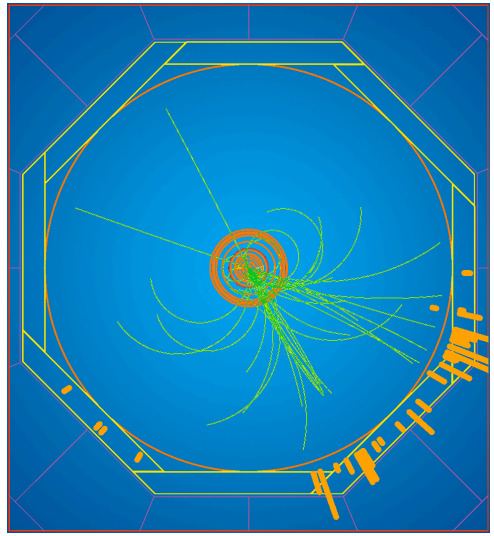
ILD Tracking

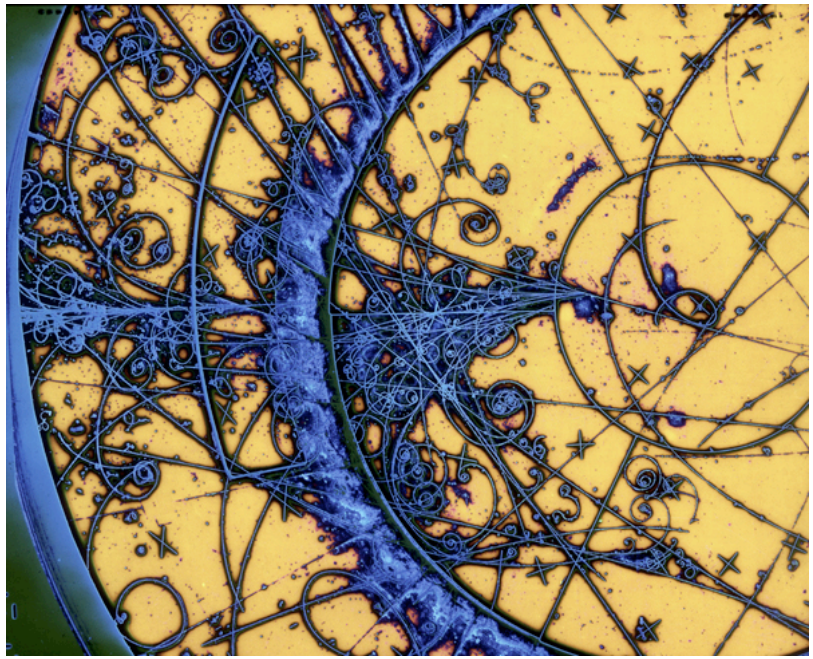
Steve Aplin
DESY

LCWS 2011 Granada
28th September 2011



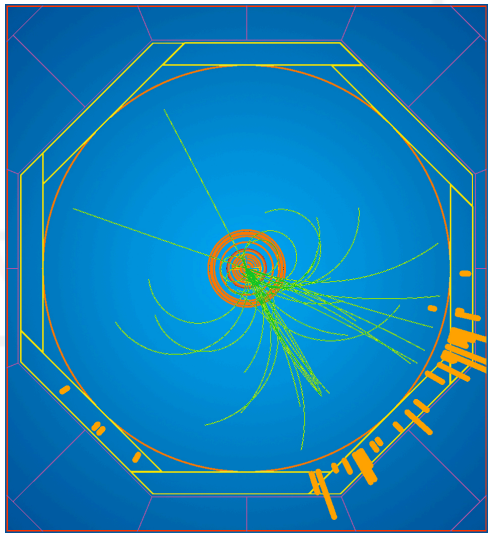
iLC SOft

- Current Status
- Towards the DBD



Tracking @ ILC

- Tracking reconstruction goals:
  - Momentum resolution $\frac{\Delta p}{p^2} < 5 \times 10^{-5} GeV^{-1}$
  - Impact parameter resolution $\sigma_{r\varphi} < 5\mu m \oplus \frac{10}{p(GeV)\sin^{3/2}\theta}\mu m$
  - Very high efficiency $\varepsilon > 99\%$
  - Very low material budget

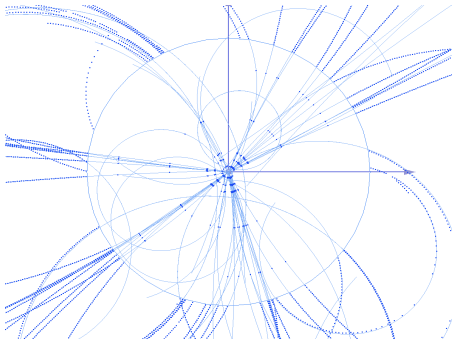


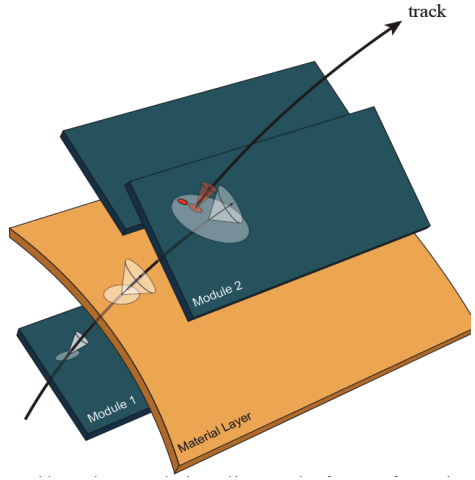
ILD Track Reconstruction

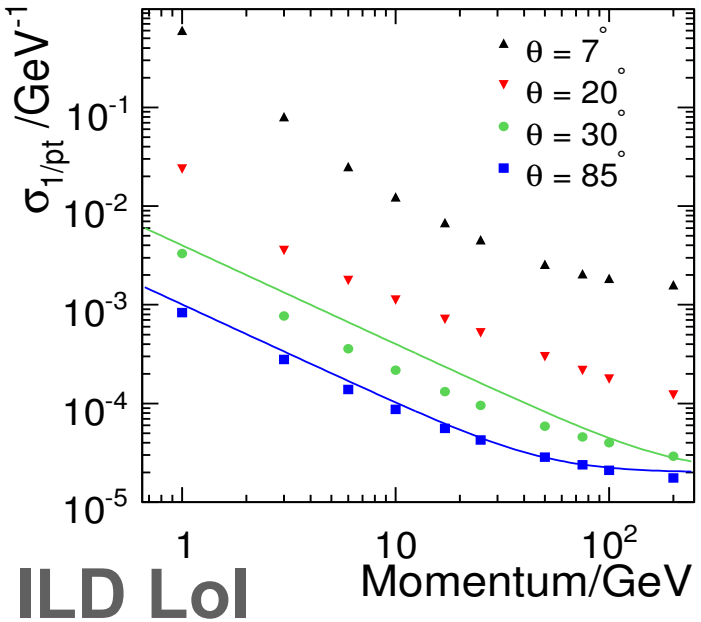
Full pattern recognition

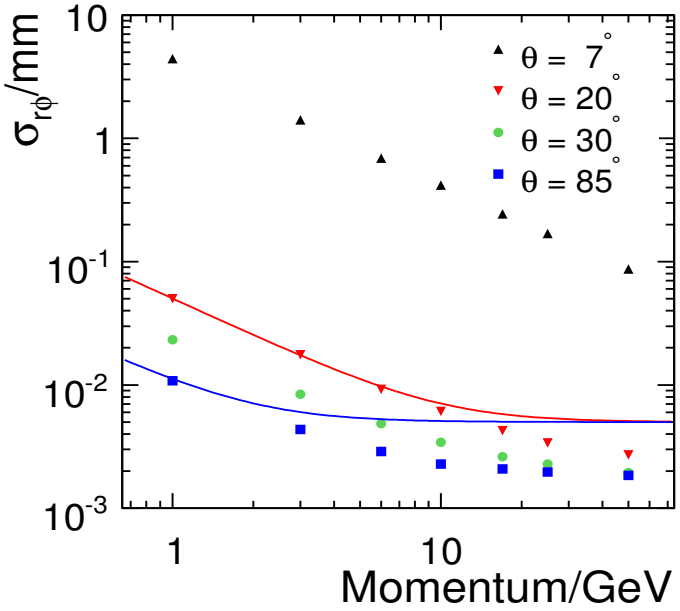
Stand-alone track finding in both Inner Silicon Trackers and TPC

