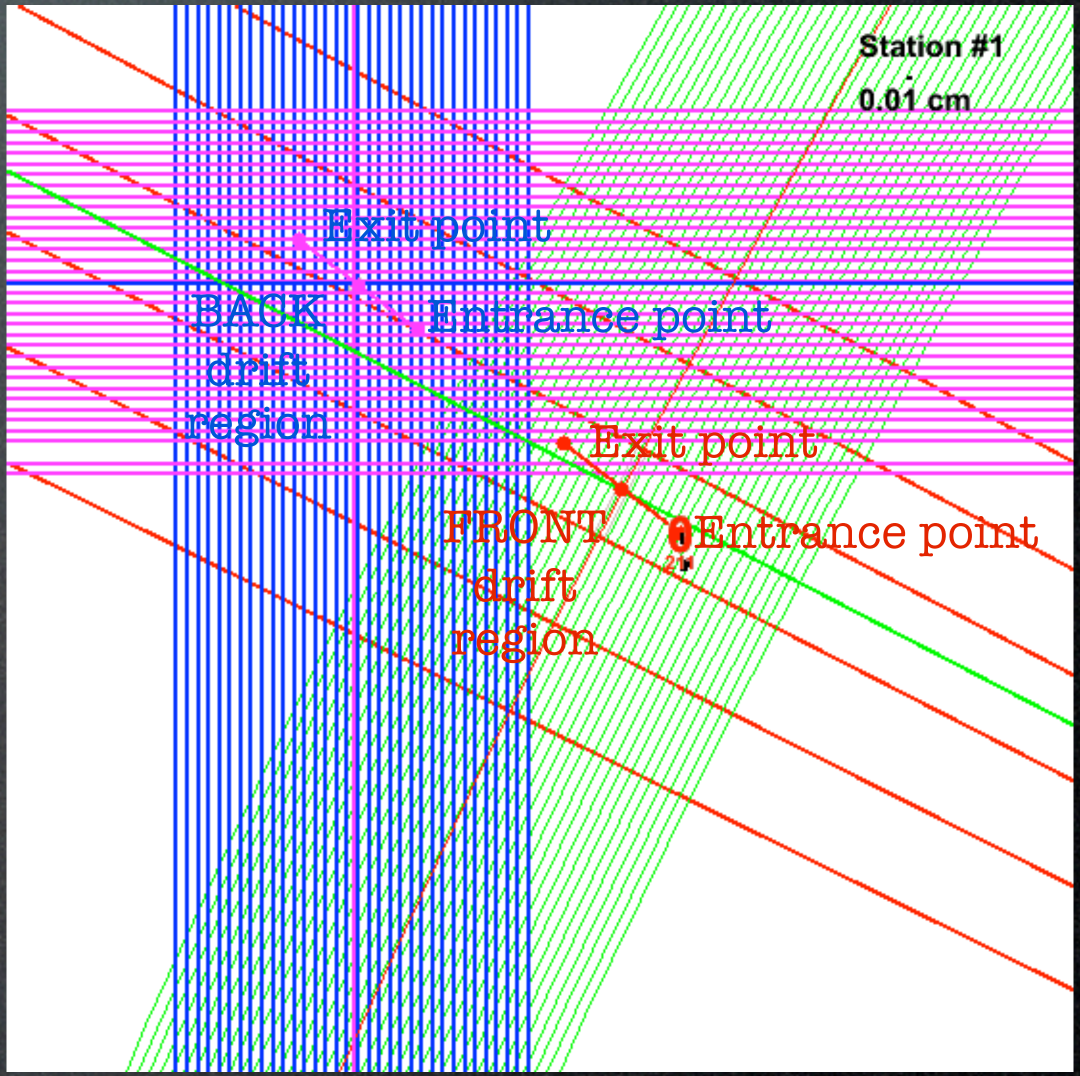
Cluster finder

PndGemFindClusters



- Realistic:
  - find maxima positions in charge strip distribution



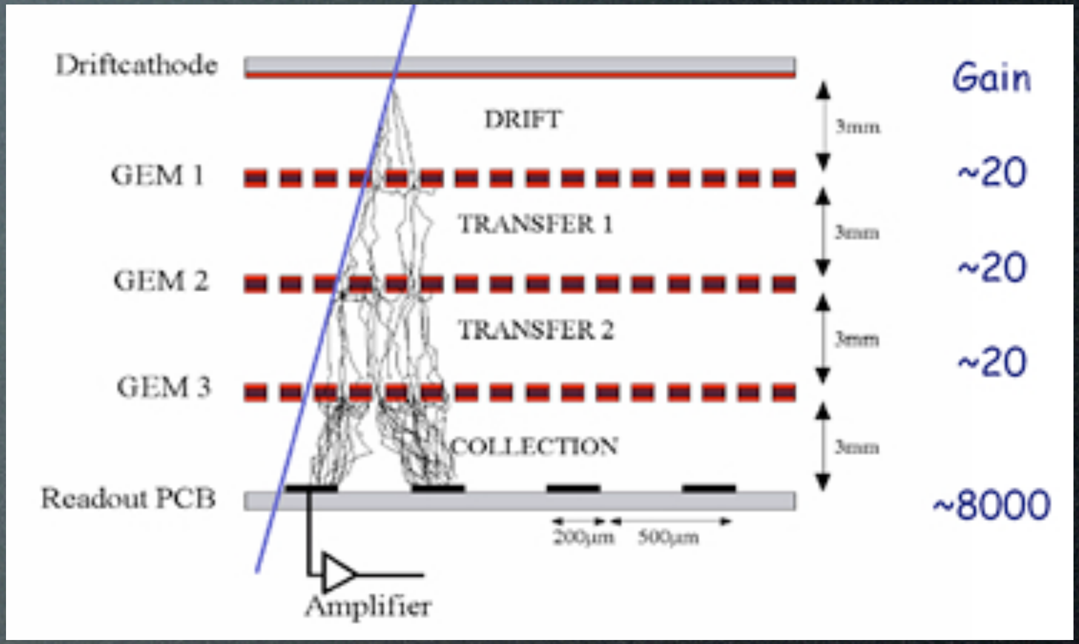


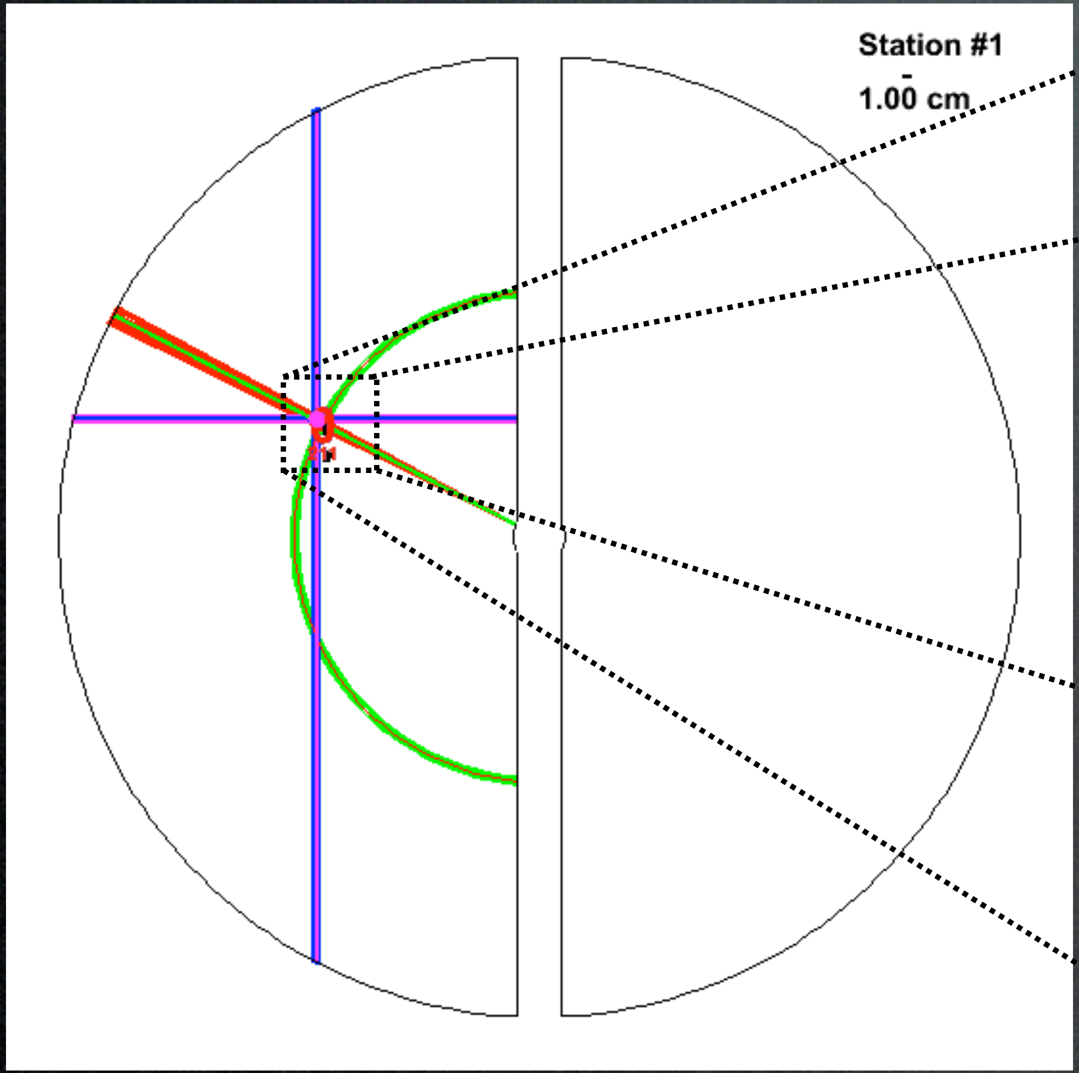
