

2006-08-01 LCD Weekly

NG: Plans for updating CVS. Would like to start moving contrib code into production area. Solicit ppl who have reported results to put code into repository. Early to mid-Sept to have a reconstruction "jamboree" to decide recommended default packages, e.g. digitizing, clustering, fitting, etc. for both clustering and tracking. Calorimeter clustering and tracking finding and fitting should be in and defaults should be given. Nick on track fitting.

NS: Fitting code is ready. Compiles with current CVS head. Using the track cheater from Mike Ronan for track finding. Moving code into lcsim contrib area. Can use real tracks once the track finding is ready. Want to introduce detect resolution using smearing code.

NG: Parts of detector?

NS: Doing all parts of detector including disks and barrel.

RC: Fitting pkg could go into production?

NS: Fitter fits track to list of hits that go with the track. Modifies track parameters to the fitted values. Any object that implements Track interface can be used. Need to use BaseTrack though for setters.


RC: If produce a track (set of hits), why do parameters need to be modified? Why can't just write out new list of tracks? Then can implement any way you want and anyone can use it.

NS: Not in the fitter itself. Provided tracks can be modified or can make new collection of tracks.

NG: If have a reason to make a different type of track, derive from BaseTrack.

NG: Rich Partridge?

RP: Using helix fitter. For circle part of fit, agrees to ten decimal places, except sign of DOCA is opposite.

JS: Difference with BaBar is that ours should not have a minus sign . One constant is defined as -1 in BaBar code.

NG: Units of charge or electron charge.

JS: Couple of differences in several places for lcsim.

NS: Mine uses BaBar conventions.

JS: Made recipes for parameters/momentum based on L3 conventions.

NG: There is corresponding LC Note for the track parameters that is public. L3 note is technically for internal use only.

RP: Have modified helical track fitter. Not fit some points in RZ plane. Also returning the separate covariance matrices from circle fit and straight RZ plane – implemented this. Assume that these are uncorrelated fits. True helix fit, there are correlations, e.g. off diagonal. Don't see good way of kludging this. Don't want to spend a lot of time fitting, just want a good fit. For track finding, seems perfectly adequate to treat as separate fits and not worry about the full 5 parameter fit with correlations. Not the original author of this code, so not sure.

NG: What about the finding portion, is it contrib?

RP: Wanted to check the curvature part first. By end of week, hope to put it in.

Fred: Can the track finding code be accessed while fitting portion being developed?

RP: Need some way of estimating the helix. Not meant as final track fit. Just given space points, what's helix to swim to another layer. Shouldn't be the final track fit. Need something that can get track parameters to plug into helix swimmer.

NS: I have code to define track parameters from three space points.

RP: Also similar code in the contrib area which will do arbitrary number of points.

NS: Is there a helix fit?

RP: Circle, line, and helix fitters.

NG: Also a 3 point circle if just want the circle that goes through it. No covariance matrix.

RP: Wanted to use the covariance matrix.

NG: Tracking meeting this week?

RP: Need to talk to Marcel on that.

Fred: Since Vancouver, code that does the hit adding has been completed. Now have the helix fit that uses the track parameters. Committed the code to contrib area.

NG: Uses a cheated version of a vertex stub and extrapolates into barrel.

Fred: Yes. Expect that once Rich Patridge's code is available, should be able to feed tracks in as stubs [?](#).

RP: Pulling seeds out of a track?

Fred: List of MCParticles, find which have hits in vertex detector. For each MCParticle, take list of vertex detector hits and use as input.

RP: Will get a list of TrackerHits when Tracks are used.

Fred: That's great. List of list of hits. As long as there is a way to get a list (one way or another) is fine.

NG: As soon as Nick releases the hit smearing code, should make use of that.

TN: Every hit becomes a smeared tracker hit?

NS: No. A lot of hits in the same layer.

TN: Should dummy up combining and make new tracker hits.

NS: Can deal with any release of MC simulation. Old files had not enough hits.

RP: What are you doing in the forward disks for smearing?

NS: Same thing. Looking for closed hits. No detector digitization.

NG: Can release digitization as package?

TN: Sure but doesn't do anything.

NG: Will get the strip orientation, etc. later.

RP: When do smearing, provide a covariance matrix? (Yes).

NS: Provide matrix into Cartesian coordinates.

RP: Off-diagonal XY. (Yes)

JS: No tracks implement the L3 convention in a way I can reproduce. Have put together a package in my contrib.tracking area. Put in tests to make sure consistent with the helix swimmer. Right now the ReconTrack is not. Found possible bug in helix swimmer. Haven't propagated back into trunk.

NG: Nick Sinev come across any bugs?

NS: Using helix swimmer in the CVS.

JS: Made these tests. Made new FastMCTrack but didn't want MCParticle to be required. Put this capability into contrib area code. Parameterization, L3 parameters are all defined with respect to a reference point. Current FastMCTrack uses the origin as the RP and then creates a track with particle.getOrigin but doesn't swim to reference point. Have a particle that's created by lambda then get back nonsensical answer from FastMCTrack.

NS: Need to swim back to DOCA.

JS: Now tracks that I made now creates a helix, swims the helix to RP (by default 0,0,0), and then calculates at that point the track parameters. Current function isRefPointDCA is nonsense because always true by definition, otherwise parameters don't make sense. Always calculated at DOCA. If change ref point, need to reswim the track.

NG: Should be "is respect to 0,0,0" but just my guess.

JS: Now ReconTrack.getReferencePoint does not return reference point but origin of particle used to create it. Most of the time it is true but not always e.g. for ones I want to vertex. Corrected all these in package I made and made consistent with the helix swimmer, so can make a track, get parameters, make new track with the parameters, and make sure that the points I used can be retrieved from the 2nd track. Documentation of all of this into my contrib area. Put posting to forum outlining this package.

JS: Need to have a mechanism to access a derivative matrix. Not going to try and figure out what everyone is using.

NG: Should all be using the standard definition in track.

NS: Problem with BaBar definition. If curvature close to 0, change sign of track. Sign of D0 changes. May be large. Not a smooth function. Not good for fitting.

JS: Everyone needs to use the L3 conventions. Then can use the track interface for fitting.

RP: People in MARLIN using the standard convention?

JS: LC Note outlines for MARLIN.

JS: $P_t = B * q * r$

Need parameter a to cancel units. Found 5 different values for unit parameter. LC Note uses value of 3. Put my derivation of what a should be.

CM: Which reconstruction should we start with?

NG: Only one representation for track. Couple of implementations are different. Track parameters should be the LCIO (L3) parameters. Doesn't matter whether central or forward.

CM: Also reworking the code for the fitting using the second order Taylor where check distance from point to border and time to arrive and what is smaller.

NG: Design for forward disk tiling?

TN: Not hard part. Hard part is keeping track of 5000 sensor models. Can't just look at one piece, need to throw all tracks to all pieces.

NG: Square hexagons and wedges?

TN: Yes. Biggest difficulty is doing all the coordinate transforms e.g. if at x,y,z what piece I'm in. (geometry system) Little things that need to be done right w.r.t. geometry system. For specific cases, can do something workable but general infrastructure not my specialty.

RC: Assume there are parameters that will get. Maybe Tim Nelson is right person to see which parameters need to be fed in.

NG: Do examples. Circular wedges or square with overlaps? Stereo? etc.

NS: Doing this for CCDs.

TN: For CCDs, get a 3D hit with a CCD so can ignore readout segmentation to first order.

NS: Keep this at configuration level.

TN: Can assume one layer is a big device with pixels on it.