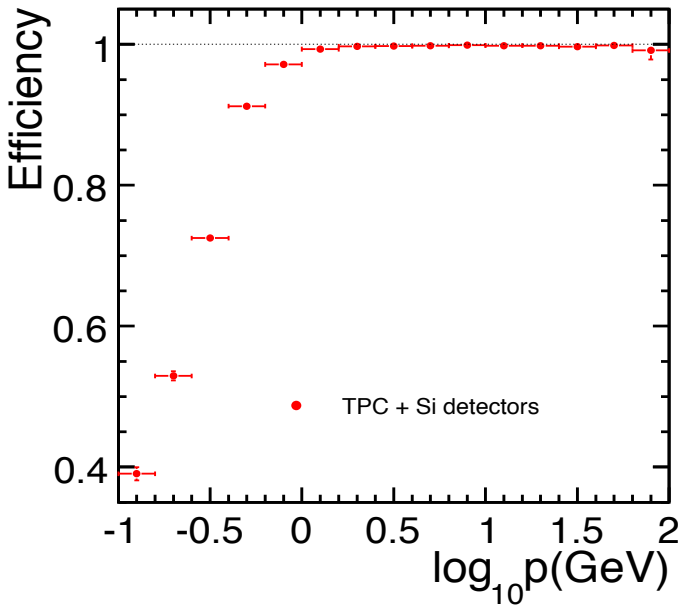
Kalman Filter Track Fitting









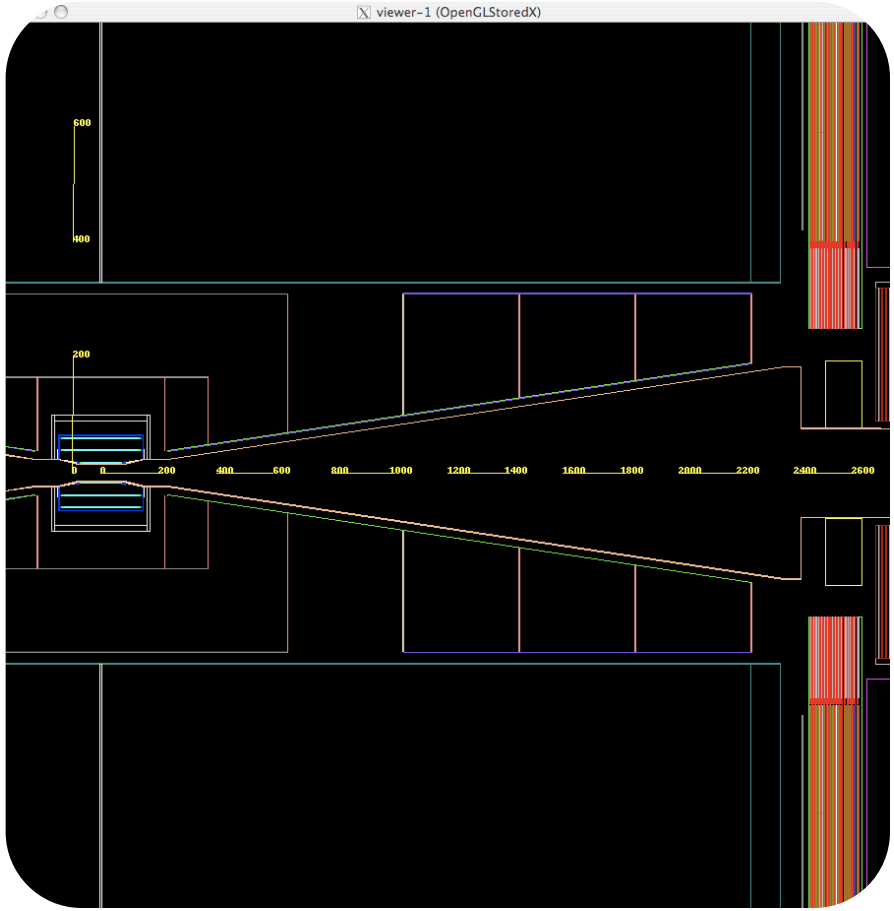


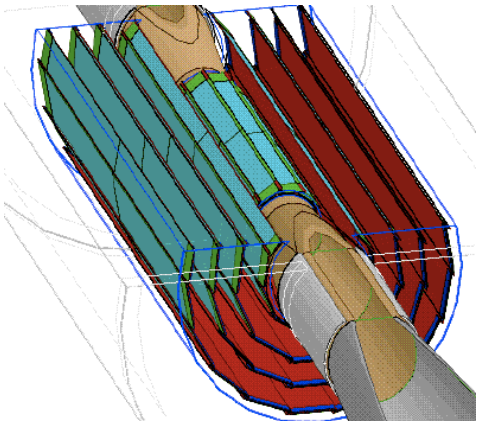
ILD LoI

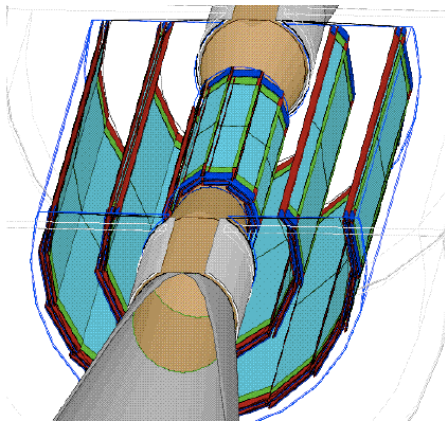
ILD Tracker Simulation

DBD calls for more realistic detector descriptions

LoI Simulation uses a mixture of realistic and simplified detector descriptions



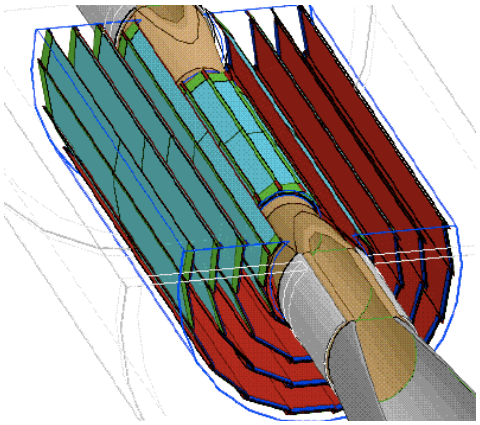


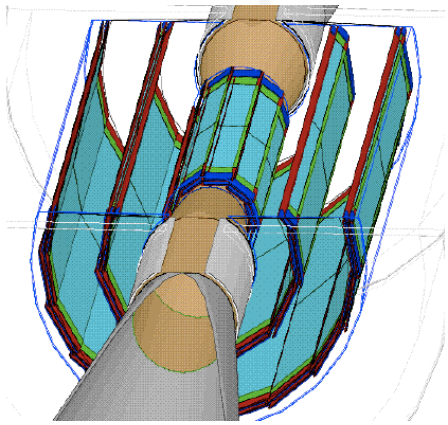


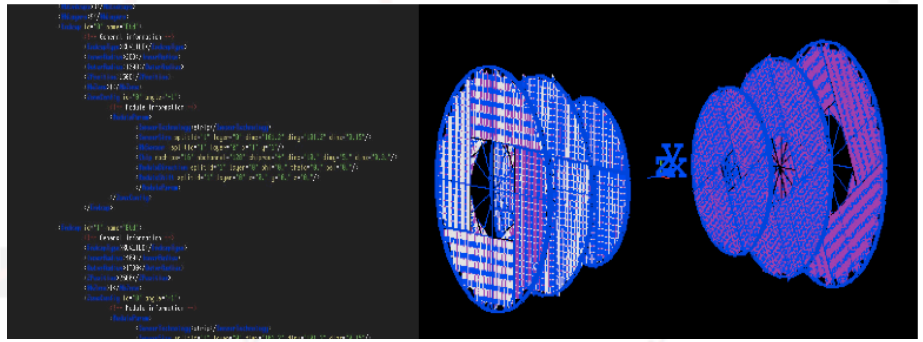
ILD Tracker Simulation

DBD calls for more realistic detector descriptions

Silicon Trackers have now been revised to bring them up to the same level of realism

