New C++ tracking in Marlin

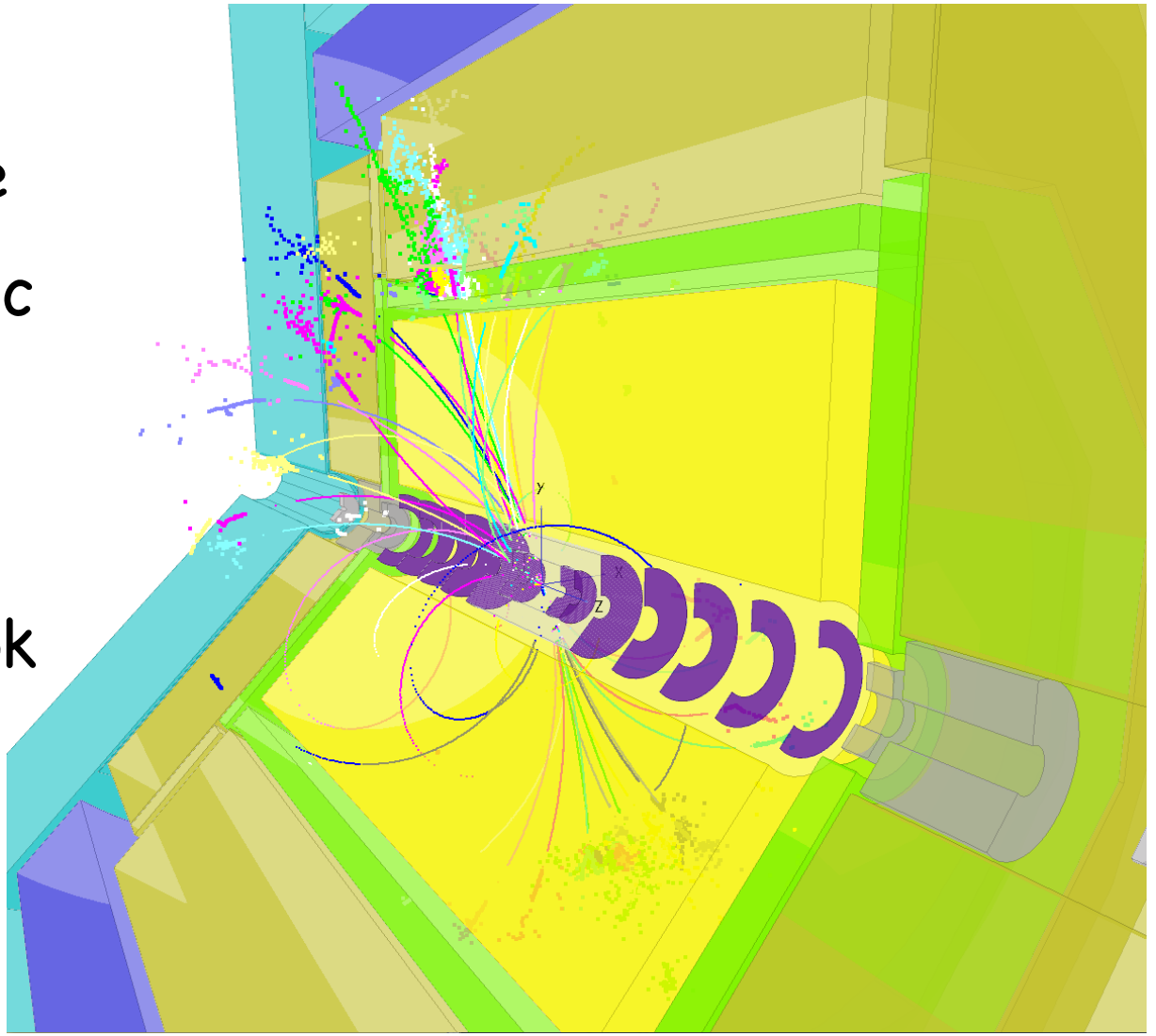
Frank Gaede, Steven Aplin, DESY
Robin Glattauer, OeAW

KILC 2012

Daegu, Korea, Apr 23-27, 2012

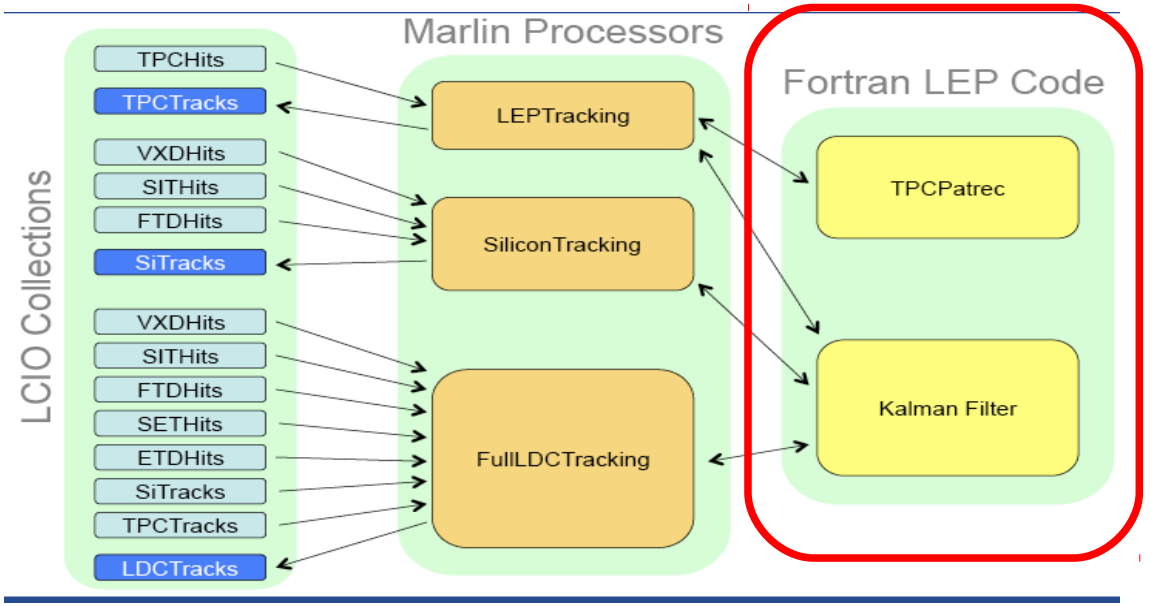
Outline

- Introduction
- KalTest/KalDet
- MarlinTrk interface
- Clupatra TPC patrec
- C++ Si-Tracking
- ForwardTracking
- Summary & Outlook



Introduction

- for ILD we identified the need to replace old f77-tracking code in order to improve the sw maintenance and the performance (background studies, 1 TeV)



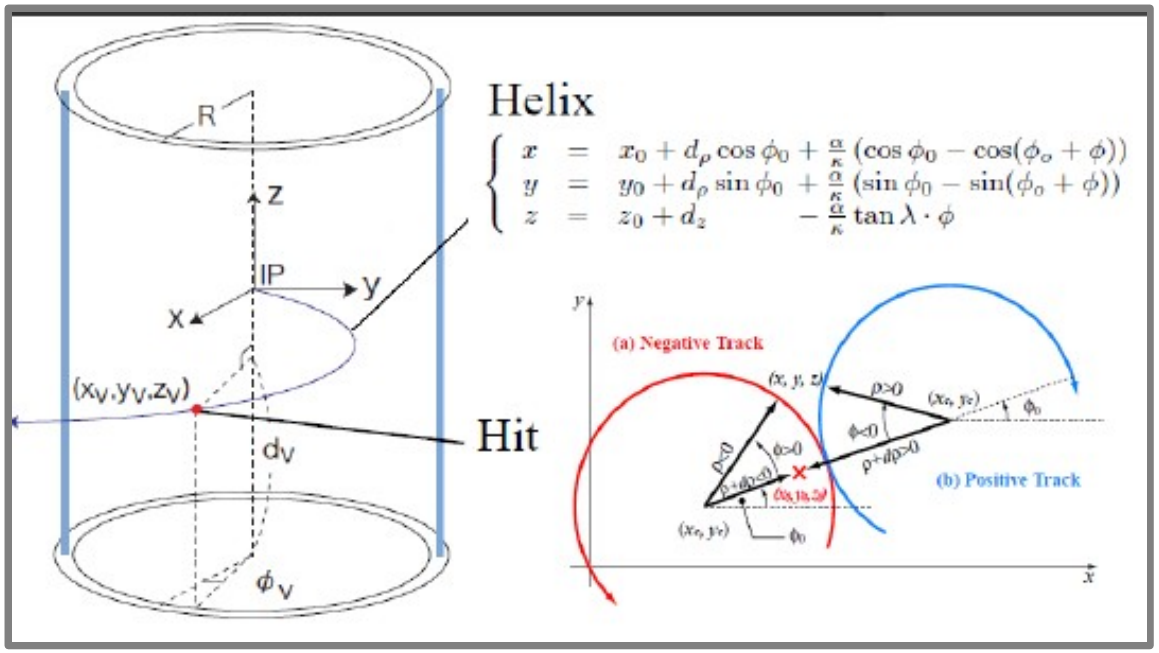
- for this we need:
  - a new C++ Kalman filter tool
  - rewrite the TPC pattern recognition
  - adopt SiTracking to new Kalman fitter
    - possibly improve/develop new algorithm (Fwd !)

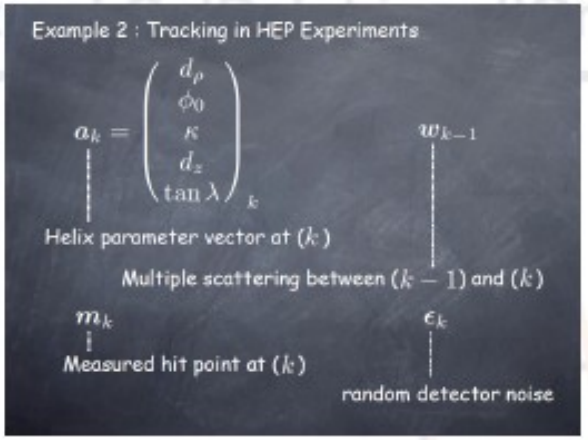
KalTest Kalman Fitter package

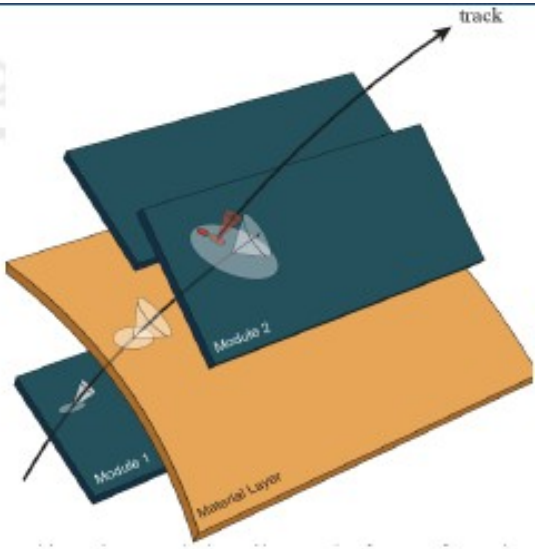
- KalTest
  - Kalman Fitting library (Keisuke Fujii et al)
    - developed in context of Jupiter framework
    - recently added planar measurement surfaces (D.Kamai)
- KalDet
  - detector description (geometry and material) for KalTest
    - recently wrote geometry build up from GEAR
    - including cylindrical measurements for TPC and planar detectors for SI-Tracking – parallel and orthogonal to z (S.Aplin)
- both packages included in iLCSoft since v01-10
- both packages are also used by LCTPC (MarlinTPC)
  - -> try to share as much common code as possible, i.e. is reasonable given the slightly different requirements for testbeam and global detector optimization