Hit finding

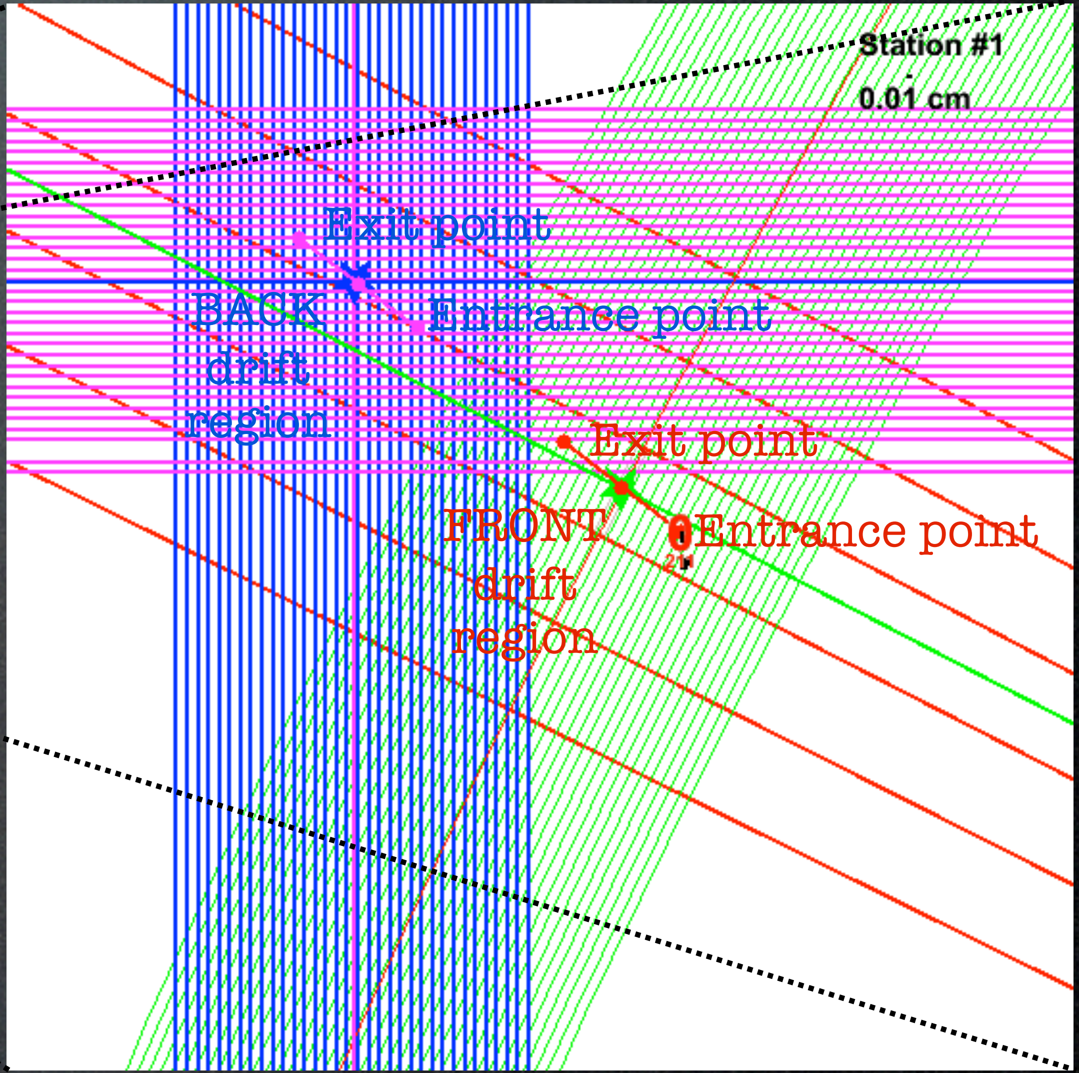
`PndGemDigitize`

`SetUseClusters(kTRUE)`

- Realistic:
  - create hits on front/back sensors separately where two clusters cross







Tracking

- GEM standalone track finding