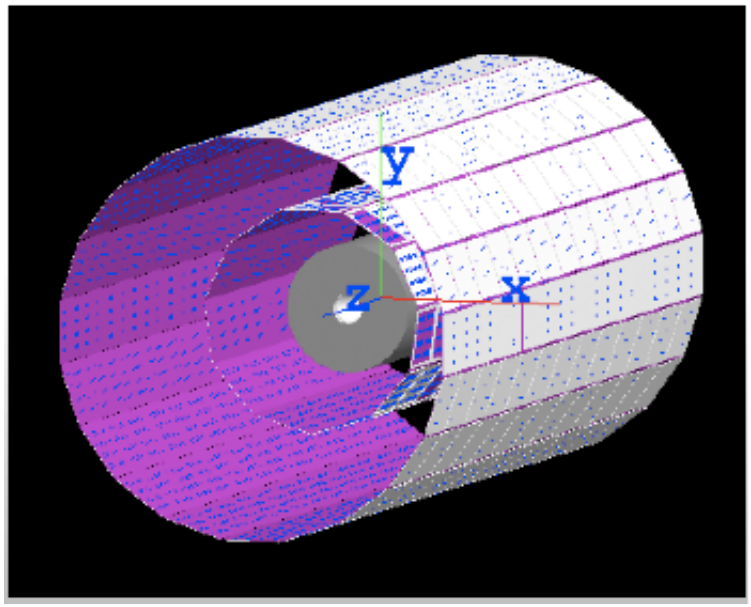


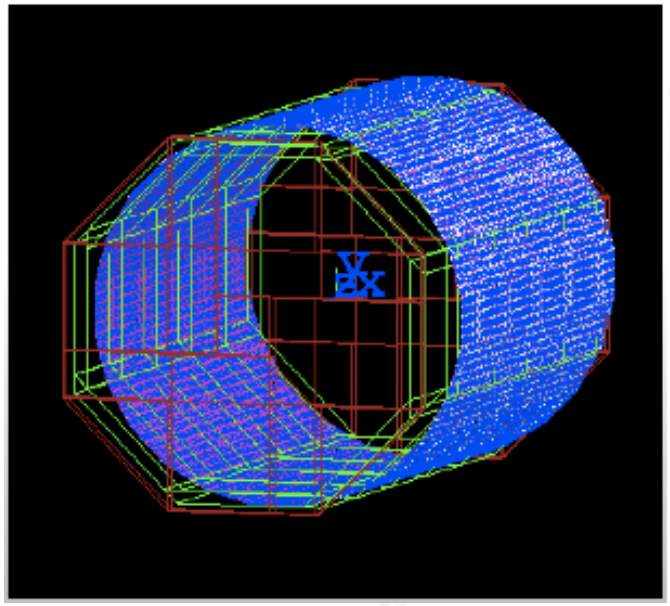


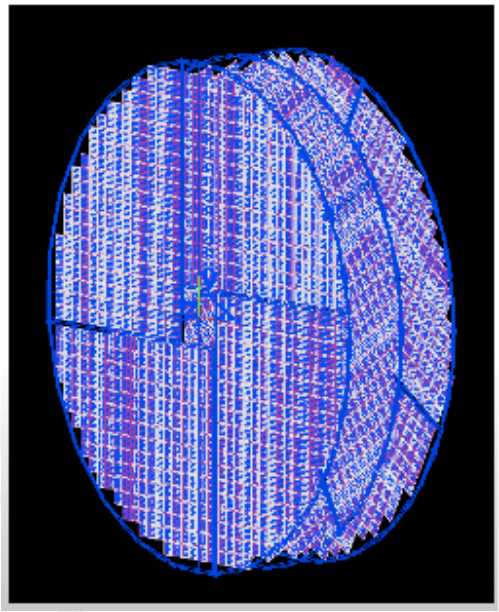


ILD Tracker Simulation

SIT, SET & ETD



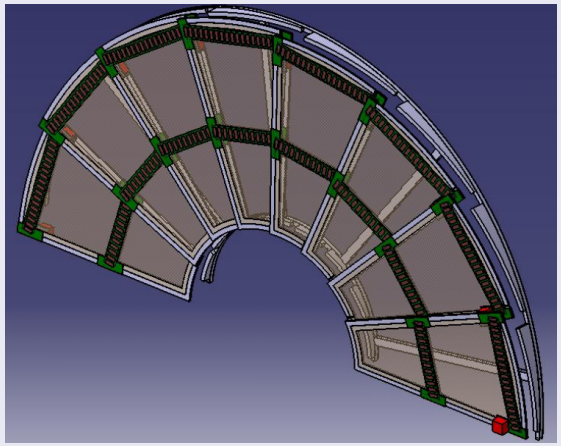


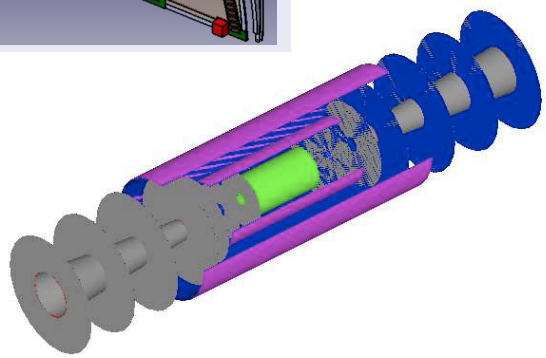


Alexandre Charpy & Konstantin Androsov

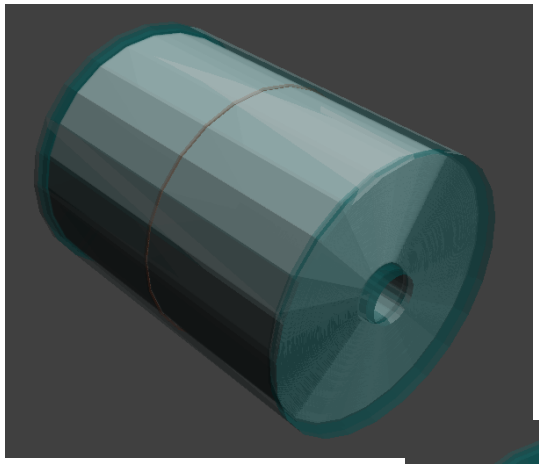
ILD Tracker Simulation

FTD and TPC

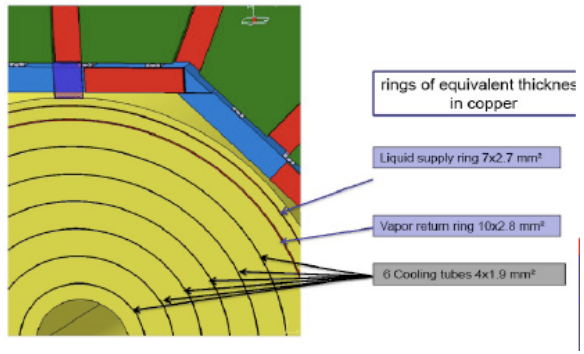


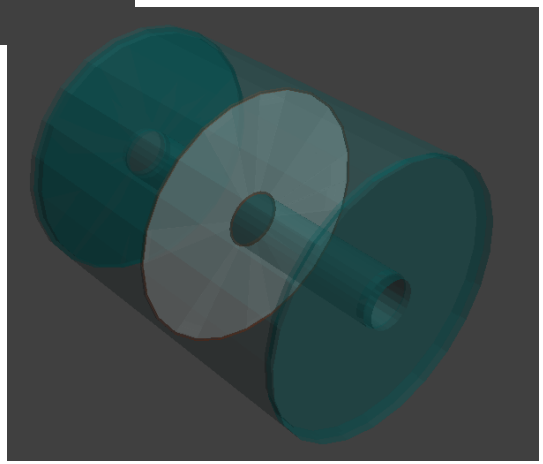


Jordi Durate Campderros



S. Aplin & Gabriel Musat

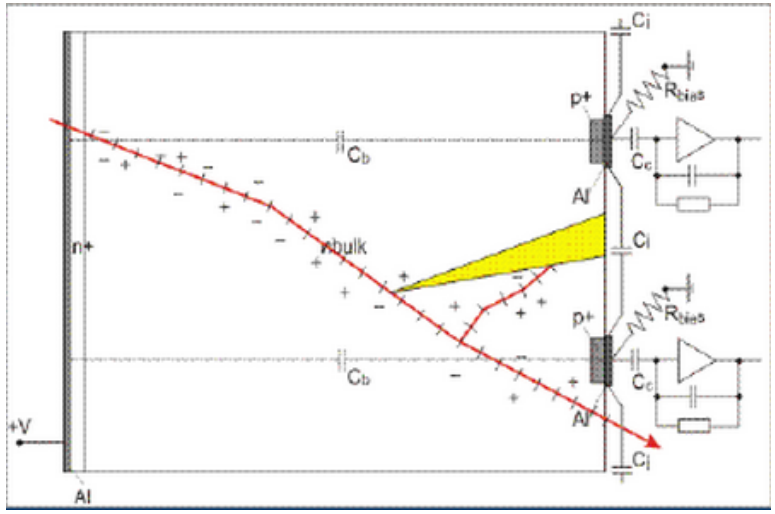




Detector models now increasingly including material from services

Digitisation

- TPC uses long established parameterised smearing.
- VXD has option of parameterised smearing or detailed digitisation in the case of FPCCD (Daisuke Kamai).
- FTD, SIT,SET and ETD adopting the dedicated digitisation and clustering from Zbynek Drásal (Charles University Prague). Code currently in use for the digitisation of the SVD subdetector in Belle II.
- Digitisation of micro-strips:
  - adapted to barrel and forward geometries
  - drift in electric field
  - diffusion due to multiple collisions
  - Lorentz shift in magnetic field
  - mutual micro-strip crosstalks
  - electronics noise



Digitisation

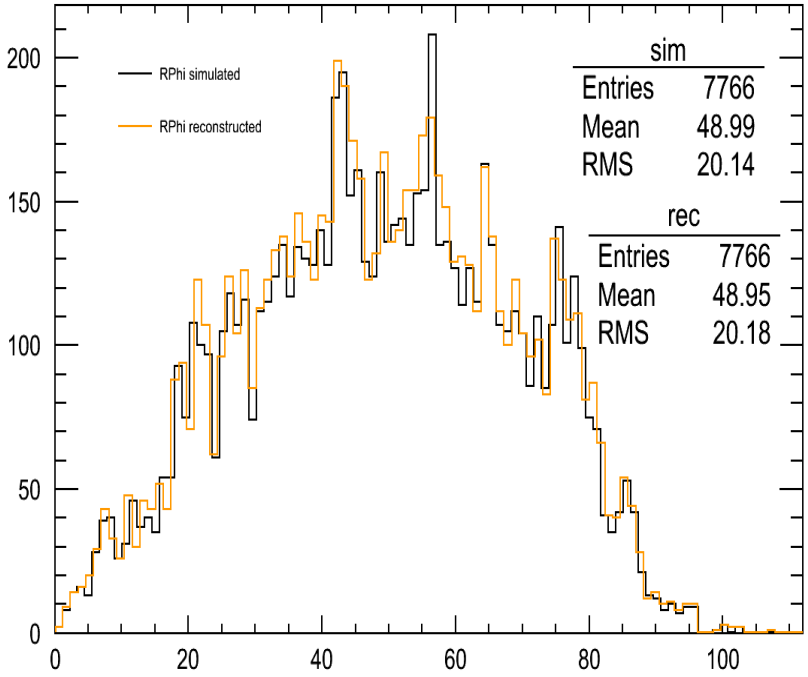