KalTest library

- based on ROOT
  - TGeo, TMath, TObjArray
- structured in sub-libraries
  - geomlib - geometry
  - kallib - Kalman filter
  - kaltracklib - Kalman tracking
  - utils - utilities
- built into one libKalTest.so
- users need to define their detector classes (KalDet):
  - TVMeasLayer
    - meas. layer, coordinate to track state transform. ...
  - TVDetector
    - position of meas. layer and material properties

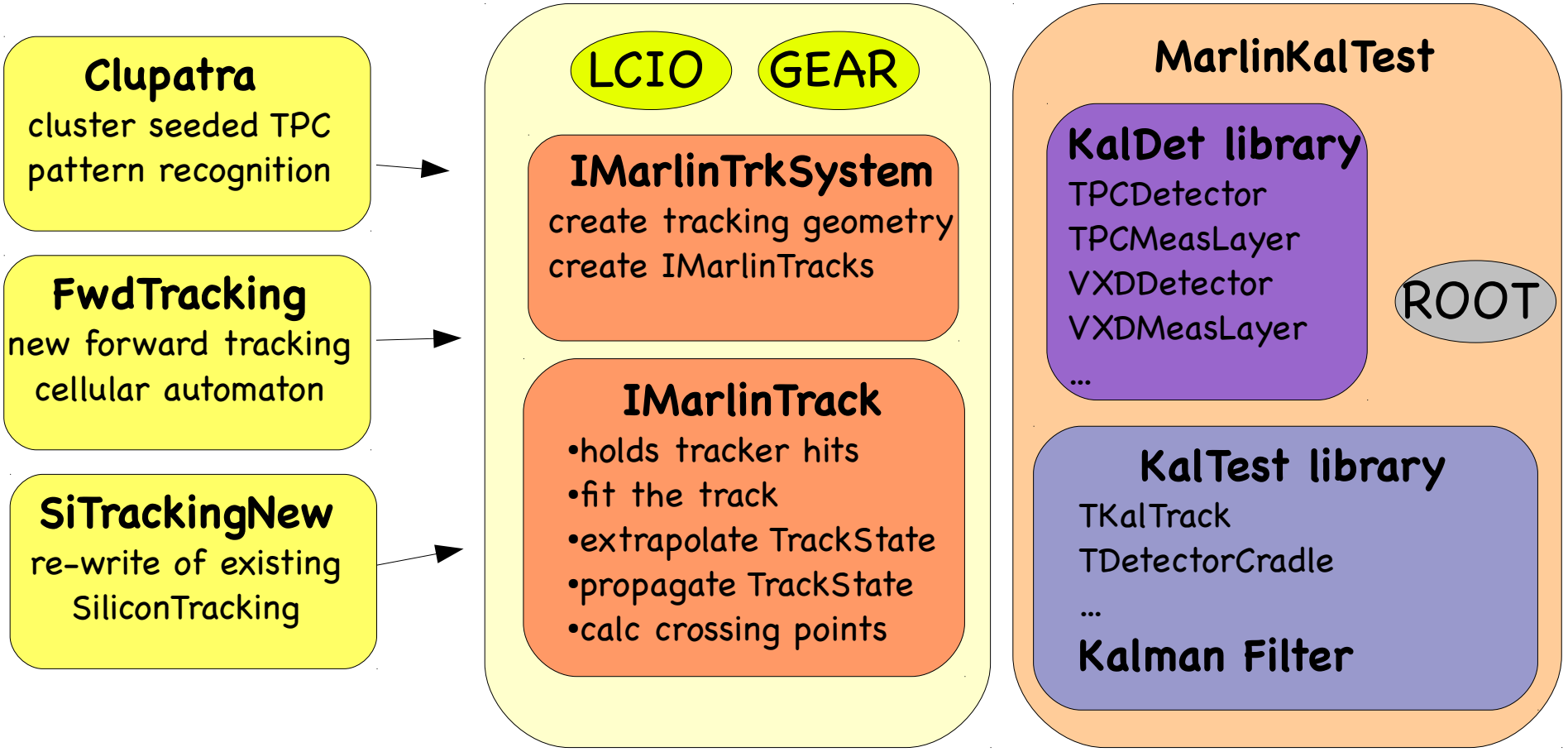






new C++ tracking: MarlinTrk

- new common API for developing tracking code (TPC, Silicon, Fwd)
- provides loose coupling between patrec and fitting
- defined abstract interface IMarlinTrk and implement using KalTest/KalDet
- currently lives in MarlinTrkProcessors



IMarlinTrk & IMarlinTrack interface I

IMarlinTrack interface should provide a convenient interface when using an iterative fitter and also during pattern recognition.

Examples of methods provided:

```
//** initialise the fit using the supplied hits only, using the given order to determine the direction of the track
virtual int initialise( bool direction ) = 0 ;

//** initialise the fit with a track state
virtual int initialise( const IMPL::TrackStateImpl& ts) = 0 ;

//** update the current fit using the supplied hit, return code via int. Provides the Chi2 increment to the fit from adding the hit via reference.
virtual int addAndFit( EVENT::TrackerHit* hit, double& chi2increment, double maxChi2Increment=DBL_MAX ) = 0 ;

//** get track state, return code via int
virtual int getTrackState( IMPL::TrackStateImpl& ts ) = 0 ;

//** get track state at measurement associated with the given hit, return code via int
virtual int getTrackState( EVENT::TrackerHit* hit, IMPL::TrackStateImpl& ts ) = 0 ;
```

continued …

IMarlinTrk & IMarlinTrack interface II

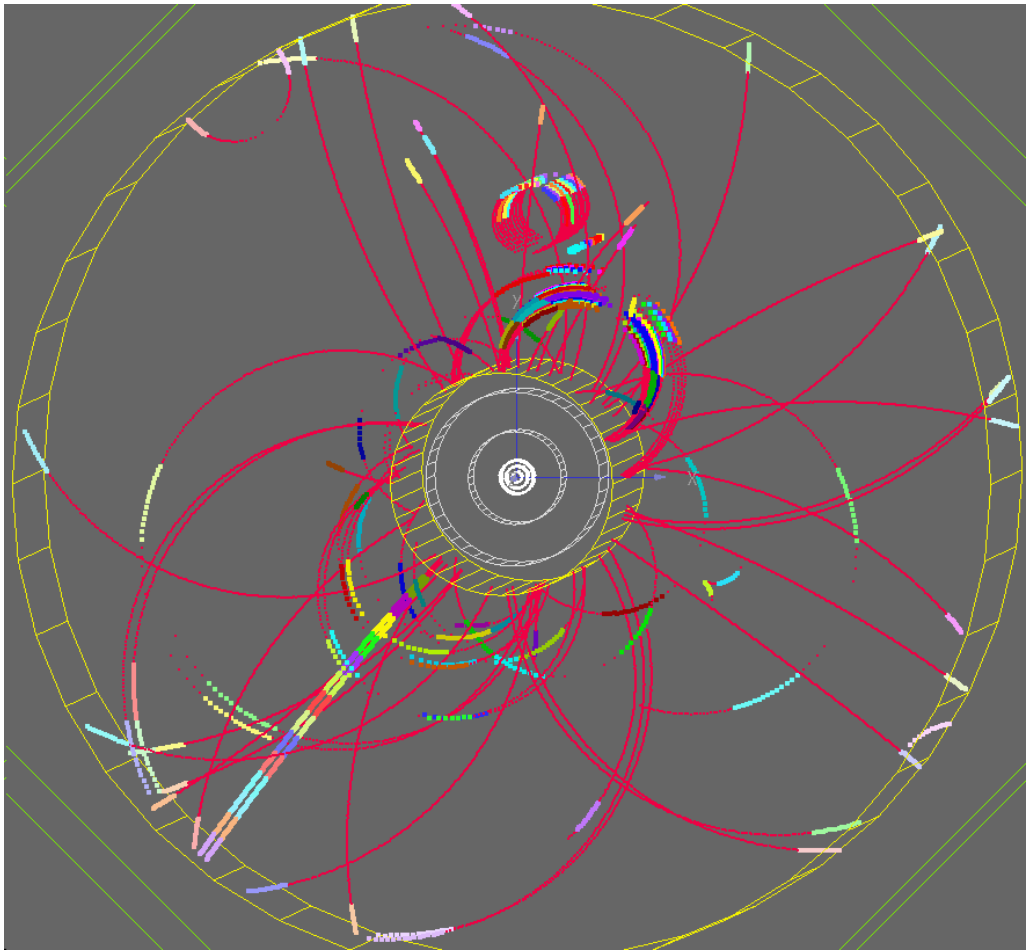
IMarlinTrack interface should provide a convenient interface when using an iterative fitter and also during pattern recognition.

Examples of methods provided:

```
//** propagate track state at measurement associated with the given hit, the fit to the point of closest approach to the given point.
virtual int propagate( const gear::Vector3D& point, EVENT::TrackerHit* hit, IMPL::TrackStateImpl& ts) = 0 ;

//** propagate track state at measurement associated with the given hit, to numbered sensitive layer, returning TrackState via provided reference
virtual int propagateToLayer( bool direction, int layerNumber, EVENT::TrackerHit* hit, IMPL::TrackStateImpl& ts) = 0 ;

//** extrapolate track state at measurement associated with the given hit, to the point of closest approach to the given point.
virtual int extrapolate( const gear::Vector3D& point, EVENT::TrackerHit* hit, IMPL::TrackStateImpl& ts) = 0 ;

//** extrapolate track state at measurement associated with the given hit, to numbered sensitive layer, returning TrackState via provided reference
virtual int extrapolateToLayer( bool direction, int layerNumber, EVENT::TrackerHit* hit, IMPL::TrackStateImpl& ts) = 0 ;

//** extrapolate track state at measurement associated with the given hit, to numbered sensitive layer, returning intersection point in global coordinates
virtual int intersectionWithLayer( bool direction, int layerNumber, EVENT::TrackerHit* hit, gear::Vector3D& point) = 0 ;
```

Clupatra step 1

- **NN-cluster in pad row ranges** (e.g. 15 rows) **– going inwards**
- identify **clean track stubs**
- **extend clean stubs forward & backward using Kalman fitter**
  - add best matching Hit if delta(chi2) < 35.
  - update track state !
  - search in next row



example:

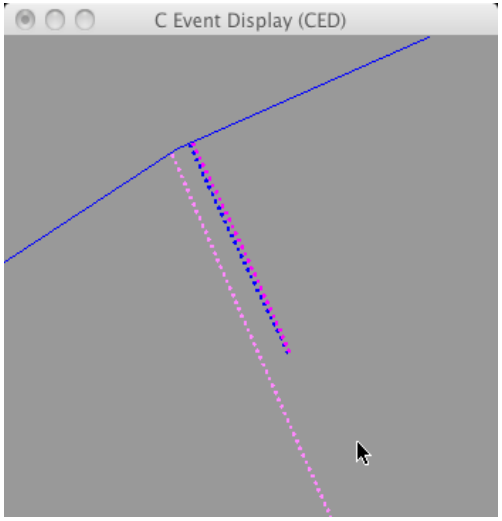
- ttbar event @ 500 GeV

- results in clean tracks and segments for curlers
- little leftover hits (red)

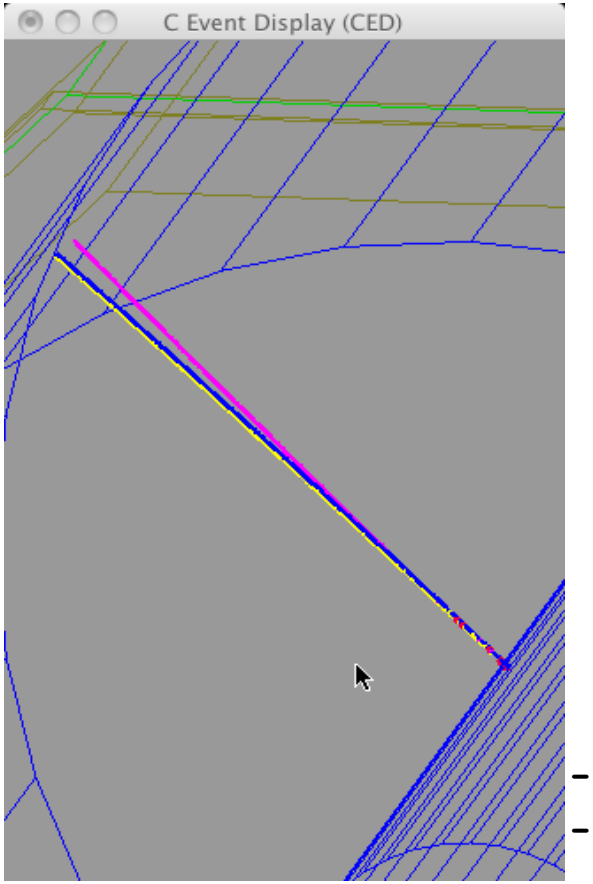
- some very close by tracks lost (fixed in step2)

Clupatra step 2

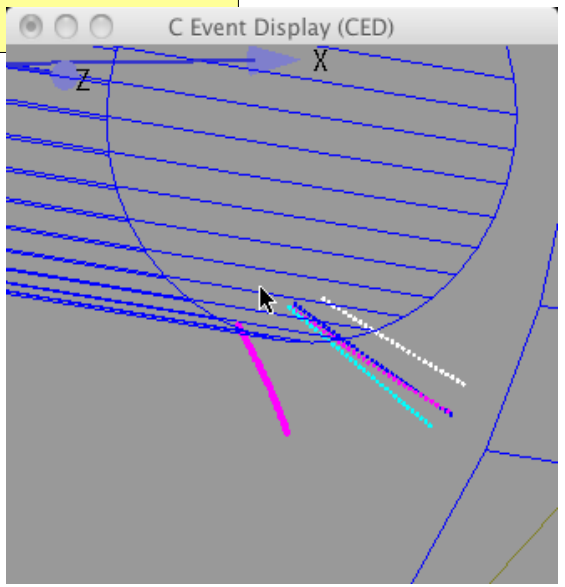
- re-cluster in leftover hits (NN clustering)
- based on pad row multiplicity force into
  N=2,....9 clusters
- apply KalTest fit to throw out falsely merged hits (rare)
  - higher multiplicity: repeat iteratively in smaller row ranges until only three or two tracks left



- gamma conversion in barrel
- forced into two tracks



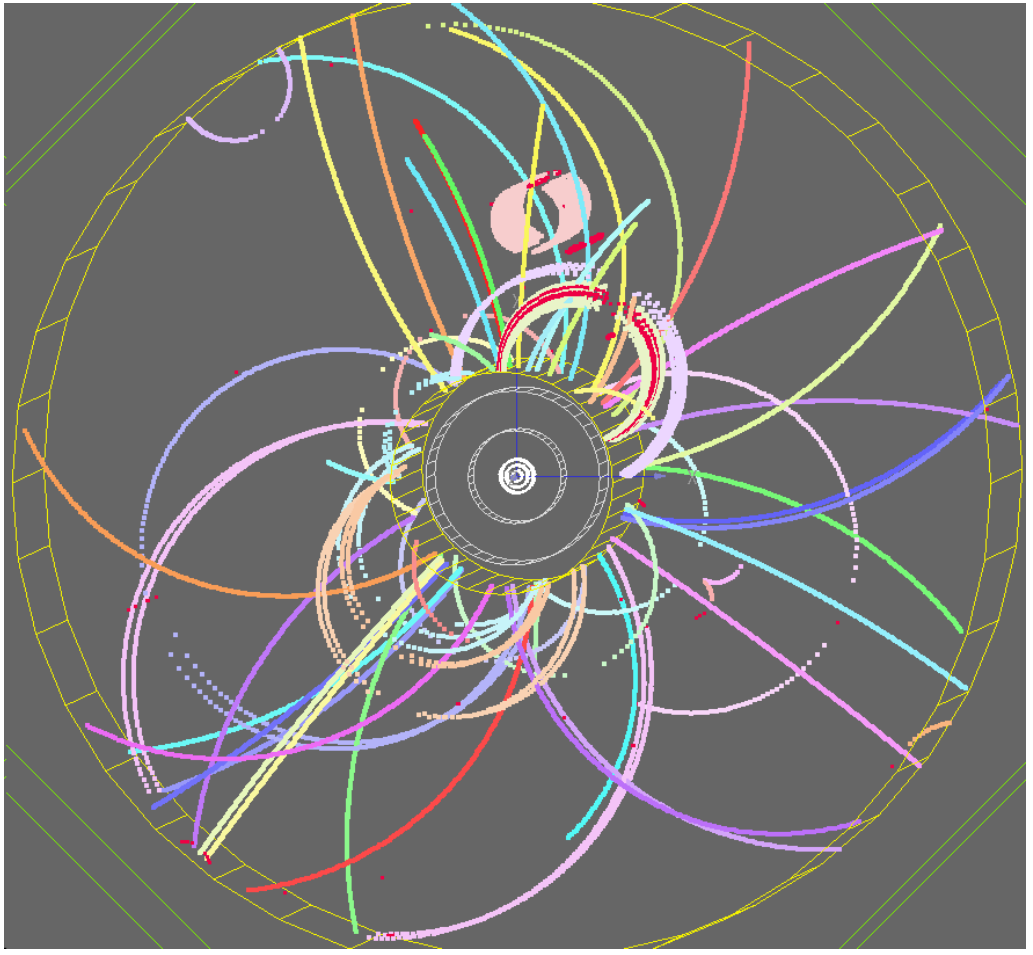
- three prong tau - barrel
- two close-by tracks forced into two tracks



- five prong tau - forward
- three close-by tracks forced into three tracks

Clupatra step 3

- **merge track segments (from curlers)**
- based on rough (O(10%)) criterion for R, delta(xc,yc), tan(lambda)
- disallow overlaps in z

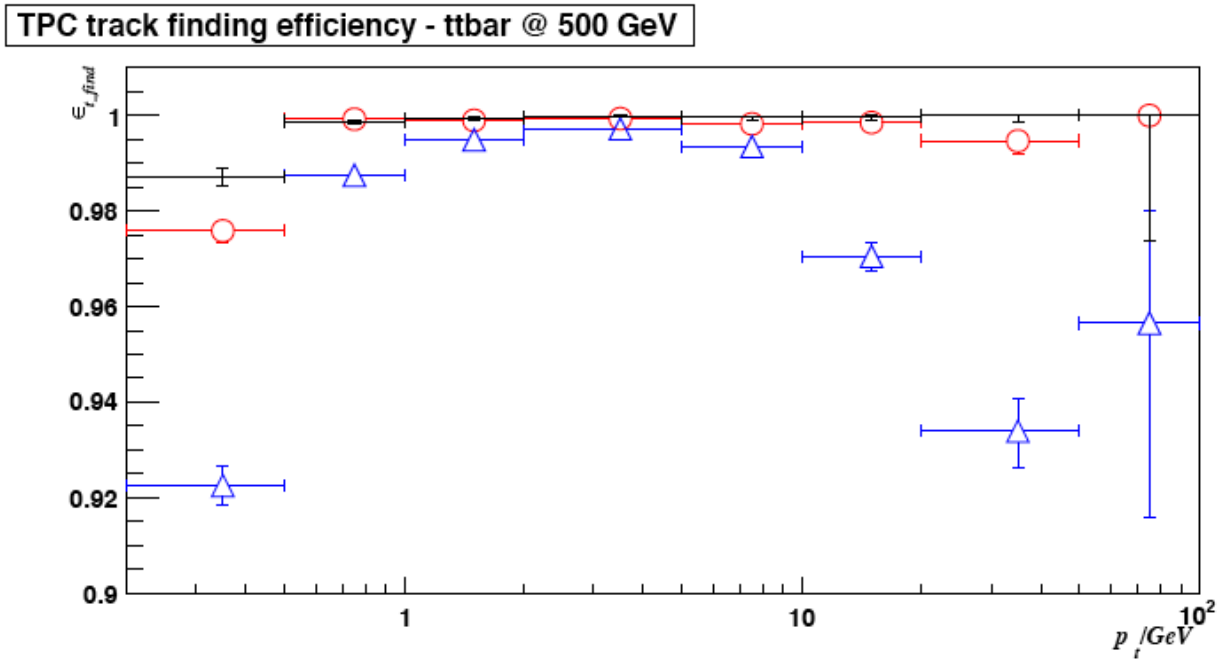


example:

- ttbar event @ 500 GeV

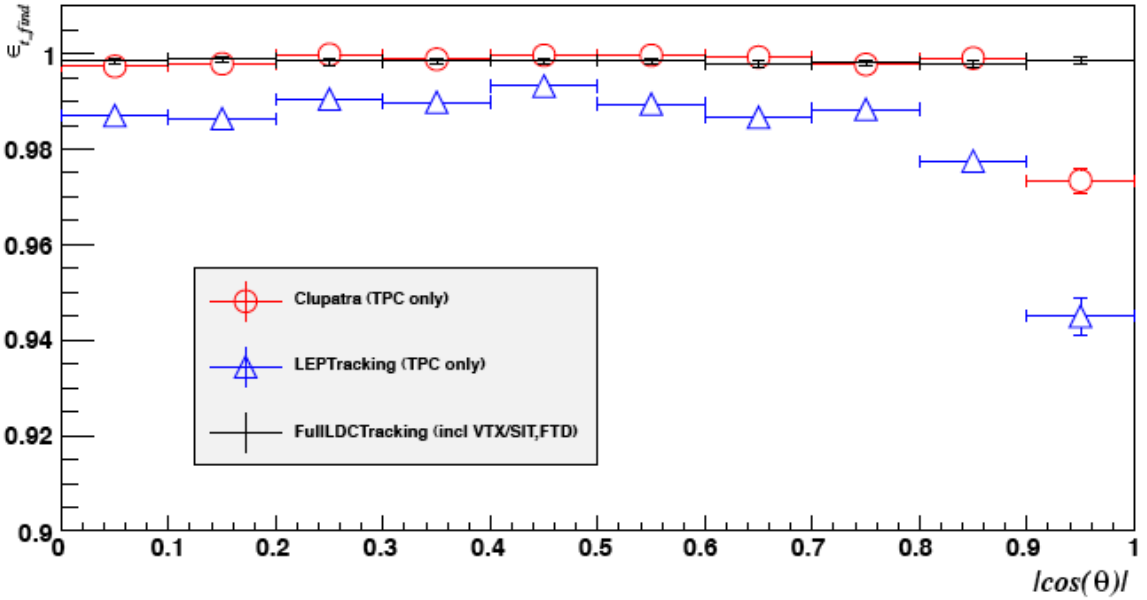
- works nicely
- few segments are not merged
- most of these curler segments where lost in old patrec

track finding efficiency



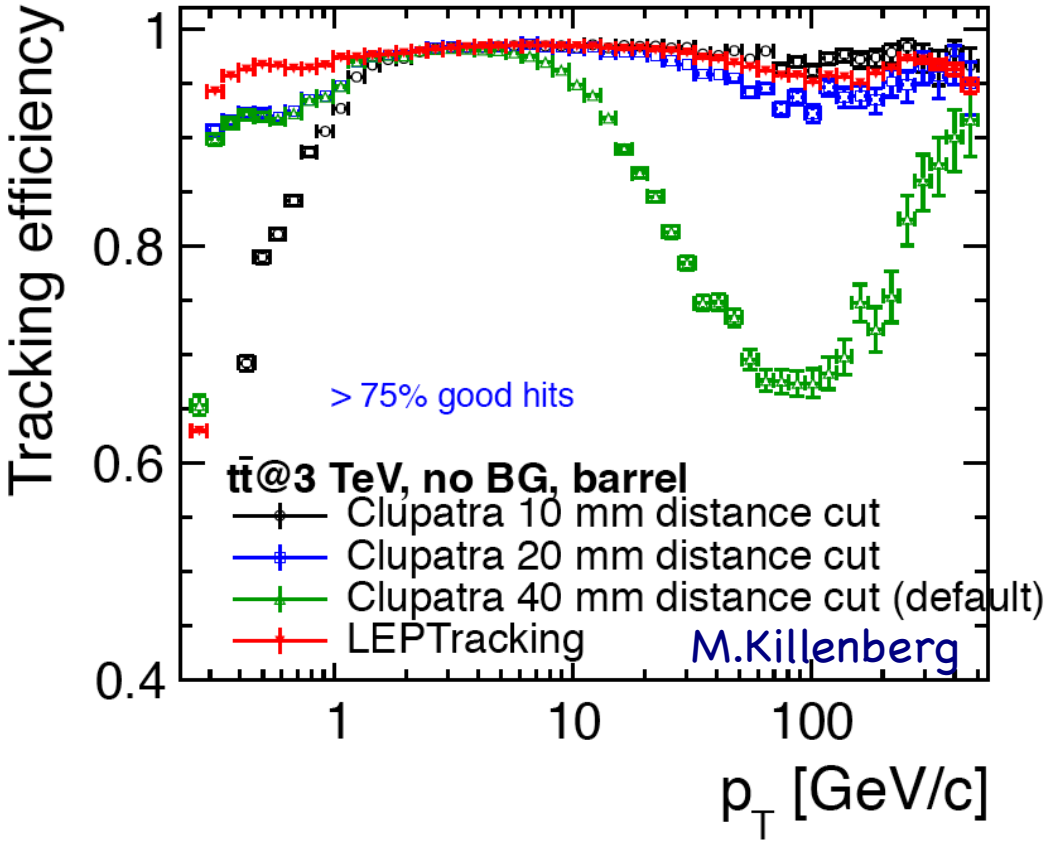
- prompt tracks PCA(IP)<10cm
- > 5 TPC Hits
  - (pt >100 MeV)
  - (|cos(th) |>.99)

- comparison to LEPTracking pattern recognition



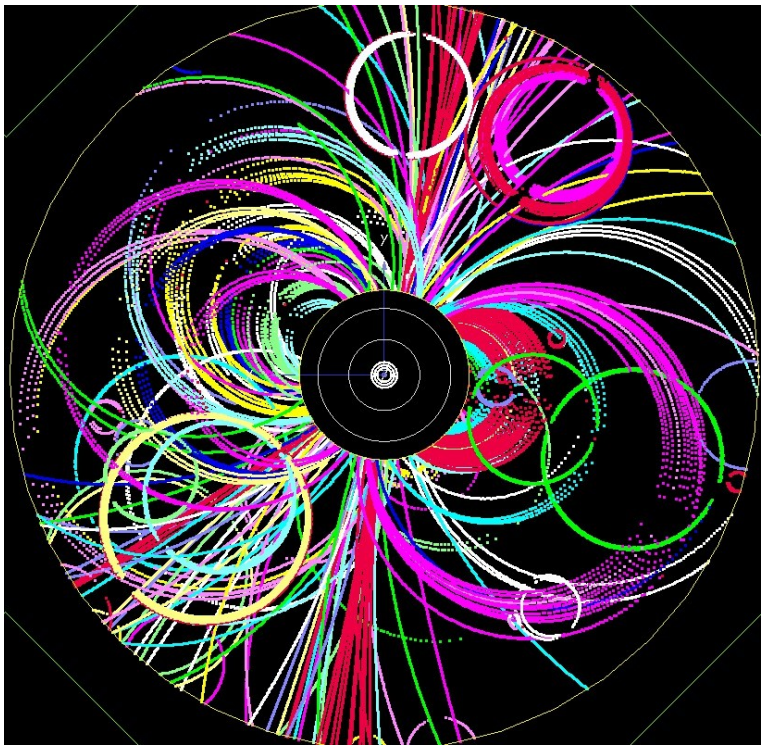
- Note
  - older version of Clupatra
  - no quality cuts applied yet

Clupatra issue @ 3TeV

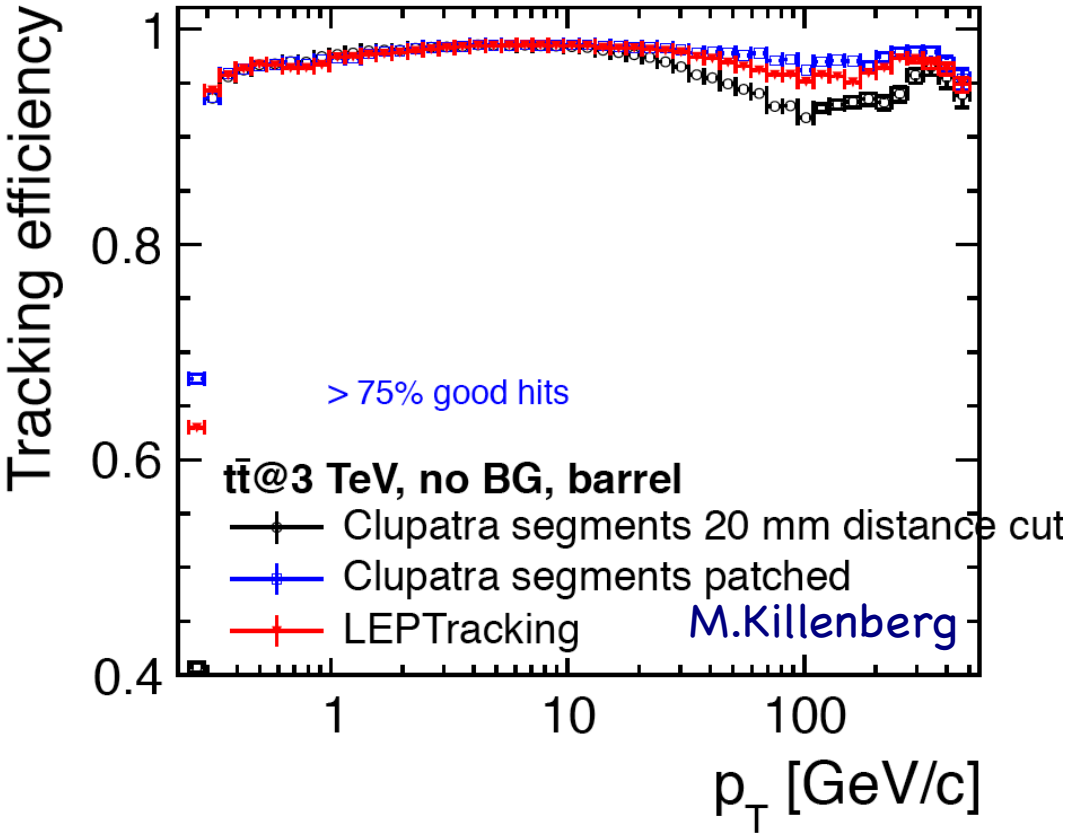


- issue with old Clupatra at higher energies (3TeV)
- poor efficiency with default parameters (optimized @500GeV)
- observed by M.Killenberg for ILD_CLIC detector

- example event
  - high pt tracks in dense jets are sometimes not separated enough along complete length
    -> red tracks are lost

