Subject: Implementing Software Alignment into PandaROOT
Posted by Roman Klasen on Tue, 24 Oct 2017 10:04:20 GMT
View Forum Message <> Reply to Message

Hello,

I'm plan to integrate a software alignment framework into PandaRoot. For now, that means that our alignment parameters are stored to transformation matrices that are multiplied to the ideal transformation matrices to simulate a realistic, non-ideally aligned geometry.

As far as I can see, our simulation chain currently works like this:

0. MC tracks are generated by the BoxSim or DPM generator and fired on a geometry that can either be perfectly aligned or realistically misaligned
1. MC hits on the sensors are then digitized in the 1-Digi macro, and digital row, column and sensor ID data are stored to lumi_digi.root
2. The 2-reco macro reads those digi hits and reconstructs the (x,y,z) position in the panda frame of reference assuming an ideal geometry without misalignment. This is done by the PixelClusterTask and more specifically a PndSdsChargeWeightedPixelMapping object
3. Afterwards, our own PndLmdDim class reads this ideal position, and transforms it to a realistically misaligned position using the transformation matrices for the ideal and a real, misaligned geometry (in simulation only, since we obviously don't have those in the actual detector). This could also be where we introduce the alignment parameters found by software alignment.
4. Those hit positions are saved to the reco.root file after they have been transformed from the position they would have been were the geometry perfectly aligned to the position they are in case the geometry suffers from misalignment (the realistic case)

Firstly I'd like to ask if I understood this correctly.

If so, I wanted to ask how other groups are doing this since the PndLmdDim Class is specific to our detector, or if it were better to change this method to actually introduce the misalignment step earlier in the PndSdsChargeWeightedPixelMapping class. That means the reconstructed hits are stored in their realistic, misaligned position by this class directly. Afterwards, a separate aligner class would transform those positions back to the ideal positions using the alignment parameters found by software alignment. It seems the second version is mapping reality more closely and doesn't depend on our PndLmdDim class, so it could be used by other groups as well. But no such aligner class currently exists.

Best wishes,
Roman