  `PndGemTrackFinderOnHits`
- global track finding (MVD, STT, GEM)
  `PndBarrelTrackFinder`

GEM standalone tracking (2009)

- uses only GEM hits
- combines hits on each station's front/back sensors to reduce fake hits
- forms pairs of hits from different stations into simple tracklets, with momenta calculated assuming the tracklet emission from the vertex
- tracklets with similar momenta are joined to form tracks

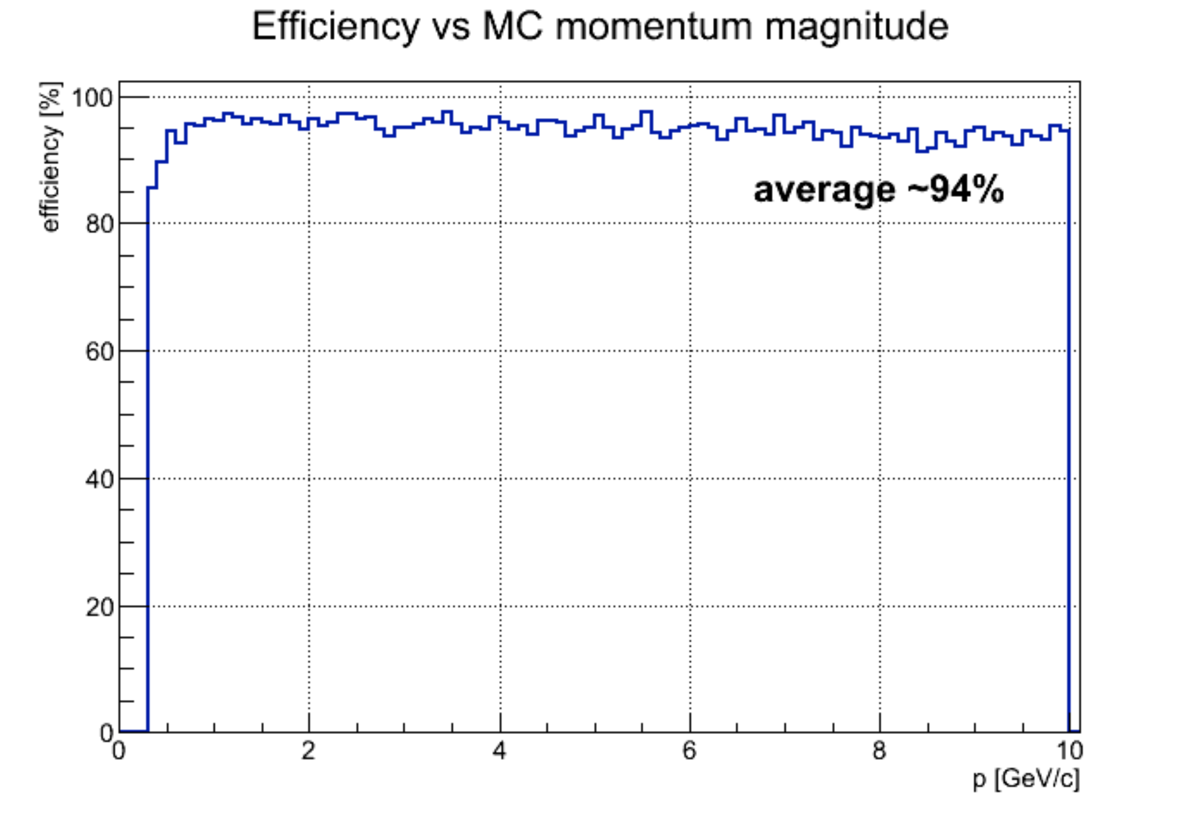
Global track finding (2010)