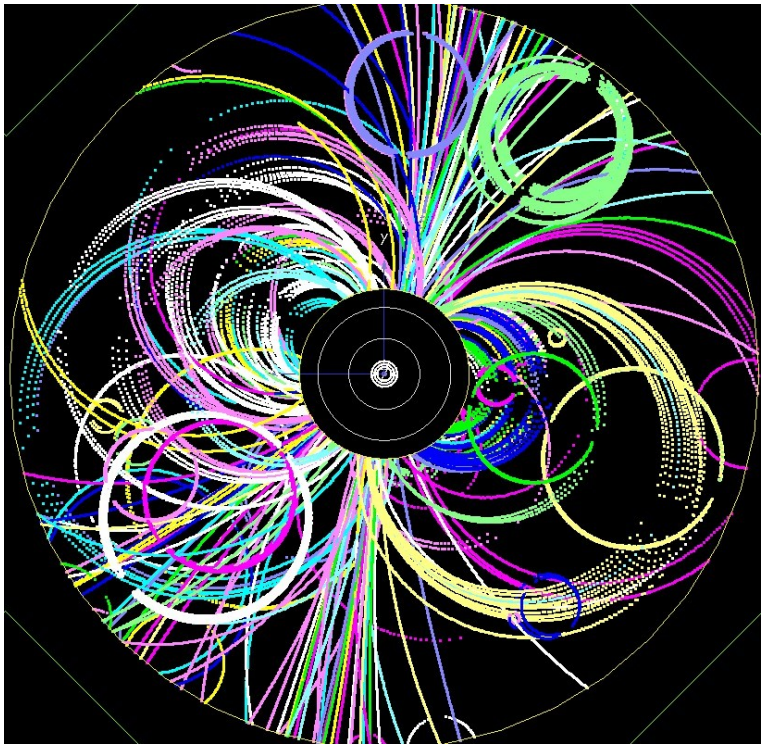


Clupatra issue @ 3TeV fixed



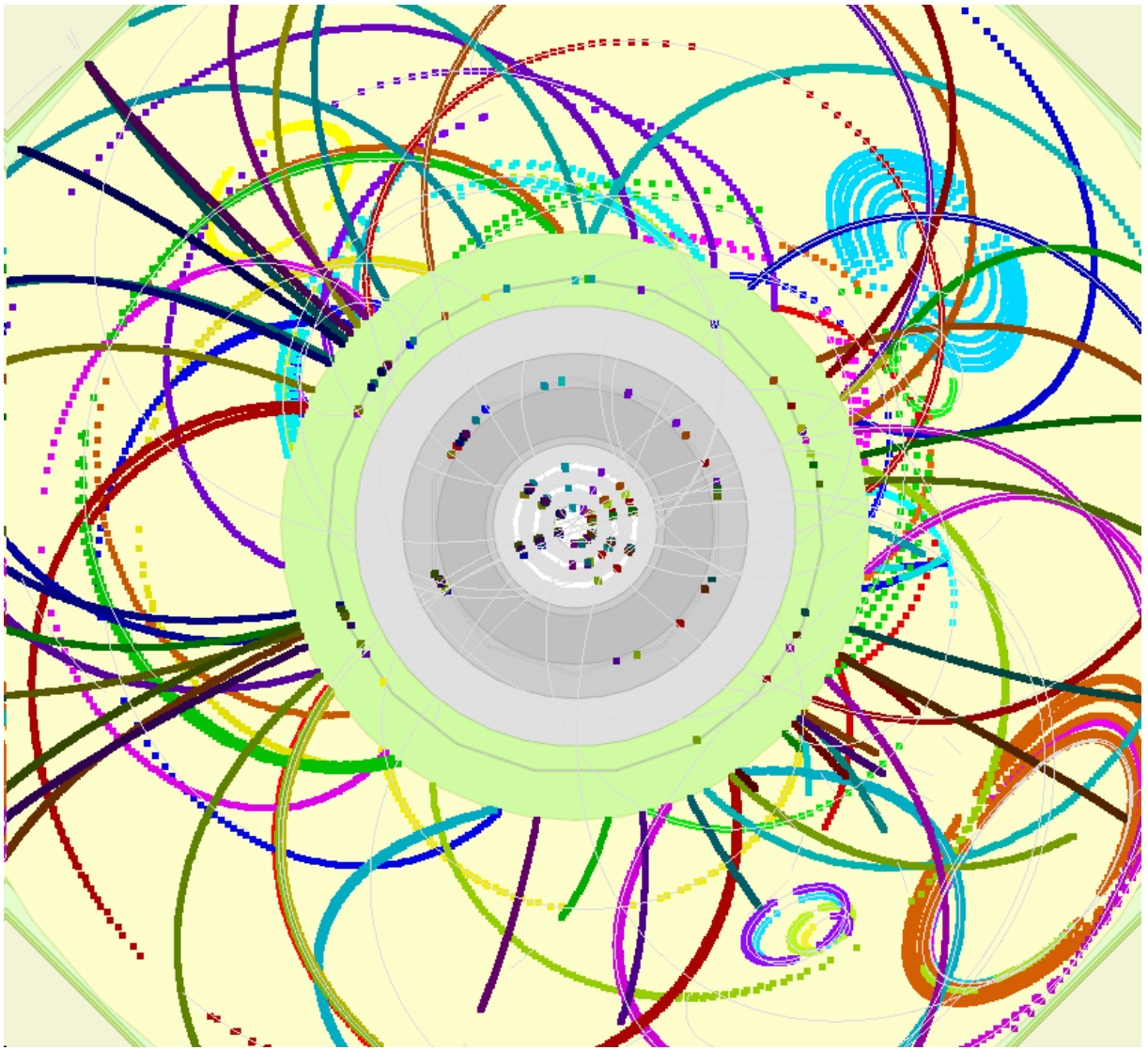
- new Clupatra fixes issue at higher energies (3TeV)
- now loop over different distance cuts for initial seed clustering
- -> track efficiency improved
- further studies needed ...

- example event
  - no tracks are lost with new Clupatra



extending Clupatra inwards

- extended Clupatra to optionally extend hit search further inwards using MarlinTrK interface
- try to pick up hits from SIT and VXD
  - FTD not yet
- **could use as backup strategy for large background**
- standard ILD tracking:
  - have standalone tracking in TPC and Si-trackers and then merge

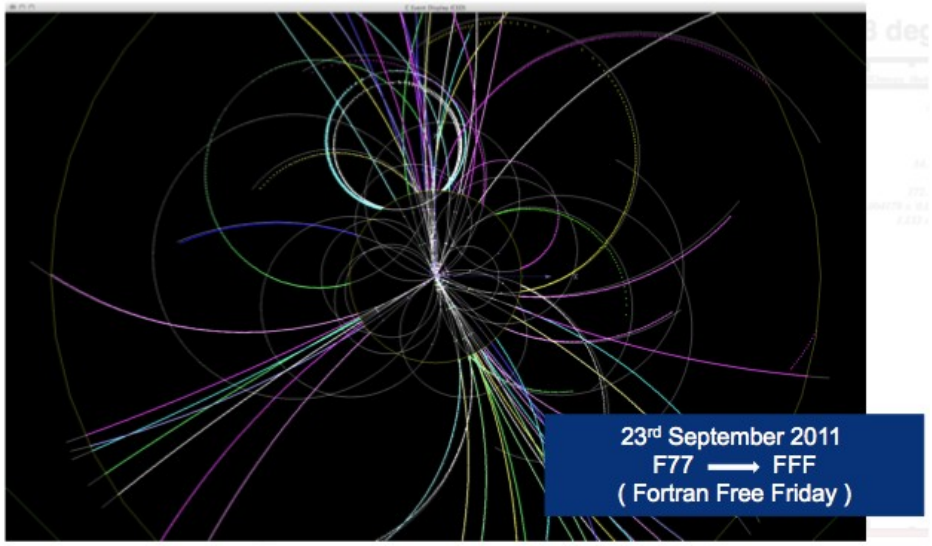


fisheye view of Clupatra tracks with SIT and VXD hits picked up (ttbar @ 500 GeV)

Clupatra memory usage

- large memory consumption for Clupatra observed (T.Tanabe)
  - 1.2 GByte for ILD event w/ 70 k TPC hits
  - -> not a major problem for ILD DB - however prohibitive for CLIC w/ background (several 100k hits)
- Clupatra uses KalTest Kalman filter for track extrapolation - keeping all tracks in memory (twice) until end of event
  - KalTest stores three track states per hit (predicted, filtered, smoothed)
  - every track state has 5 TMatrixD objects (264bytes)
- modified Clupatra to mostly keep only one complete track fit in memory at any given time
- memory usage dramatically reduced
- -> should be usable for CLIC as well now

new Si-Tracking - full tracking



ttbar event @ 500 GeV reconstructed using Clupatra and SiliconTracking_MarlinTrk
then combined into full tracks using FullLDCTracking_MarlinTrk

- shown in Granada @ LCWS11:
- re-write of SiliconTracking and FullLDCTracking (from LOI) using the new MarlinTrk track fit
- using 3d space points in SIT/SET and FTD (as was done in LOI) yet with planar wafers

- since then:
  - extended Gear with MeasurementSurfaceStore
    - local coordinate systems on rotated planes
  - write out proper 1d strip measurements for Si-Trackers using the new lcio::TrackerHitPlane
    - x,y,z, u, v, du, dv
  - implemented 1d fit in MarlinTrkKalTest (KalDet)
  - implemented SpacePointBuilder (next slide)

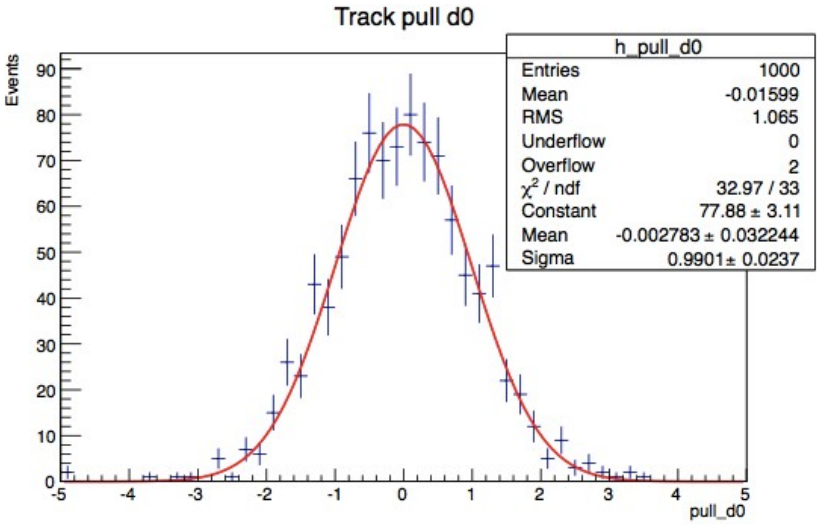
SpacePointBuilder

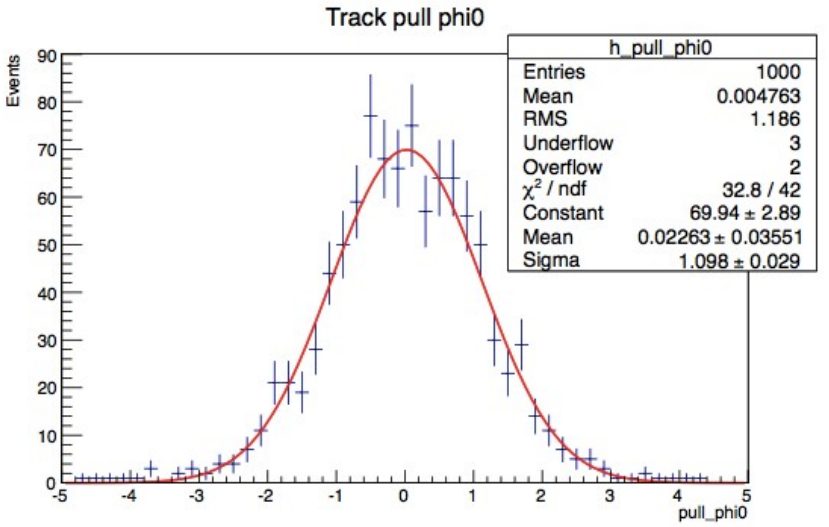
- Marlin processor that combines pairs of digitized 1d TrackerHitPlanes from double layers with strip stereo angle into on TrackerHits with 3d space points - including somewhat correct errors
- all possible hit pairs that result in hits laying within the bounded surface of the wafer (rectangle/trapezoid) are used -> including ghost hits
- these space points are used for pattern recognition
- the final track fit (after arbitration for doubly used hits) then uses the proper 1d measurements and errors