- Clustering: cluster finding algorithm based on COG method (cluster size < 3) or on head-tail analog method (cluster-size > 2). Transform electric pulses into real hits.
- Integrated in Marlin framework: two main processors:
- SiStripDigi:
  - LCIO input collection: SimTrackerHit
  - LCIO output collection: TrackerPulse
- SiStripClus:
  - LCIO input collection: TrackerPulse (output of SiStripDigi processor)
  - LCIO output collection: TrackerHit
- SiLCDigi: for SIT and SET currently working on the clustering, GEANT4 Model already in place. ETD to follow. (Konstantin Androsov & Alexandre Charpy)

Digitisation

Jordi Durate Campderros

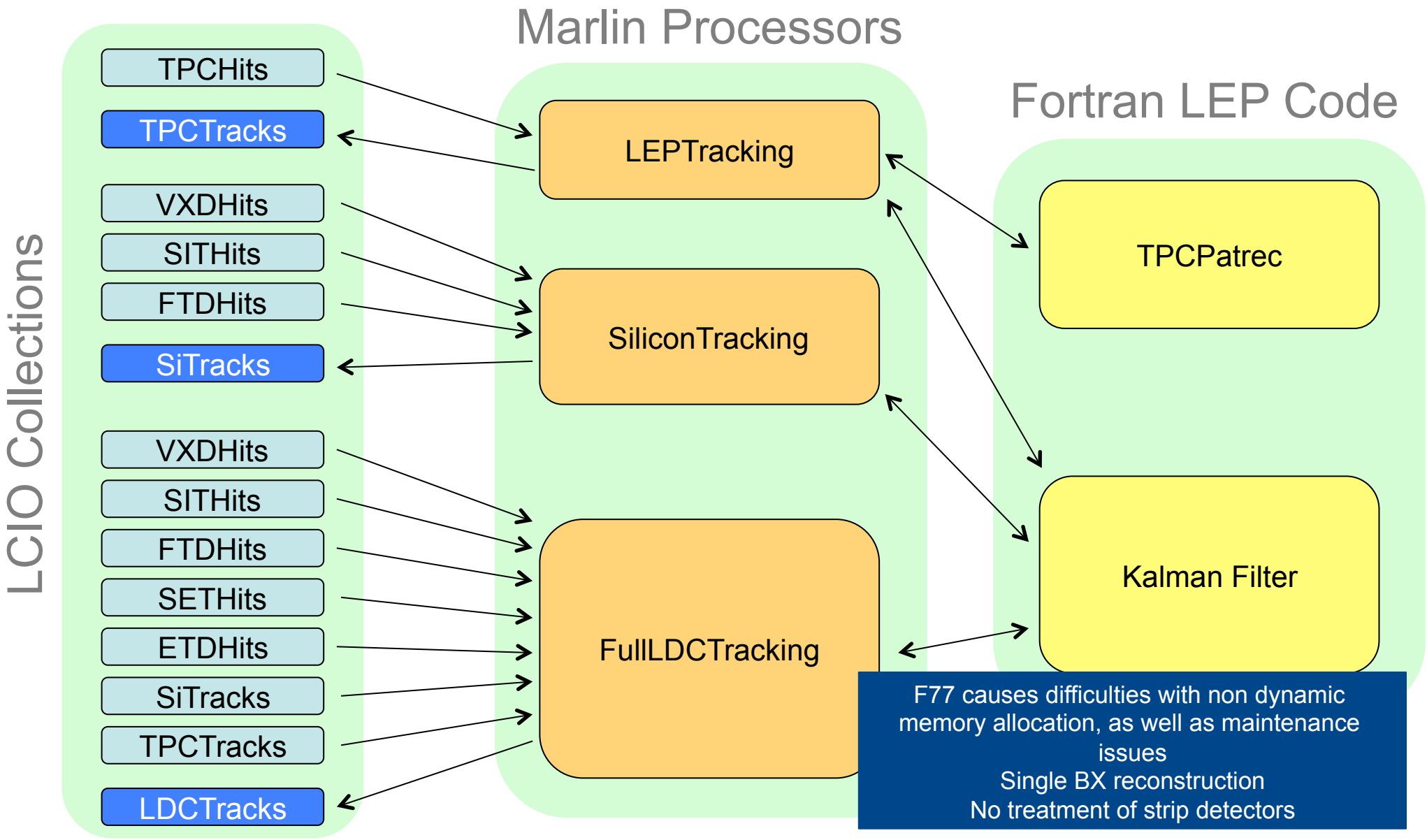
- FTD: Geometrical interface decoupled – FTD geometrical description added and fully operative
- Code already provides Digitising and Clustering
  - Caveats:
  - First version: checking the behavior with the FTD disks
  - Using Single Side Sensors covering both faces of the support petal to obtain RPhi coordinates
  - Recently discussed the micro-strips orientation on the petal: stereo-angle design

Plots VERY PRELIMINARY:
demonstrate technical implementation



Track Reconstruction

Track Reconstruction used for the LOI



Track Reconstruction used for the LOI

Nevertheless the reconstruction survived pretty much everything CLIC could throw at it …

