Pixel telescope analysis with Marlin & C.

Antonio Bulgheroni - INFN

Contents

- Main goal of the analysis / reconstruction software
- Logical diagram of a “typical” analysis procedure
- A list of what has been already done and what is missing
- Event model
- Histogramming and DQM
- On-line monitoring
- Conclusion

More than a talk, this should be considered as a working session.

What I’m proposing is just my opinion and of course our review is needed.

Main goal of the analysis / reco software

- Provide to the DUT users a set of high level objects (essentially tracks and data quality histograms), they can use to perform their measurements.
- Provide access to raw data, calibration constants and conditions in a well documented data format (LCIO) in order they can perform their reconstruction algorithm.
- Our framework should be flexible enough to embed the DUT and allow the users to perform their analysis within the framework or to use thier software with the provided high level objects.
  - Along with the LCIO output, maybe it is worth to provide also ROOT TTree as output file.

Analysis tasks

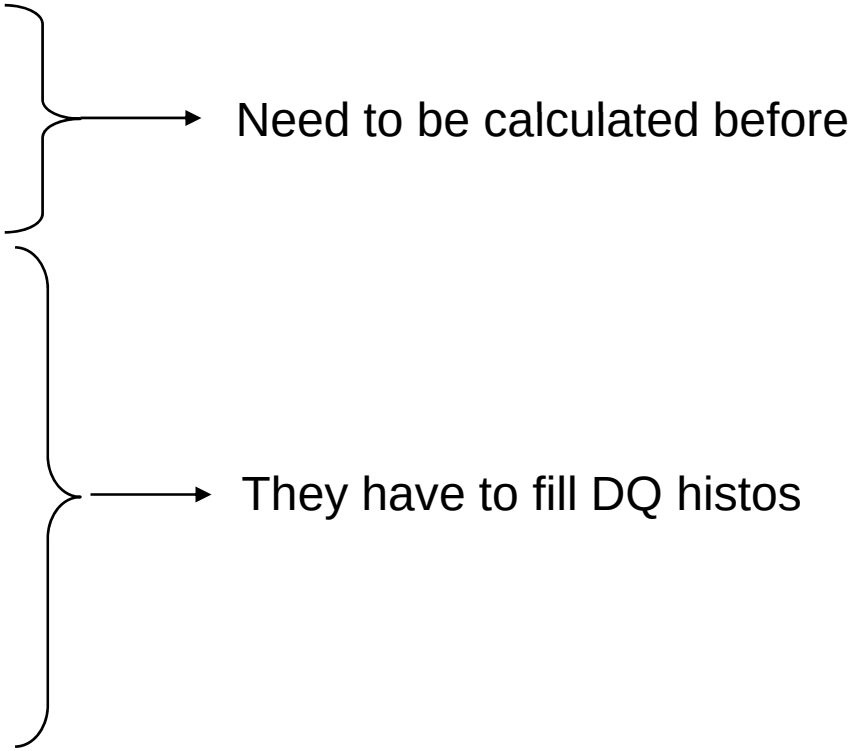
To go from “DAQ raw data to tracks” we need a set of “calibration constants”:

- Pedestal + Noise + Bad pixel masks
- Eta functions (for each detector planes)
- Alignment constants (for each detector planes)

For each of those, we need a specific analysis task to produce them...

Typical analysis procedure

```
getPedestalNoiseStatus();
getEtaFunctions();
getAlignment();
while ( iEvent < nEvent ) {
    removePedestalAndCM();
    lookForClusters();
    applyEtaCorrection();
    fromPixelToSpacePoints();
    fitSpacePoints();
    updatePedestalNoise();
    saveTrackToFile();
}
```



(Pedestal + Noise + Mask) processor

- Requires one or more loops over all or a fraction of events. (class RewindDataFilesException)
- Pedestal and noise values are calculated according to few algorithms already implemented, other can be added!
- In the case of Self Biased detector we can reduce it to a very simple processors assigning to all pixels the same initial value of pedestal (= 0 ADC) and a resonable value of noise (~ 1 ADC). The pedestal/noise update processor can be used to correct them. What about bad pixel masking?
- Common mode suppression algorithm applied a certain number of times in event loops following the first one. Only one algorithm implemented.
- The output data are saved into a LCIO condition file. Consider to save it into the condition db.