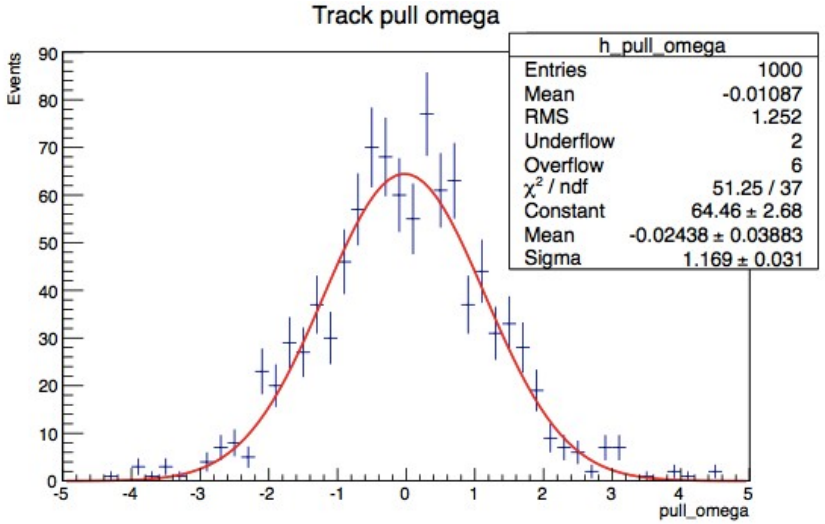
=> major step in realism wrt LOI

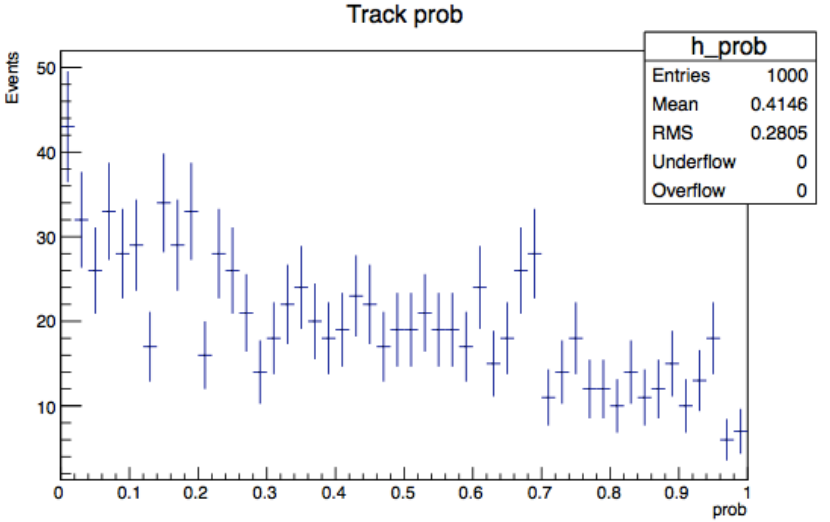
first test of new Si-Track fit

single 10 GeV muons central: VXD/SIT (no TPC)



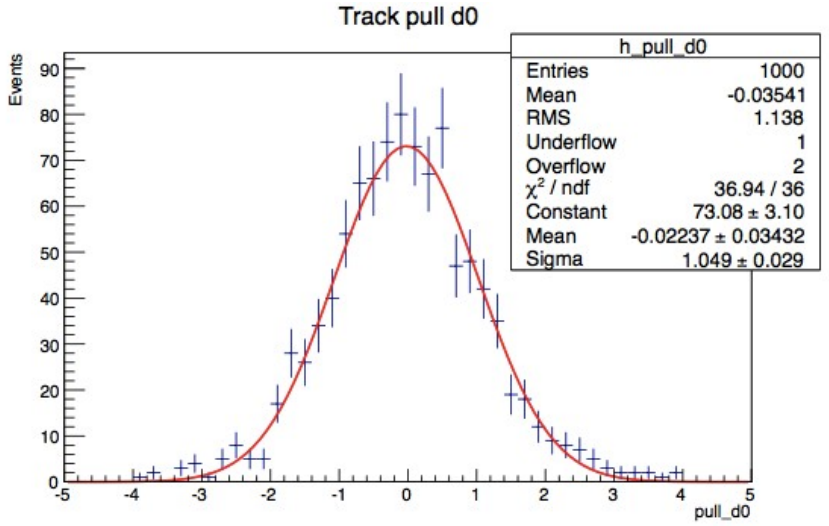


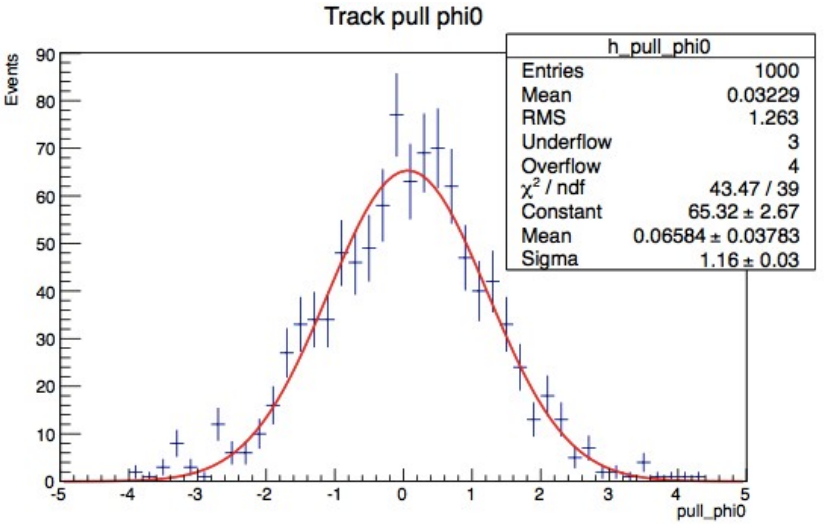


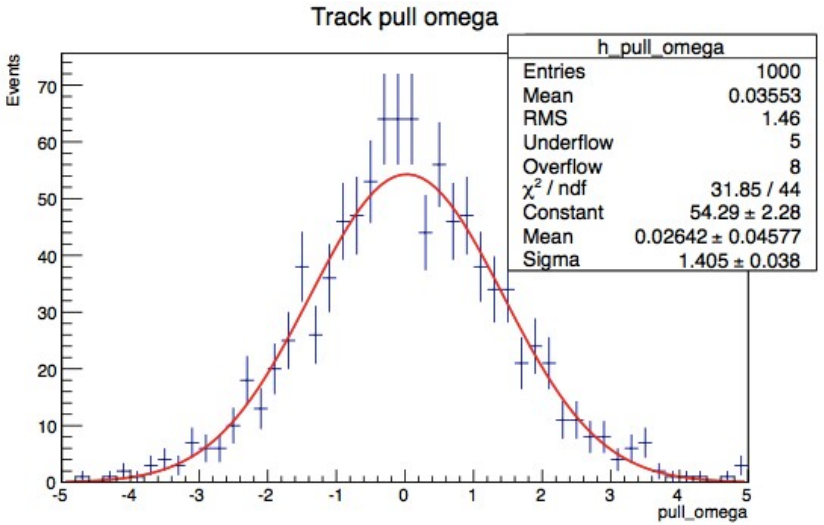


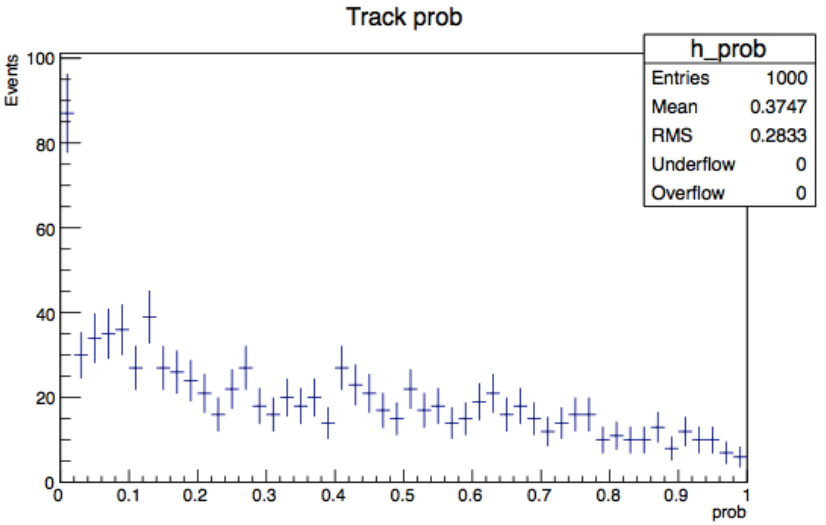
first test of new Si-Track fit

single 3 GeV muons central: VXD/SIT (no TPC)