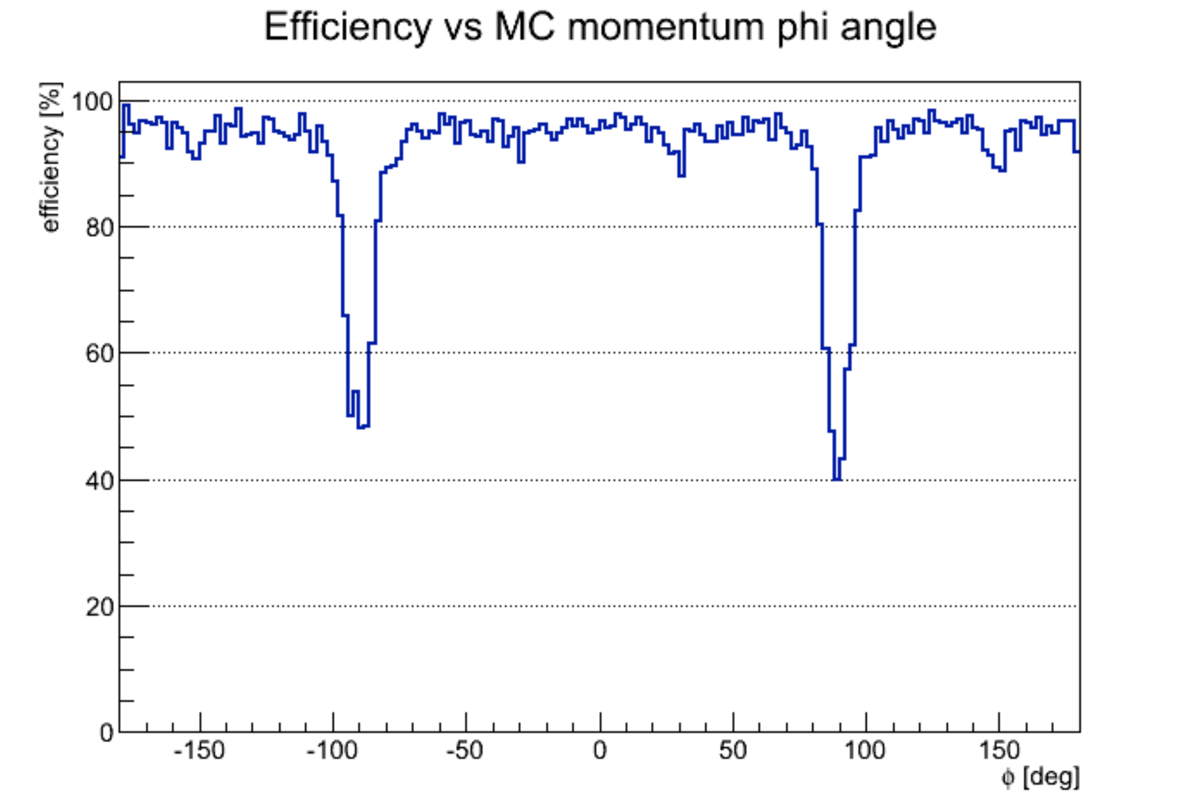
- uses MVD, STT, GEM hits

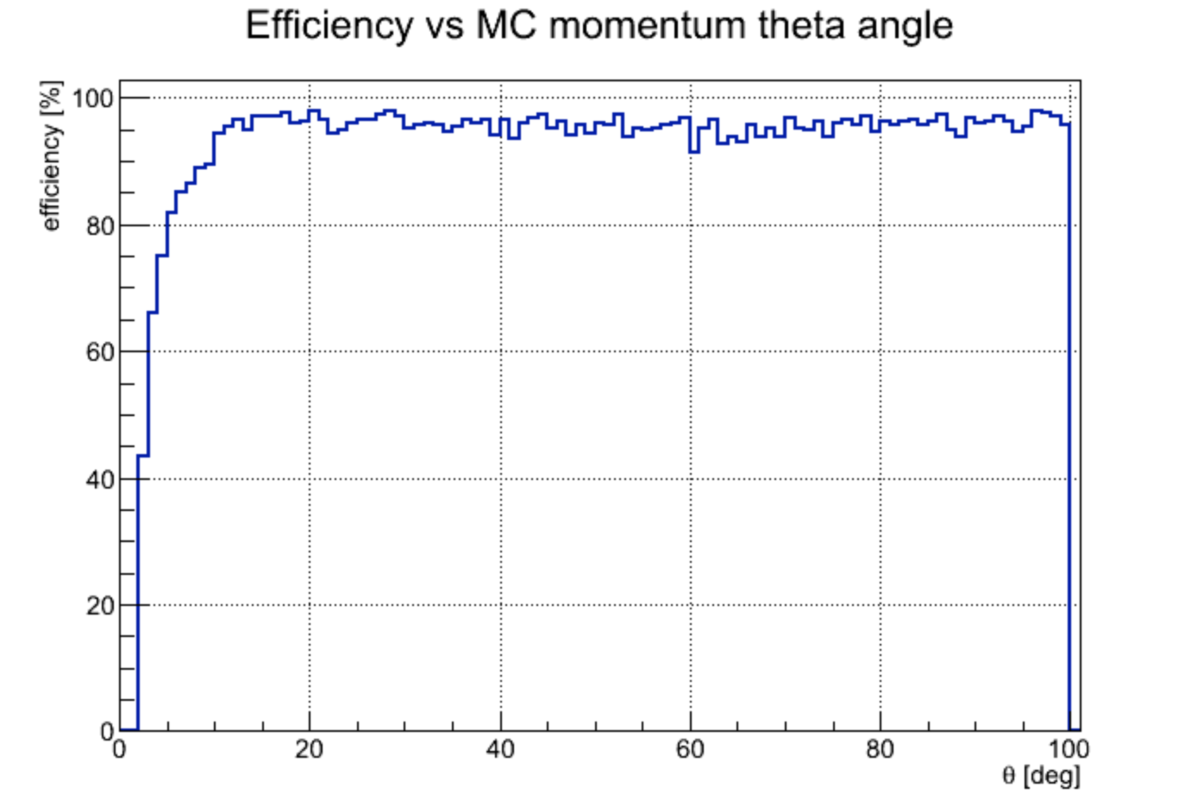
```
std::vector<Track> possibleTracks(0);
std::vector<Int_t> unusedHits(0);
for ( Int_t ihit = 0 ; ihit < nofAllHitsInEvent ; ihit++ ) {
  Int_t matching = 0;
  for ( Int_t itrack = 0 ; itrack < possibleTracks.size() ; itrack++ )
    matching += MatchHitToTrack(ihit,itrack);
  if ( matching > 0 ) continue;
  matching = 0;
  for ( Int_t iunh = 0 ; iunh < unusedHits.size() ; iunh++ )
    matching += CreateTrack(ihit,iunh);
  if ( matching > 0 ) continue;
  unusedHits.push_back(ihit);
}

CleanTracks();
```

