Subject: Re: PID package
Posted by StefanoSpataro on Fri, 07 Aug 2009 14:04:29 GMT
View Forum Message <> Reply to Message

Hi Bertram,

Bertram Kopf wrote O.k. this means that the correlator should be only responsible for the collection of the properties of all subdetectors which might be useful for PID. When I look to your proposed data flow I come to the conclusion that it is only foreseen to collect all PID information, to apply a network etc. and based on this result to create a RhoCandidate. This means that just a part of the global PID will be supported. As you know most experiments make use of likelihood based algorithms where e.g. each detector provides PID likelihoods which will be combined afterwards. Does it mean that subdetector PID will be skipped completely? Abstract classes, mechanisms for the possibility to choose different algorithms at runtime and common tools to handle probabilities etc. are - I think - also very helpful to provide.

I don't find any of such classes and tools in the present code. Is there an idea and a concept to implement such things?

Youa re completely correct, and I try to explain the scheme the have thought. My PID slides could help.

The correlator is just doing correlation between tracks and detectors, creating the PndPidCandidate (neutral and charged). This object is storing all the indexes and almost all the useful informations whicl will be useful for PID, but it is not doing PID at all.
Once the "Candidate" is created, or better once all the candidates are created, then you can run the algorythms (still not yet implemented). The algorythms could be single detector PID (such as MVD dE/dx), or more complicated stuff such as multi variate analysis, neural network, and so on.
Each algorythm loops over the PidCandidates, and create a new TCA made of PndPidProbability objects, and with the name related to the algorythm. PndPidProbability is just a collection of probability of each candidate of being an electron/pion/kaon/proton/gamma/pi0/etc etc etc.
This means that you store one TCA for the candidate, many TCA for the probability. Each candidate is linked to the many algorythm TCAs, i.e. position 0 of the candidate coresponds to position 0 of the MVD dE/dx Probability and position 0 of the EMC shower propertied neural network, position 1 corresponds to .. etc etc..

Each candidates is linked to many kind of probability coming from different algorythms.

Once the user want to run some selection, he should use some TPidSelector, that can select paricles according even to combination of different algorythm.
I.e., please select me electrons with probability > 0.1 in EMC shower neural network algorythm, and pions with prob >0.1 combining DIRC algorythm and tpc dE/dx. Just an example.
Of course, once the code wil be fixed, we will provide "standard" pid selectors, so that everybody will use the same stuff without increasing the amount of "entropy" in the analysis.

In this way I think and hope we can have all the features and the flexibility you are correcting asking for (but better ideas are always welcome)

Quote:
The track matching with the detectors STT, EMC, Tof and MVD are definitely done in this PndPidCorrelator. Such a track matching is in general detector dependent and not that easy and should be done separately. Does it mean that this is right now a workaround since essential parts in the reconstruction are missing? If so I am not sure if such a workaround - to put everything in one class/method - is the best solution. In my point of view life is easier to start with a proper design/structure and if something is still not there to provide dummy classes. With your workaround you have to change and reorder a lot of parts again and again.

I have thought of separating correlation and "information" extractions. You are correct, but considering that at the moment I am the only one working on this code and that alone there are many things to do, I prefered to leave the PndLhePidMaker structure, in order having (almost) everything running and ready for the next levels (pid algos and selection). I plan to de-encapsulate single parts once they will be ready. I think I will start with the muon class, which I would like to write.

Quote:
Then please remove the relevant lines immediately. Especially newcomers should not see such nonsense.

I will do it, however users should not play with task but only with data objects. This si the reason why I am slow in deleting this things.

Quote:
iii. in line 436 a stack overflow has not been caught.
(cut)
sorry, I meant an overflow of the array. This must be caught!

In theory endnum is 60% of sttCounts, which corresponds to the sixe of sttdedx. Then in theory it should be always lower than the array size. But I will take a look if maybe some exceptions could be nasty.

Quote:
Does it mean that this class provides just methods where one can ask for the index where to find the pointer in the branch? Does it mean that I have to take care of to build the transient object? This is very error prone because I have to know the correct tree and branch and in addition it is not type save. Isn't there a common mechanism available which is encapsulated from reconstruction?

I will start from the end: no, the common mechanism is not yet written. It could be useful, but at the moment I have not a clear idea on how it is possible to do this. Comments from the core experts are welcone. At the moment the "authors" should create the corresponding object activating the TClonesArray from the branch. The TCA in general are standard (MvdHit, SttHit, GemHit, TofHit, DrcHit, TpcCluster, EmcCluster, etc..), but for newcomers maybe this could raise some problems.
This point is still open.

P.S. I have already answered to this message but then the browser decided to close... this is

the second time, I am not reading again, I hope it is clear enough