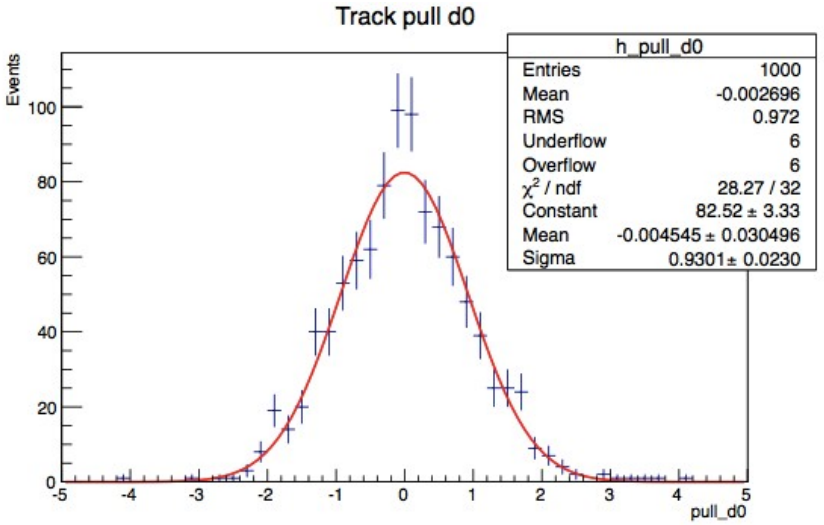


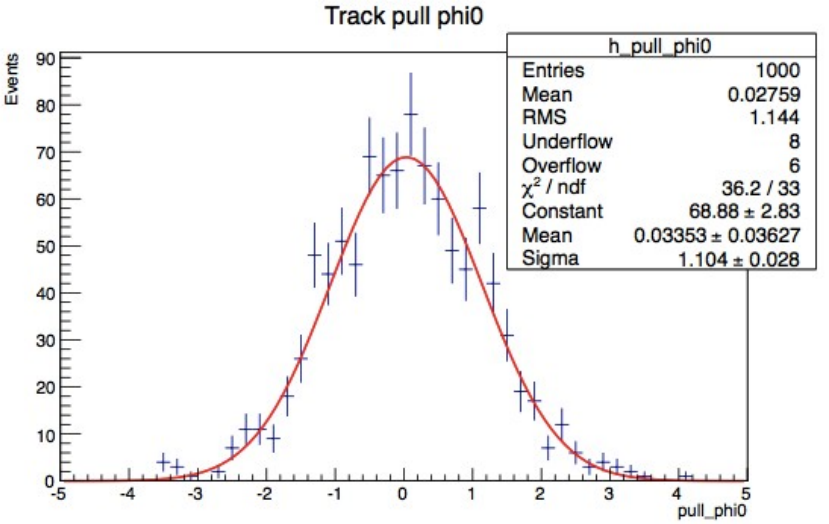


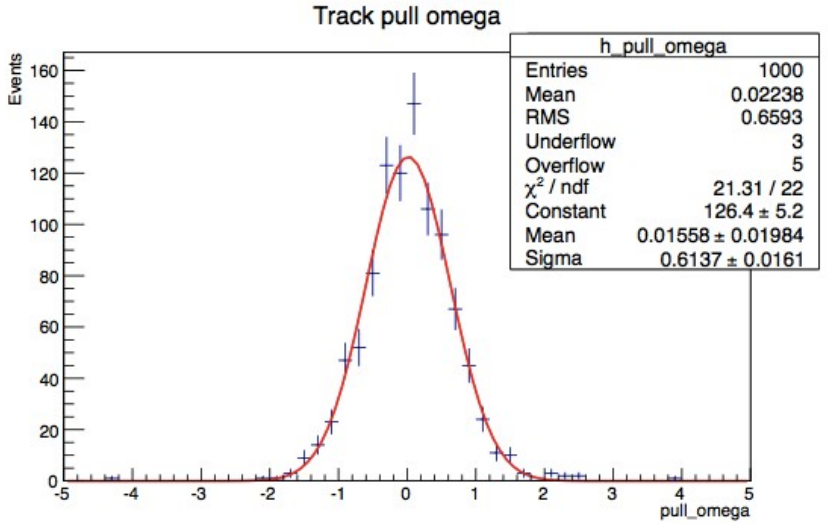


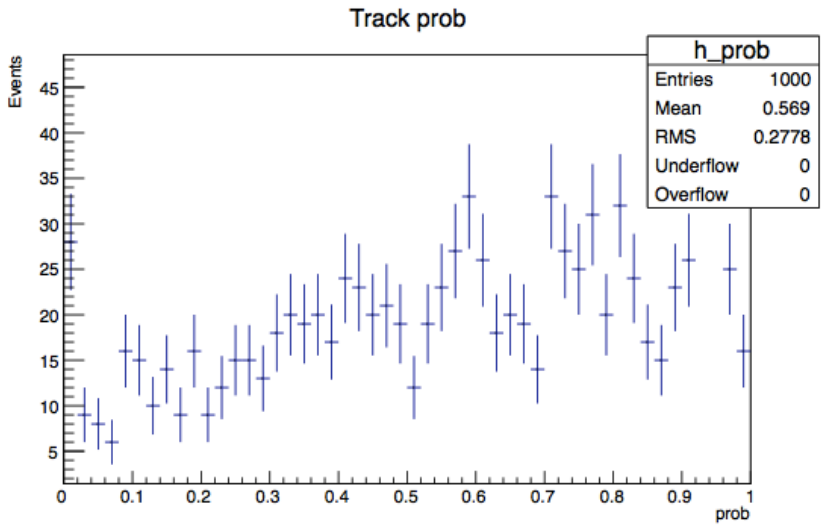
first test of new Si-Track fit

single 3 GeV muons forward: FTD (no TPC)