Tracking efficiencies

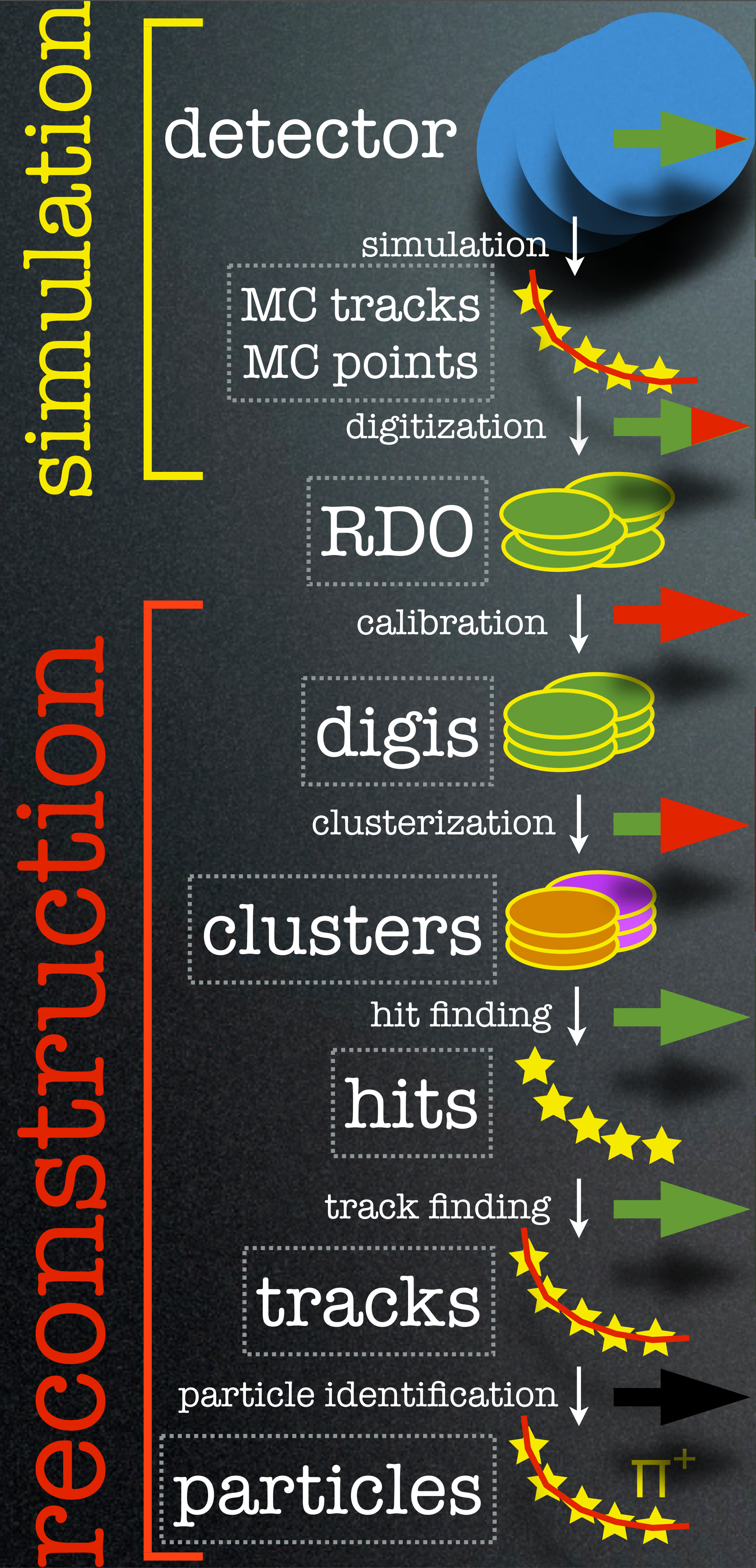
- 10000 events
- $2\mu^+$ and $2\mu^-$ / event
- $0.3<|p|<10$ GeV/c
- $0°<\theta<100°$
- $0°<\phi<360°$