class EUTelPedestalNoiseProcessor
class EUTelAutoPedestalNoiseProcessor

Tested with real and simulated data

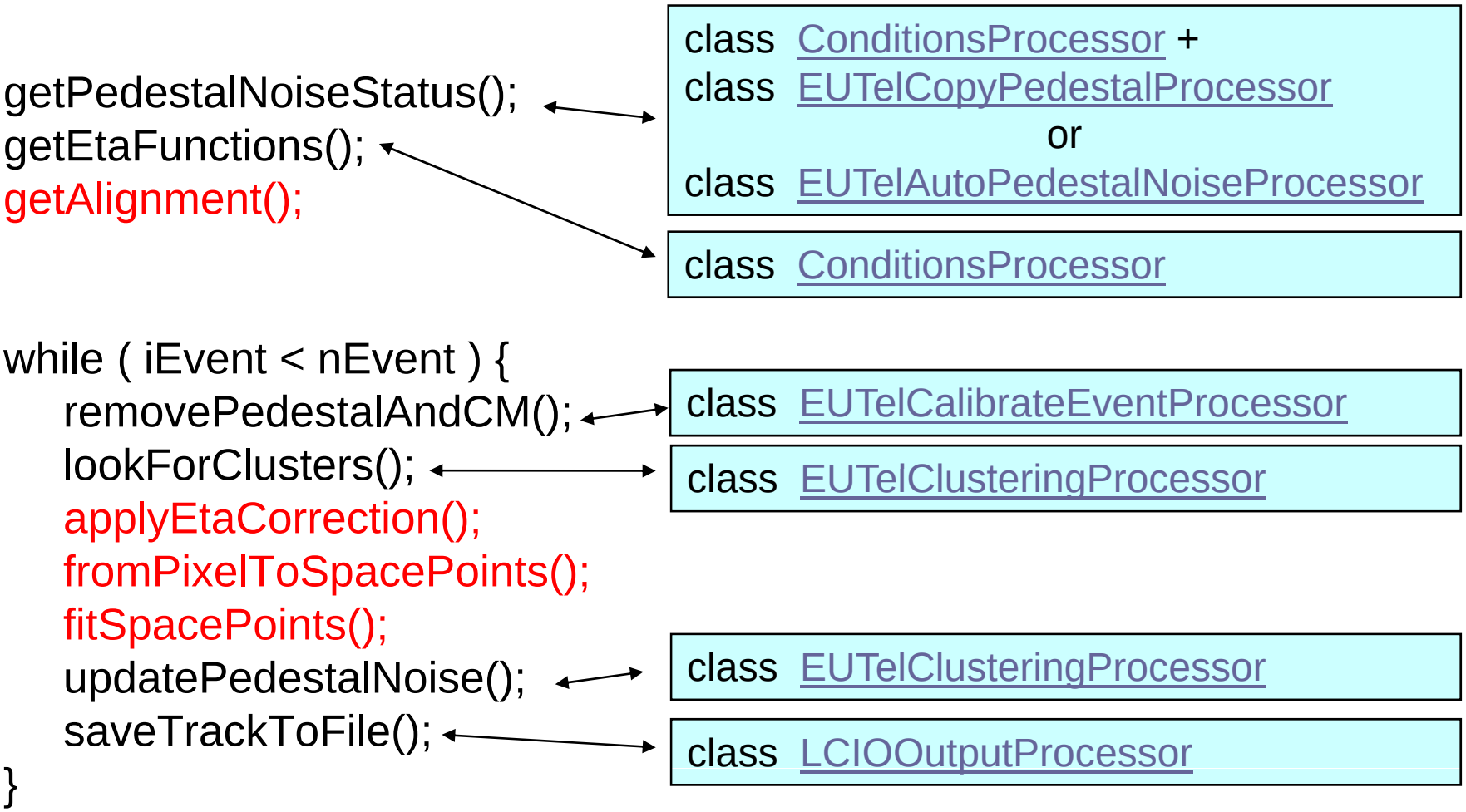
Calculate Eta processor

- It needs to have already pedestal and noise values.
- It requires (probably) one loop only on events.
- Input data are the identified clusters.
- The output are two “functions” for each detector plane. Saved into another LCIO condition file (Again consider the possibility to save them into the db)
- Different calculation algorithms are already implemented, but experts are welcome to contribute.

class EUTelCalculateEtaProcessor

To be tested with real data!!!
Expert opinions is absolutely needed!

Typical analysis procedure (2)



Event model

- Header: EUTelRunHeaderImpl is a reimplementation (decorator pattern) of LCRunHeaderImpl. Probably not needed if we want to use BORE and EORE (should we reimplement LCEventImpl ?)
- Data:
  - TrackerRawDataImpl for raw data and pixel status
  - TrackerDataImpl, noise, calibrated data and NxM clusters (class EUTelFFClusterImpl). I wasn't able to find anything better for clusters.

MATRIXDEFAULTENCODING = "sensorID:5,xMin:12,xMax:12,yMin:12,yMax:12"

CLUSTERDEFAULTENCODING = "sensorID:5,clusterID:8,xSeed:12,ySeed:12,xCluSize:5,yCluSize:5,quality:5"

Histogramming and DQM

- Histogramming: Marlin is offering a class AIDAProcessor
  - The only C++ AIDA implementation I've found is RAIDA (ROOT implementation of AIDA) that is particularly full of bug due to the early developement state. Do you know anything else?
  - AIDAJNI enables to link against any JAVA AIDA implementation such as JAS from a C++ code. But AIDAJNI is not compatible with the latest AIDA.
  - Why not making a class ROOTProcessor only for histogramming?
  - We need to define a bunch of reference histogram each processor has to fill in

On line monitoring

- It was told that the same framework can be used for on-line monitoring.
- This requires the integration with an interactive graphical user interface.
  - How? Which GUI?
- Is it a DAQ monitor or a Marlin processor?

Conclusion

- More than a conclusion this is a new starting point.
- After a careful observation, there are no show stoppers in using Marlin. There are still some issues (as the DataRewind or the histogramming) that have to been better implemented by the framework developers.
- A great part (say 75%) of the sucimaPix code has been moved to Marlin Processor.
- There is still a lot to do and time is ticking and July is approaching!
- Paris software meeting in May: should we go?