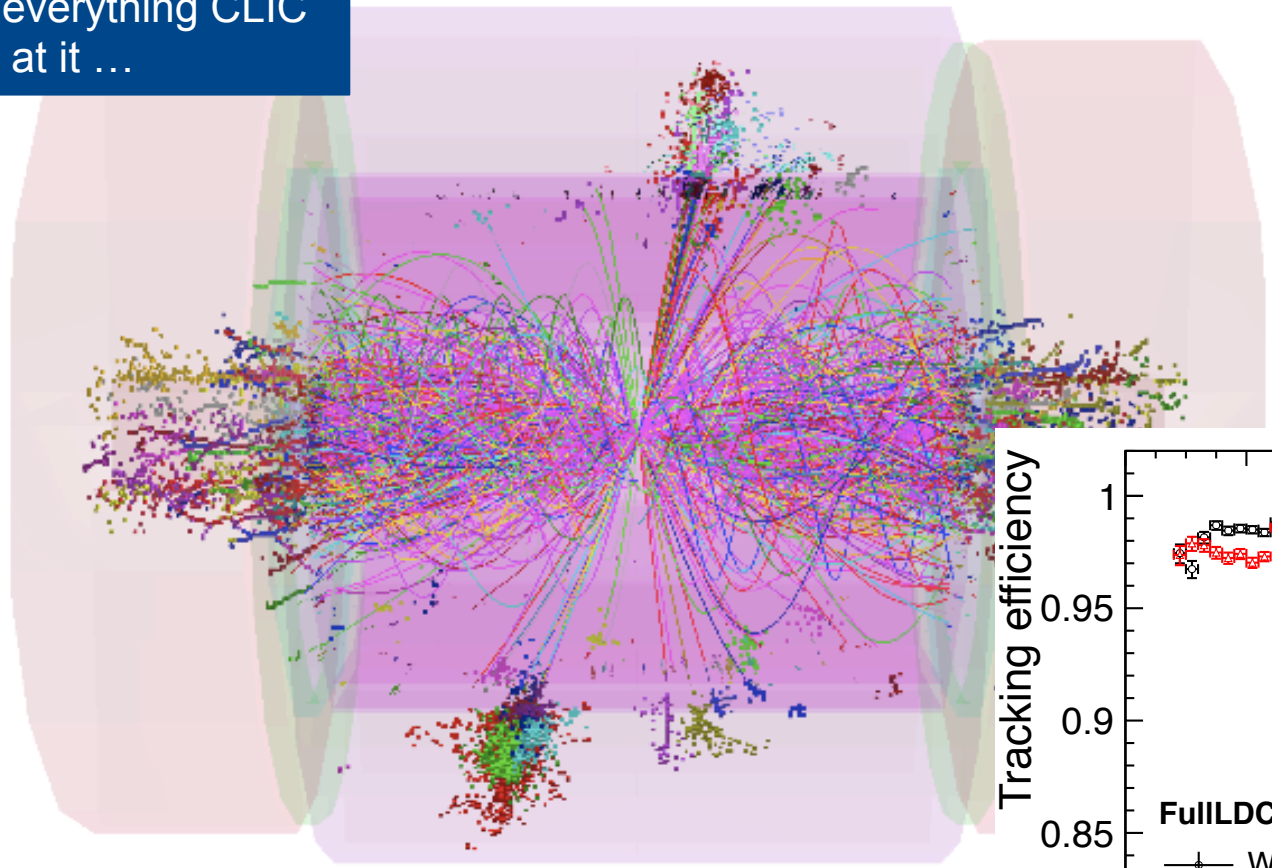
Marlin Processors

LCIO Collections

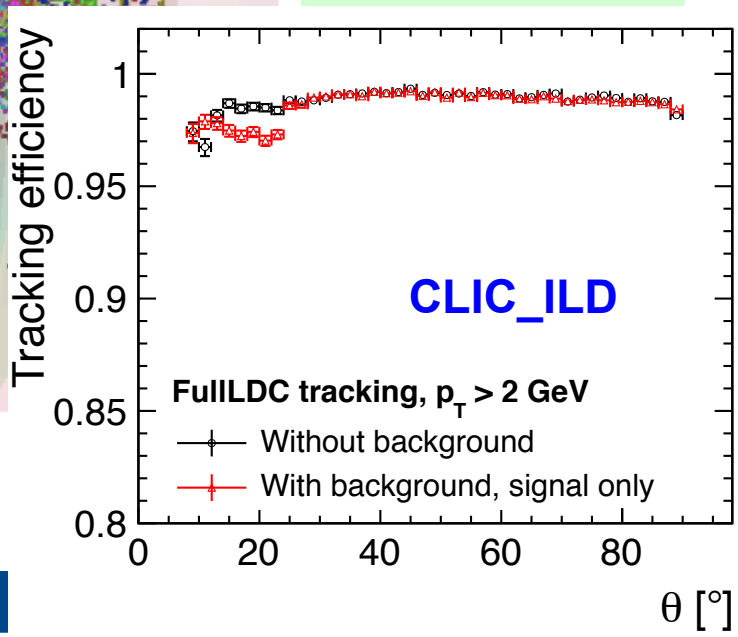
VXDHit
SITHit
FTDHit
SiTrack
VXDHit
SITHit
FTDHit
SETHit
ETDHit
SiTrack
TPCTrac
LDCTracks

