

LArSoft cvs (legacy site) - Idea #344

Redefining the Hit Class

03/04/2010 11:54 AM - Brian Page

Status:	Closed	Start date:	03/04/2010
Priority:	Normal	Due date:	
Assignee:	Brian Page	% Done:	0%
Category:		Estimated time:	0.00 hour
Target version:			
Description			
<p>As we have discussed, the hit class is in need of some changes. This is my suggestion for the Hit class definition:</p> <p>We change the constructor to take a vector of pointers to wires. This way we can define 2D hits as having 1 entry and 3D hits as having > 1 entry. Eliminate the fView datamember and make the View() function return k3D if the size of the wire vector is greater than 1 and if it is 1, return the view from the wire.</p> <p>I would eliminate the UpTime, DownTime, UpADC, DownADC, and CrossingTime methods and data members as these are only relevant to the specifics of the raw data shape, which is really not needed downstream of hit finding. And if it is it can be found from the raw data in the wire object.</p> <p>I would add an fCenterTime data member which will be the central position of the hit, as defined by the hit finding algorithm. Whether this is the peak or the crossing or some other position is left up to the algorithm writer.</p> <p>Adding a fWidth , and fChiSquare parameters would be useful for many things. In the case of a unipolar shape the width could be the width/2 at 1/2 maximum, or perhaps the (maxTime-minTime)/2 if working with a bipolar shape. There is currently no error parameters in our hits, and the ChiSquare parameter would quantify how well the hit fit is, if one is done. The width and fit ChiSquare may be very useful if trying to disentangle whether a hit is a superposition of 2 very close hits, or if a hit is part of a very steep track, which can probably only be done with the clustering/tracking angular information. Doing this would probably mean the fMultiHit flag is no longer useful and should be removed.</p> <p>We need two signal parameters, the integrated area(fIntSignal) and the signal amplitude(fAmplitude). We can then create a method which returns the corrected ionization using the appropriate method for each detector. If we do this, then fMIPS is probably redundant and can be removed, or just make MIPS() the function for the corrected ionization.</p> <p>Also there is no reason to store the signal vector in the hit, we store the beginning and end time as well as the wire, so we can create a Signal() function using the c++ assign() function to return this vector without double-storing these values.</p> <p>Please let me know what you all think of this scheme</p>			

History

#1 - 03/04/2010 02:55 PM - Brian Page

- File Hit.h added

- File Hit.cxx added

I've gone ahead and implemented this idea just to test it out, everything is incorporated except that the MIPS() function currently just returns fAmplitude, awaiting a proper method for calculating ionization. I've tested it out with evd and my hit finder and so far it seems to work.

#2 - 03/09/2010 10:48 AM - Brian Rebel

- Category set to Reconstruction

This is a pretty good proposal, but I have a few comments:

We change the constructor to take a vector of pointers to wires. This way we can define 2D hits as having 1 entry and 3D hits as having > 1 entry.

I am not thrilled with the idea of having a hit point at more than one wire. It seems that we open ourselves up for a lot of confusion if we allow m --> n dependencies like that.

Eliminate the `fView` data member and make the `View()` function return `k3D` if the size of the wire vector is greater than 1 and if it is 1, return the view from the wire.

I like the idea of having a `View()` method. But as I said, having more than 1 wire map to a hit is not ideal.

I would eliminate the `UpTime`, `DownTime`, `UpADC`, `DownADC`, and `CrossingTime` methods and data members as these are only relevant to the specifics of the raw data shape, which is really not needed downstream of hit finding. And if it is it can be found from the raw data in the wire object.

Sounds good.

I would add an `fCenterTime` data member which will be the central position of the hit, as defined by the hit finding algorithm. Whether this is the peak or the crossing or some other position is left up to the algorithm writer.

Why not just call it `fTime`? Then each algorithm defines what it means by `fTime`.

Adding a `fWidth`, and `fChiSquare` parameters would be useful for many things. In the case of a unipolar shape the width could be the width/2 at 1/2 maximum, or perhaps the $(\text{maxTime} - \text{minTime})/2$ if working with a bipolar shape. There is currently no error parameters in our hits, and the `ChiSquare` parameter would quantify how well the hit fit is, if one is done. The width and fit `ChiSquare` may be very useful if trying to disentangle whether a hit is a superposition of 2 very close hits, or if a hit is part of a very steep track, which can probably only be done with the clustering/tracking angular information. Doing this would probably mean the `fMultiHit` flag is no longer useful and should be removed.

`fWidth` seems like a fine variable. I would have a variable named `fFOM` (Figure Of Merit) rather than `fChiSquare` as I can imagine that some algorithm writer would rather use a different test than a χ^2 .

I agree that `fMultiHit` can be removed. Either the hit finder has determined that there are two hits there and made 2 hit objects somehow out of the overlapping signal, or it can't resolve the two hits.

We need two signal parameters, the integrated area (`fIntSignal`) and the signal amplitude (`fAmplitude`). We can then create a method which returns the corrected ionization using the appropriate method for each detector. If we do this, then `fMIPS` is probably redundant and can be removed, or just make `MIPs()` the function for the corrected ionization.

Sounds fine.

Also there is no reason to store the signal vector in the hit, we store the beginning and end time as well as the wire, so we can create a `Signal()` function using the `c++ assign()` function to return this vector without double-storing these values.

Also fine.

#3 - 03/09/2010 10:49 AM - Brian Rebel

- Assignee set to Brian Page

#4 - 03/09/2010 11:06 PM - Brian Page

In place of having a vector of Wires, we could make the `Wire` parameter optional, as 3D hits are constructed from information from both planes, it makes no sense to have 1 wire reference, so it might as well be 0. Then we would have a similar identifier to the one I described before for 3D hits. However there will be no backtracking, but backtracking would be impossible without letting the 3D hits have references to 2D hits as well as 2D tracks, which would likely be used to create them, so this couldn't be done in the current scheme anyway.

I'm agree with all of the other changes you suggestions as well.

#5 - 03/10/2010 10:07 AM - Brian Rebel

Perhaps what we need is a separate 3D hit class that can point back to a set of 2D hits. I hate to loose the ability to trace a reconstruction backwards to raw signals.

#6 - 03/22/2011 05:16 PM - Brian Rebel

- Status changed from New to Resolved

The Hit class has evolved beyond this discussion.

#7 - 03/22/2011 05:16 PM - Brian Rebel

- Status changed from Resolved to Closed

#8 - 10/08/2013 06:46 PM - Erica Snider

- Project changed from LArSoft to LArSoft cvs (legacy site)

- Category deleted (Reconstruction)

Files

Hit.h	3.6 KB	03/04/2010	Brian Page
-------	--------	------------	------------

