Subject: Re: Position calculations on start/stop scintillators
Posted by miree on Fri, 15 May 2015 08:48:16 GMT
View Forum Message <> Reply to Message

Hi Tayfun,

The existing code finds the position based on a grid search on (the 2-dimensional) surface of the Detector. It assumes that the detector was hit at point x,y and calculates how big the variance in the position-corrected time signal is. It moves the point x,y around until it finds a minimum in this variance.

However, the procedure assumes that one can calculate the time delay (caused by light propagation from particle to PMT) correctly based on a known relation (this is used in the "analyze" function that is called in the code). During calibration one has to find for each PMT the exact (linear) relation between particle-PMT-distance and time delay caused by this distance. You might ask: "Isn't the time delay exactly the distance divided by the speed of light in the scintillator material?". Initially I thought the same, but empirically this is not the case. So far, I didn't understand why. Especially why this value is different for each PMT. But so far the empirical lesson that I learned is: reliable position determination only works after a careful calibration of each PMT based on measured data. This calibration is done by plotting the time delay versus the distance. But this requires knowledge of the position.

As far as I understood, you are in the situation that you do not have any reliable position information (faulty TPCs) at the location of the ToF detectors. Consequently, you cannot create the plot that is required to determine the calibration coefficients. You can do the plot using the time difference and the position coming from the (uncalibrated) ToF detector. But this will not improve your calibration coefficients.

What you have to do (formally) is the following: Try all possible combinations of the 64 calibration coefficients (32 slopes + 32 offsets). For all of these you have to make a qualitative statement of how good they are. Eventually you will find a set of coefficients that is best (gives the highest quality value) and you take the corresponding calibration coefficients. This is what I tried to describe as an algorithm before. Basically this is searching a minimum/maximum in a 64-dimensional space. Each of theses search steps requires to make the 2-dimensional search for the minimum variance that is done in the code, so ... yes, the existing code could be part of this algorithm. Maybe there are some tricks that can be applied that I'm not aware of. Maybe someone has a good idea to achieve the same with simpler means.

Best regards,
Michael