LEP Code

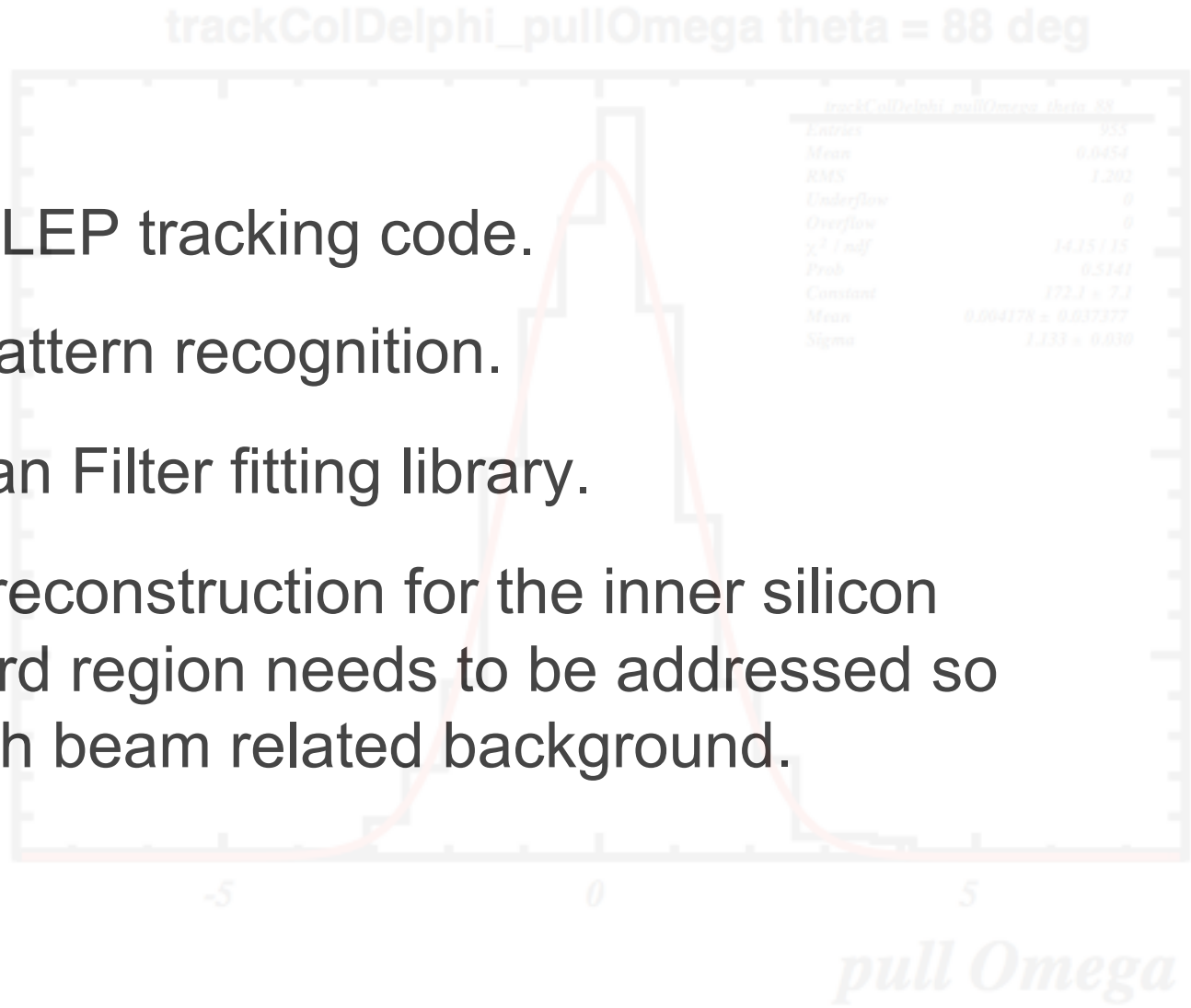
PCPatrec



1.4 TeV of background !



Tracking Code rewrite for the DBD



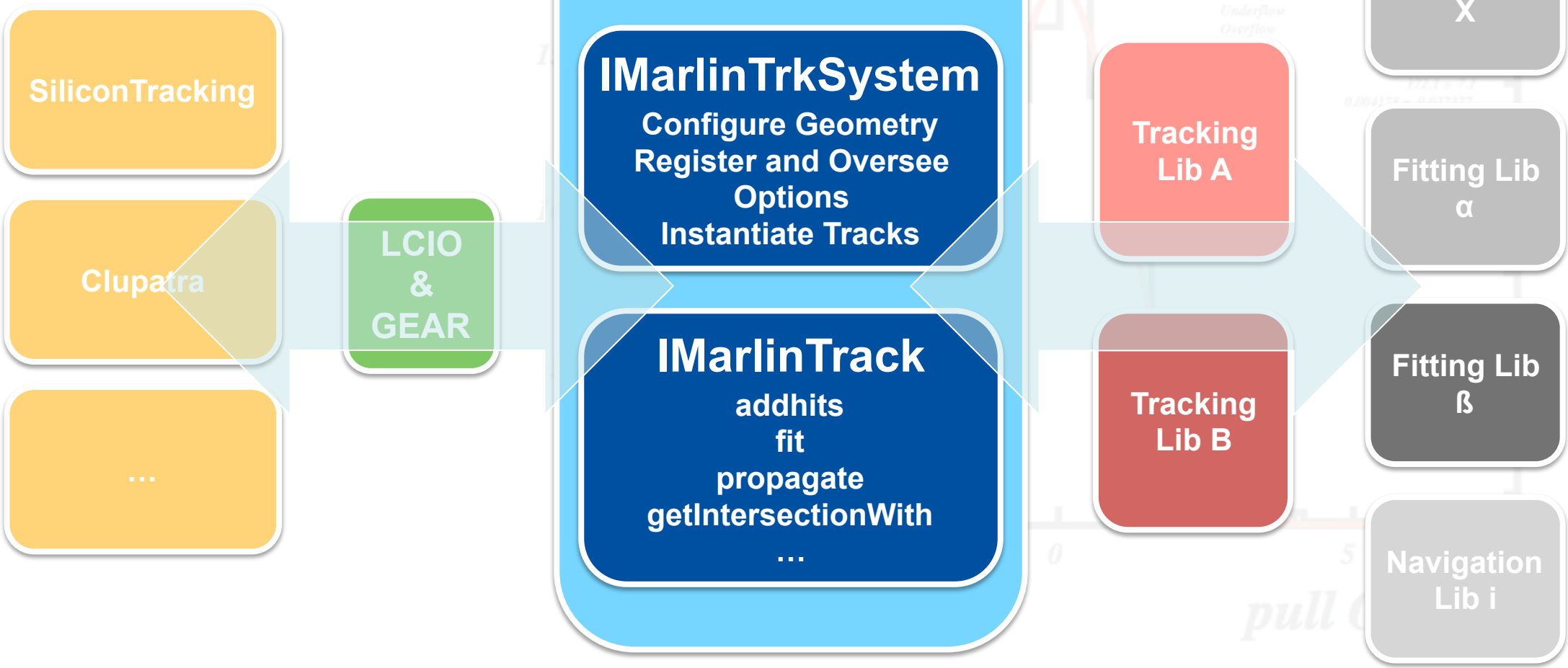
- Leave behind F77 LEP tracking code.
- Rewrite the TPC pattern recognition.
- Use KalTest Kalman Filter fitting library.
- Stand alone track reconstruction for the inner silicon trackers and forward region needs to be addressed so that it can cope with beam related background.

IMarlinTrack and IMarlinTrkSystem

- **IMarlinTrack**
  - interface class to provide access to track fitting and track parameter manipulation in **Marlin**
  - uses **LCIO** for both input and output

- **IMarlinTrkSystem**
  - responsible for managing the necessary infrastructure such as geometry for the track fitting, making use of **LCIO** and **GEAR**
  - controlling the configuration of the fitting package

IMarlinTrack and IMarlinTrkSystem



IMarlinTrack and IMarlinTrkSystem

- **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 …

IMarlinTrack and IMarlinTrkSystem

```
//** 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 ;
```

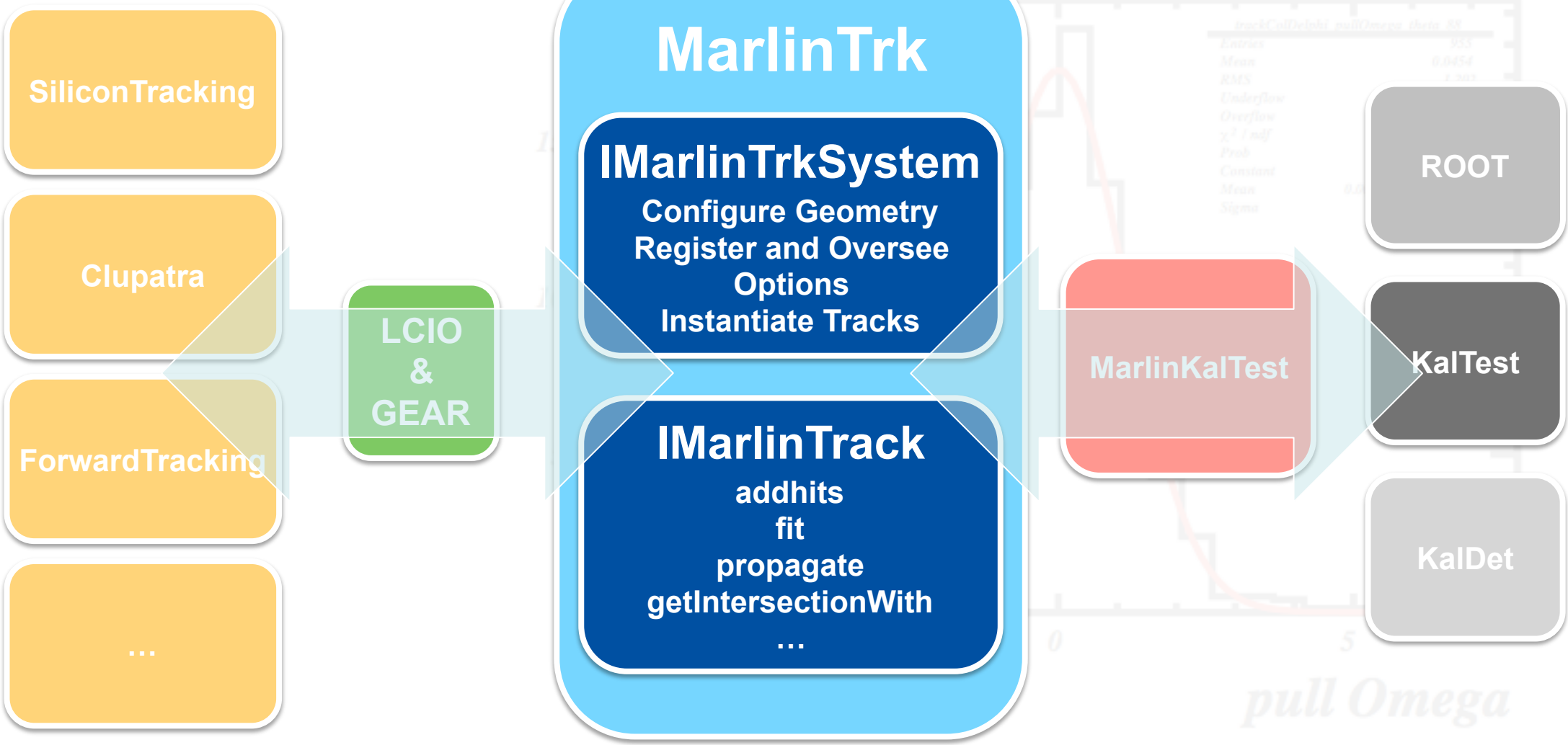
IMarlinTrack and IMarlinTrkSystem

- New developments in **LCIO 2.0** provided to support development:
  - This release contains the new **TrackState** class, as well as the new **TrackerHit** classes
  - The two new **TrackerHit** classes are: **TrackerHitZCylinder** and **TrackerHitPlane**, which especially help for dealing with strip detectors, as well as avoiding kludging measurements into 3D space-points.
- Standardised use of **CellID0** allows efficient association of hits with detector elements.
- Extending the detector interfaces in **GEAR** is a real plus concerning the stability and robustness of the tracking code, for example the new FTD layer layout.