Efficiency vs MC momentum magnitude


Efficiency vs MC momentum phi angle


Efficiency vs MC momentum theta angle

SUMMARY



simulation

detector

simulation

MC tracks
MC points

digitization

RDO

reconstruction

calibration

digis

clusterization

clusters

hit finding

hits

track finding

tracks

particle identification

particles

π^+

- realistic detector geometry
- update sizes, positions, implement readout

- realistic detector response
- no time information, decrease time spent on this phase

- NOT TOUCHED

- initial version implemented
- some work still necessary, no time information available/used

- digi/cluster intersection-not much to do

- lots of possibilities
- lots of effort put into this

- not applicable to GEM

some remarks on programming

- do READ and USE coding rules, coding conventions!!!
- you will soon discover they are NOT ENOUGH, and you will develop your own habits, of course in accordance to the common rules and conventions

my hints on programming

- NEVER EVER use single letters as variable names. Using s.l.a.v.n. does not save time. It **costs** time and money. My shortest ever variable name is `iev`, like in:

  ```
  for ( Int_t iev = 0 ; iev < nofEvents ; iev++)
  ```

  Moreover, my Int_t loop variables always start with ‘i’; class members always start with ‘f’, and never with ‘_’.

- CHOOSE meaningful variable names, function names, class names. BTW, the longest class name in PandaRoot is:

  `PndMvdTPCRiemannTrackFinderTaskCutPar.cxx`

  I have the 2nd place with:

  `PndGemMagneticFieldVsTrackParameters.cxx`, 3rd place:

  `PndEmcMultiWaveformToCalibratedDigi.cxx`.

my hints on programming

- write comments. I mean it.
- personally, I don’t. At least not the standard ones with // or /* */. Nevertheless, I was told that my code is easy to read. Because of lots of:

```
if ( fVerbose > 3 )
  cout << "this stt hit belongs to track " << trackNo
       << " (cause dist = "
       << FindCircDist(circPar,sH1) << ")" << endl;
```

- generally, when writing, debugging, checking code the programmer needs to produce lots of screen printouts. Write them decent. Later do not delete them, but put them in “if(fVerbose)” or in “//”. At least this.

GEM in myMacro.C

- Simulation

```
FairDetector *Gem = new PndGemDetector("GEM", kTRUE);
Gem->SetGeometryFileName("gem_Gas_3Stations.root");
fRun->AddModule(Gem);
```

- Digitization

```
FairParRootFileIo* parInput1 = new FairParRootFileIo();
parInput1->open(“gem_Gas_3Stations.digi.par”);
PndGemDigitize* gemDigitize = new PndGemDigitize("GEM Digitizer", verboseLevel);
gemDigitize->SetRealisticResponse();
fRun->AddTask(gemDigitize);
```

- Cluster Finding

```
PndGemFindClusters* gemFindClusters = new PndGemFindClusters("GEM Find Clusters");
fRun->AddTask(gemFindClusters);
```

- Hit Finding

```
PndGemFindHits* gemFindHits = new PndGemFindHits("GEM Hit Finder", verboseLevel);
gemFindHits->SetUseClusters();
fRun->AddTask(gemFindHits);
```

- Standalone GEM Tracking

```
PndGemFindTracks* finderTask = new PndGemFindTracks("PndGemFindTracks");
fRun->AddTask(finderTask);
PndGemTrackFinderOnHits* mcTrackFinder = new  PndGemTrackFinderOnHits();
mcTrackFinder->SetPrimary(0);
finderTask->UseFinder(mcTrackFinder);
```

- Global Tracking

```
PndBarrelTrackFinder* barrelTF = new PndBarrelTrackFinder();
fRun->AddTask(barrelTF);
```