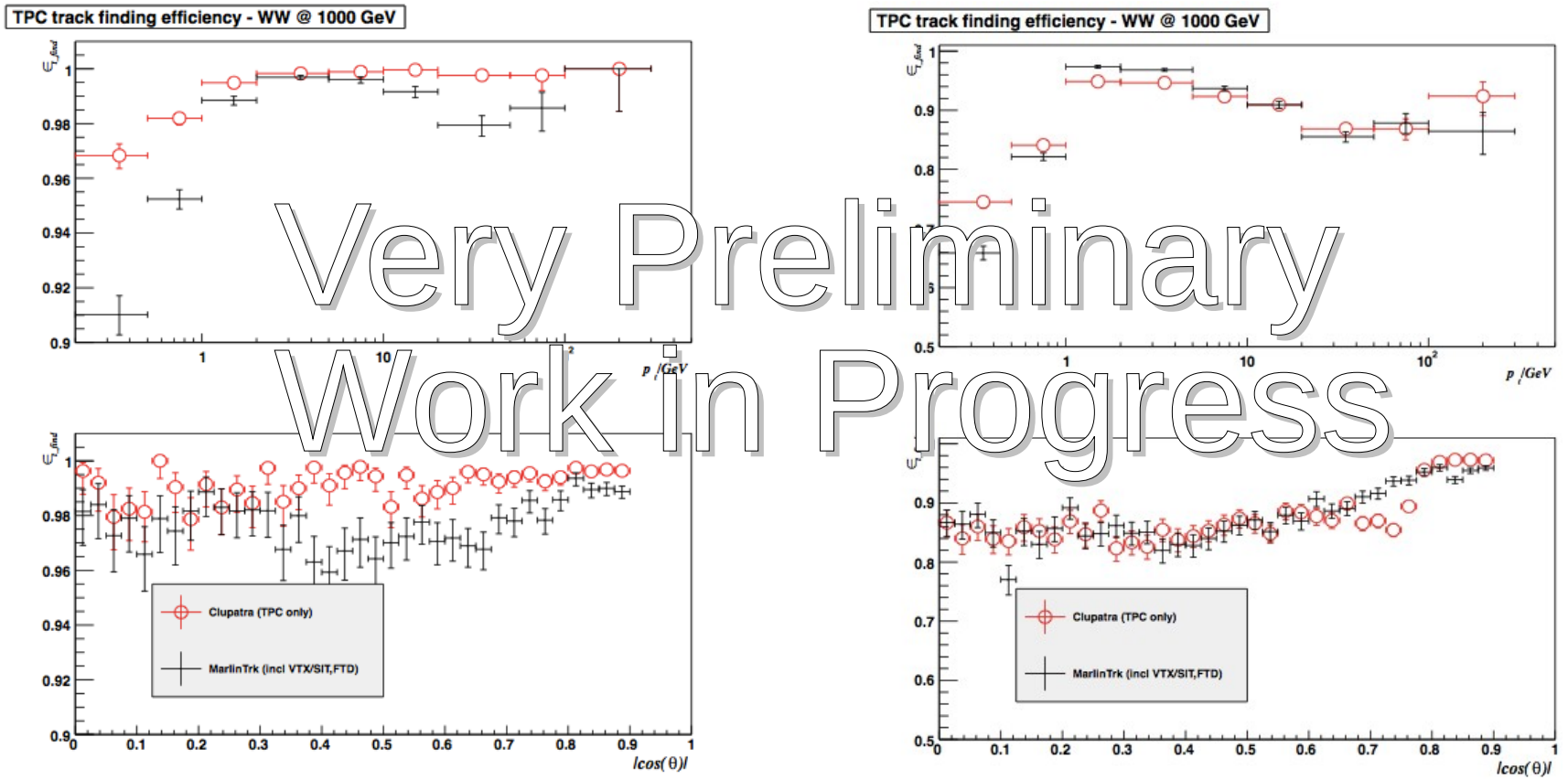








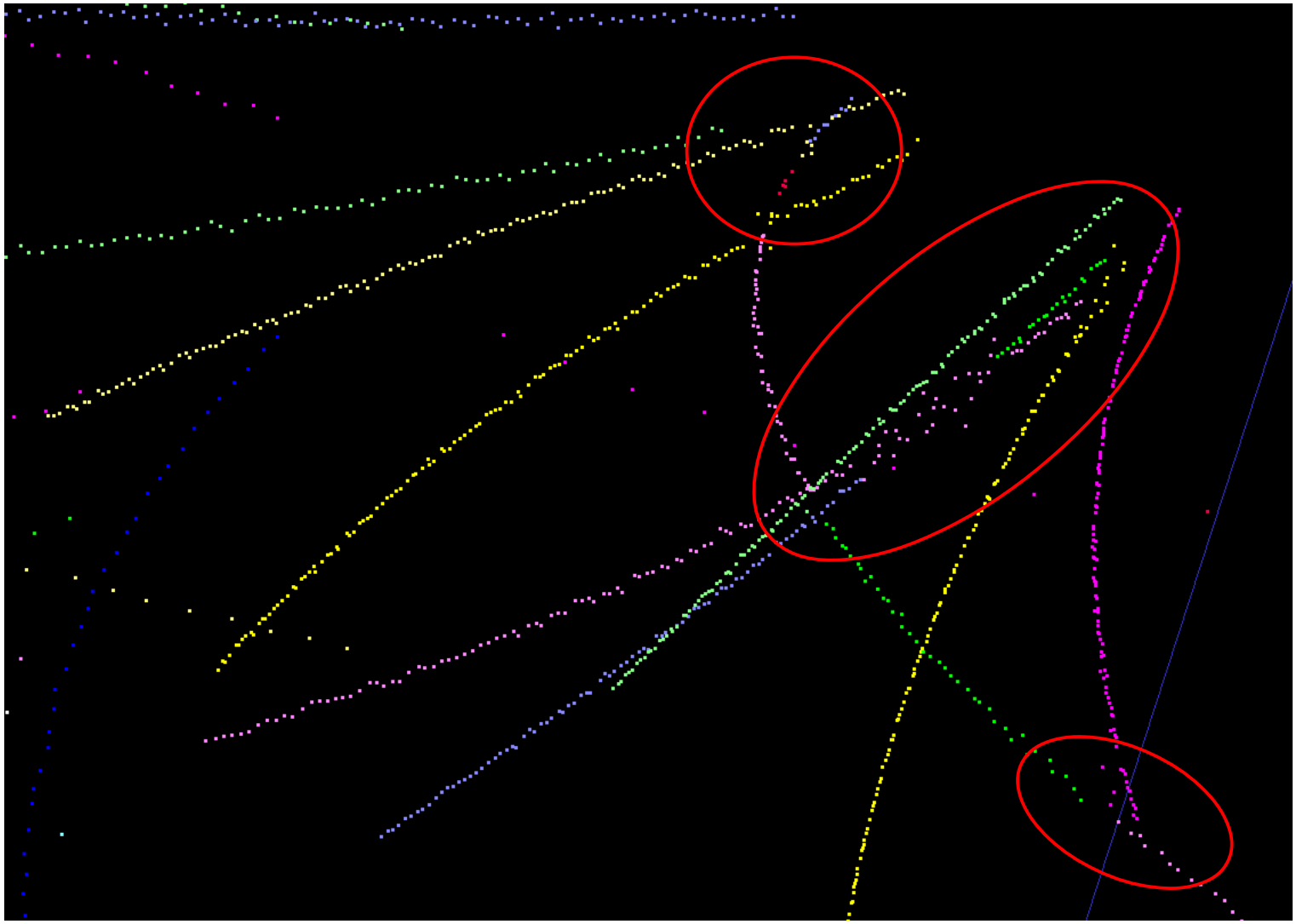
first look at efficiencies in v01-13-05



Very Preliminary
Work in Progress

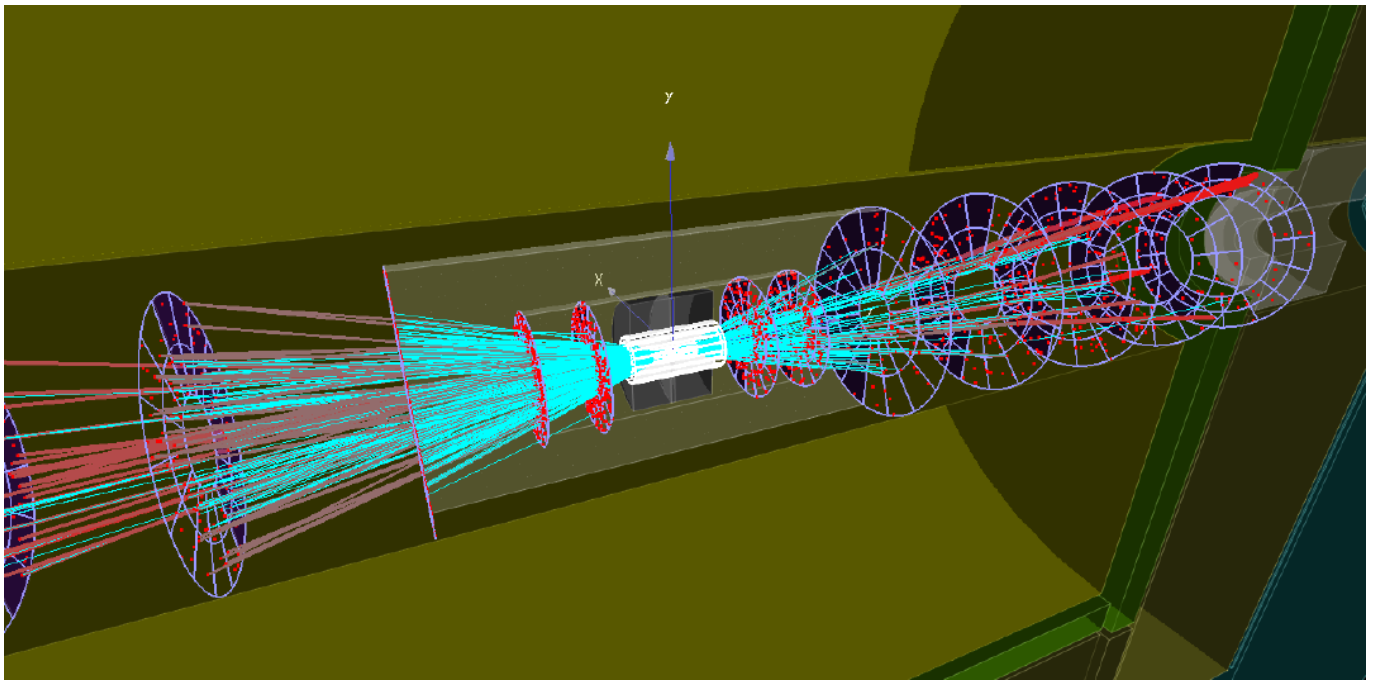
- first look at clupatra efficiencies:
  - would be acceptable (incl. 75% true hit cut - left)
  - but obvious issue w/ split tracks (right)
- first look at MarlinTrk efficiencies (incl. Clupatra):
  - work to be done - loss partially understood:
    - probability cut and poor errors...

Clupatra split tracks example



- currently working on merging of split tracks – due to merged TPC hits in dense environments (> 1TeV)

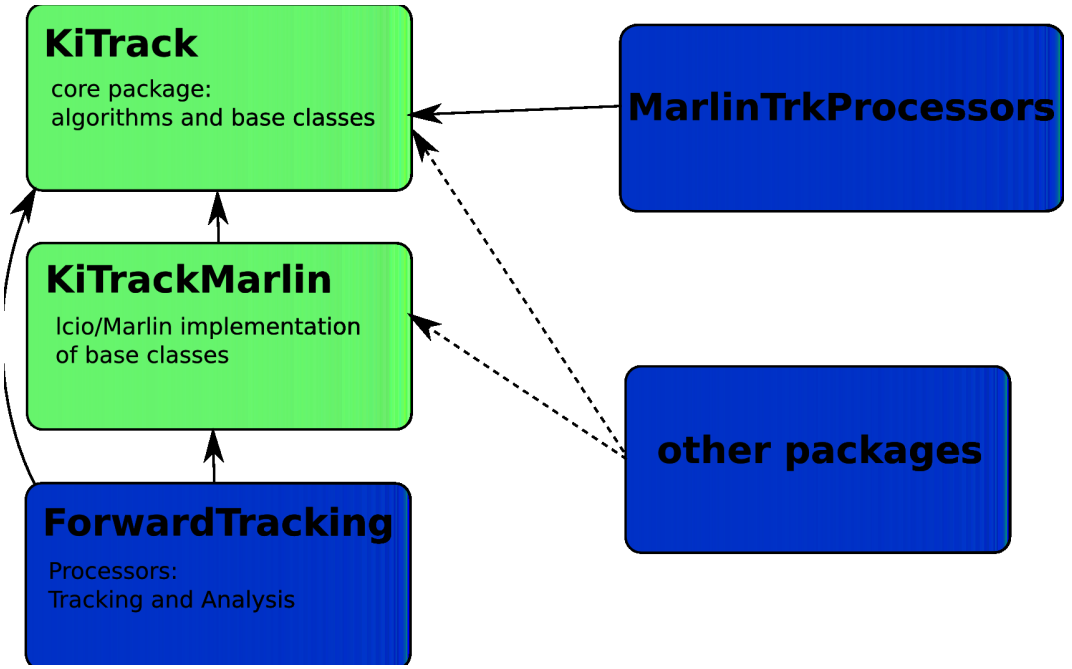
new ForwardTracking



- ForwardTracking: new standalone forward tracking package
- based on
  - Cellular automatons (find track candidates)
  - Hopfield networks (arbitrate between candidates w/ mutual hits)
- current release v01-02 included in ilcsoft v01-13-05

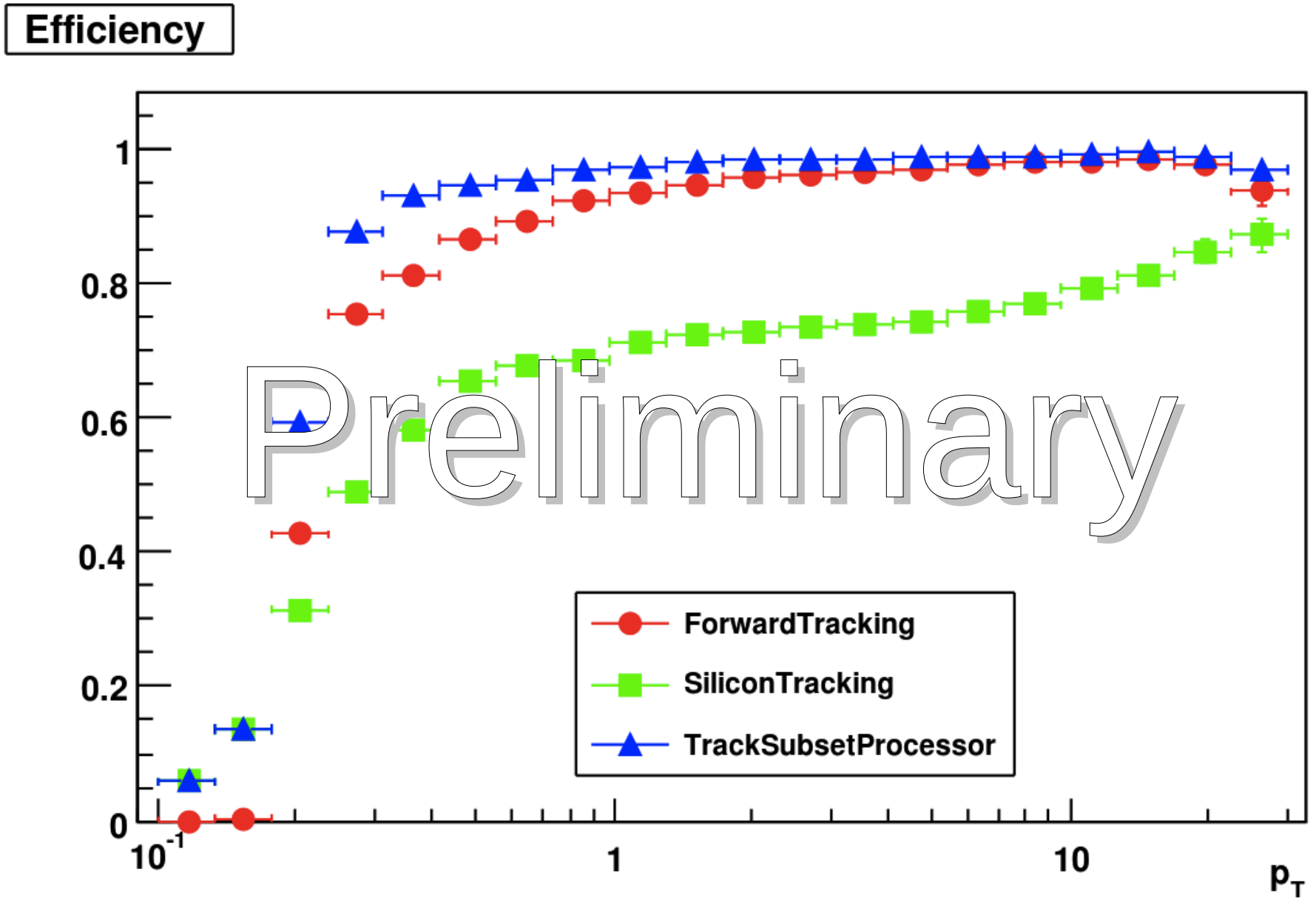
new development in ForwardTracking

- split up software package ForwardTracking :
  - KiTrack:
    - standolone algorithms
    - CellularAutomaton
    - Hopfield Net
  - KiTrackMarlin:
    - iLCSoft dependency
  - ForwardTracking:
    - actual MarlinProcessor



- new processor TrackSubsetProcessor
  - combines different track collections to one and uses the Hopfield Neural Network to search for the best subset of non conflicting tracks
  - combines ForwardTracking and SiliconTracking tracks into one track collection

forward tracking efficiency



Summary & Outlook

- new tracking for ILD has been developed and released in iLCSoft v01-13-05:
  - Clupatra (topological TPC patrec)
  - C++ re-write of SiliconTracking and FullLDCTracking
  - ForwardTracking
- recently included the proper treatment of 1d hits for double strip stereo layers in SIT/SET and FTD
- started to combine ForwardTracking and SiliconTracking

- To Do
  - test everything more thoroughly
  - understand (in)efficiencies (and fix issues)
  - iterate the material description to get probabilities and pulls right
  - time before DBD Monte Carlo production is short
    - let's see...