Marlin and KalTest

- MarlinTrk and MarlinTrkProcessors packages provided in the **MarlinReco** svn repository:
  - **MarlinTrk** – this contains the interface classes as well as the implementation of the interfaces, presently only for **KalTest**
  - **MarlinTrkProcessors** – Contains example **Processors** which use the functionality provided in **MarlinTrk.** Presently an example Refitter processor is provided as well as simple planar digitiser, demonstrating how to use the new **TrackerHitPlane** class, as well as the use of **CellID0** for the track reconstruction.

MarlinTrk KalTest Implementation



KalTest

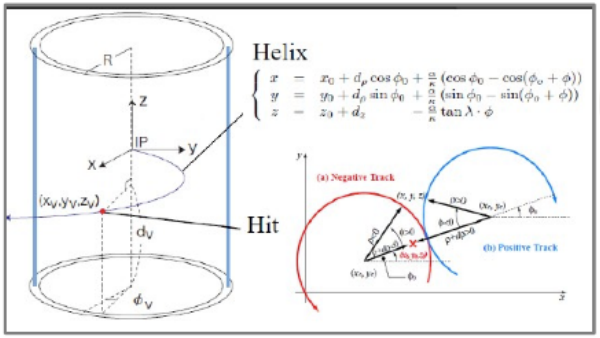
Kalman Filter fitting library (Keisuke Fuji et al)

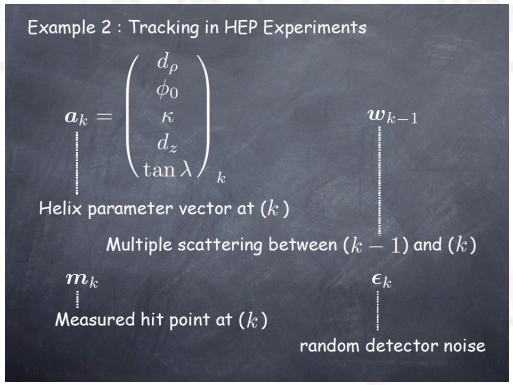
Based on Root

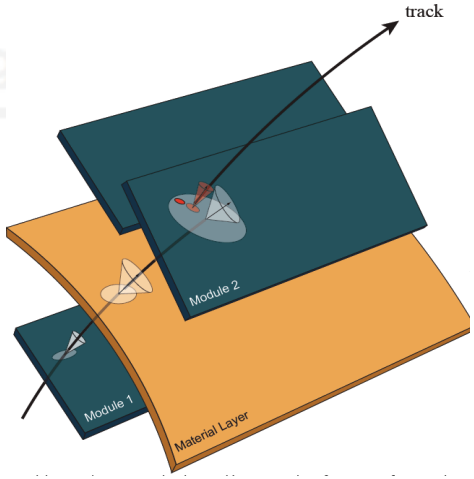
Structured in sub-libraries

- geomlib -- geometry
- kallib -- Kalman filter
- kaltracklib -- Kalman tracker
- utils -- utilities

Built into one libKalTest.so







User needs to define their detector classes (**KalDet**)

- TVMeasLayer: meas. layer, coord. to track state transformation
- TVDetector: position of measurement layers and material properties

•Since ALCPG treatment of bounded and rotated planes have been added to **KalTest**, by Daisuke Kamai.

KalDet

- **KalDet** has been augmented with geometry and measurement classes needed to describe the sub-detectors in ILD for use in **KalTest**
  - base classes have been defined for both measurement layers and tracker hits.
  - these are then used to provide implementations of planar (**VXD**, **SIT**, etc.) and cylindrical specialisations (**TPC**).

- These classes are then used to provide concrete implementations of the sub-detectors both in terms of detector layout via **GEAR,** and the necessary conversion of position measurements provided in the form of **LCIO TrackerHits**

Propagators

- Track propagation functions previously buried deep inside the F77 tracking code, not available for lcio track class.
- New set of track propagators added to MarlinTrk:

// Propagate track to a new reference point

IMPL::TrackImpl* PropagateLCIOToNewRef(EVENT::Track* trk, double xref, double yref, double zref) ;

// Propagate track to a new reference point taken as its crossing point with a cylinder of infinite length centered at x0,y0, parallel to the z axis.

IMPL::TrackImpl* PropagateLCIOToCylinder(EVENT::Track* trk, float r, float x0, float y0, int direction=0, double epsilon=1.0e-8) ;

// Propagate track to a new reference point taken as its crossing point with an infinite plane located at z, perpendicular to the z axis

IMPL::TrackImpl* PropagateLCIOToZPlane(EVENT::Track* trk, float z) ;

// Propagate track to a new reference point taken as its crossing point with a plane parallel to the z axis, containing points x1,x2 and y1,y2.

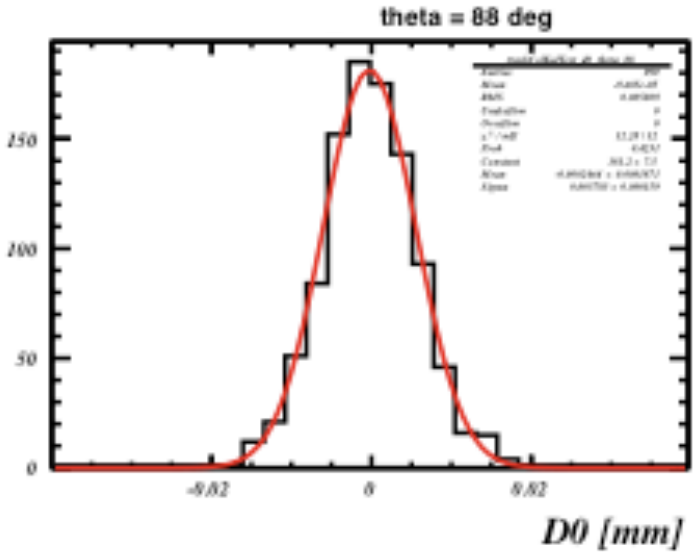
IMPL::TrackImpl* PropagateLCIOToPlaneParralelToZ(EVENT::Track* trk, float x1, float y1, float x2, float y2, int direction=0, double epsilon=1.0e-8) ;

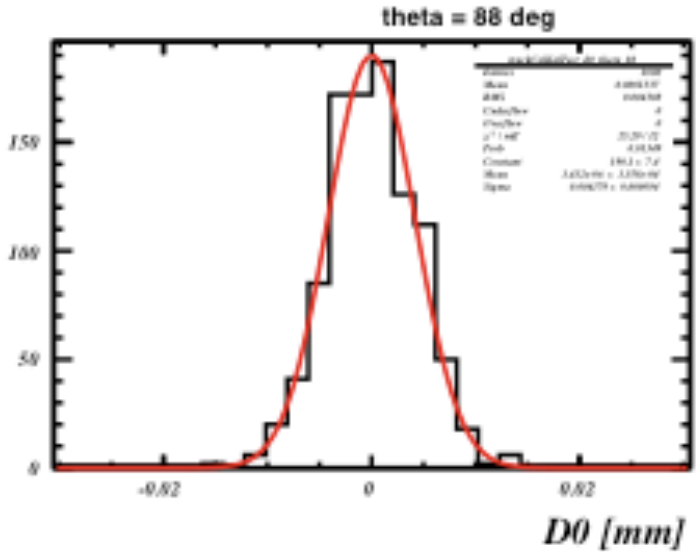
LCIO Tracks can now be propagated to an arbitrary reference point with Cov Matrix

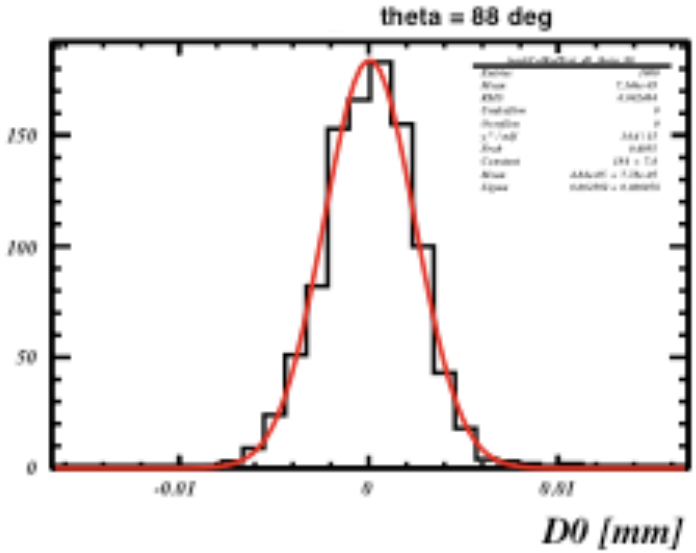
Track Parameter Comparison

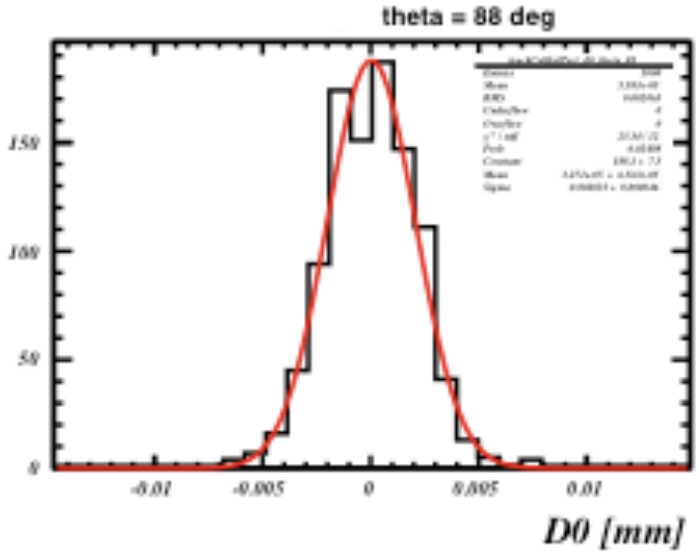
- Refitting Processor used to test development of the Tracking API as well as Track Parameter and error determination in the **MarlinTrk KalTest** implementation
  - Takes **LCIO** Track collection produced by **FullLDCTracking** and refits the associated hits using the Kaltest Kalman Filter.
  - Presently fits are compared only at the IP
- Testing performed using a mock-up of inner detectors in Mokka, not ILD_01
  - Comparison made with Track Parameters and errors determined by F77 LEP fitting code using single muons at p = 3, 6, 40, 100 GeV and theta = 88, 40, 32 degrees

Track Parameter Comparison

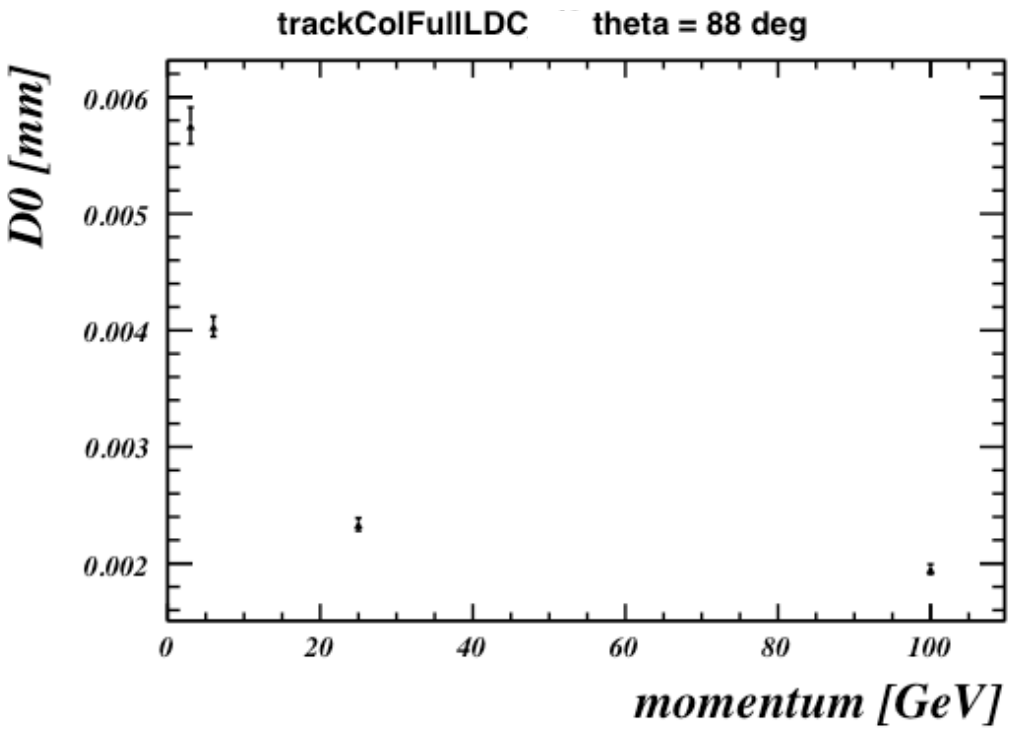


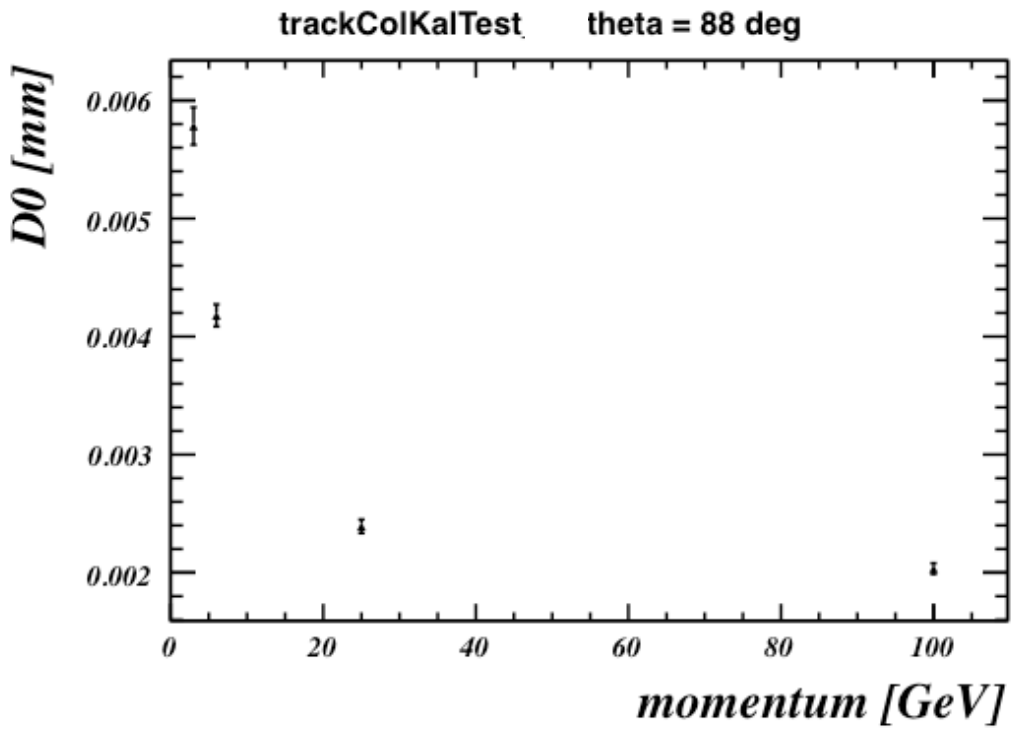




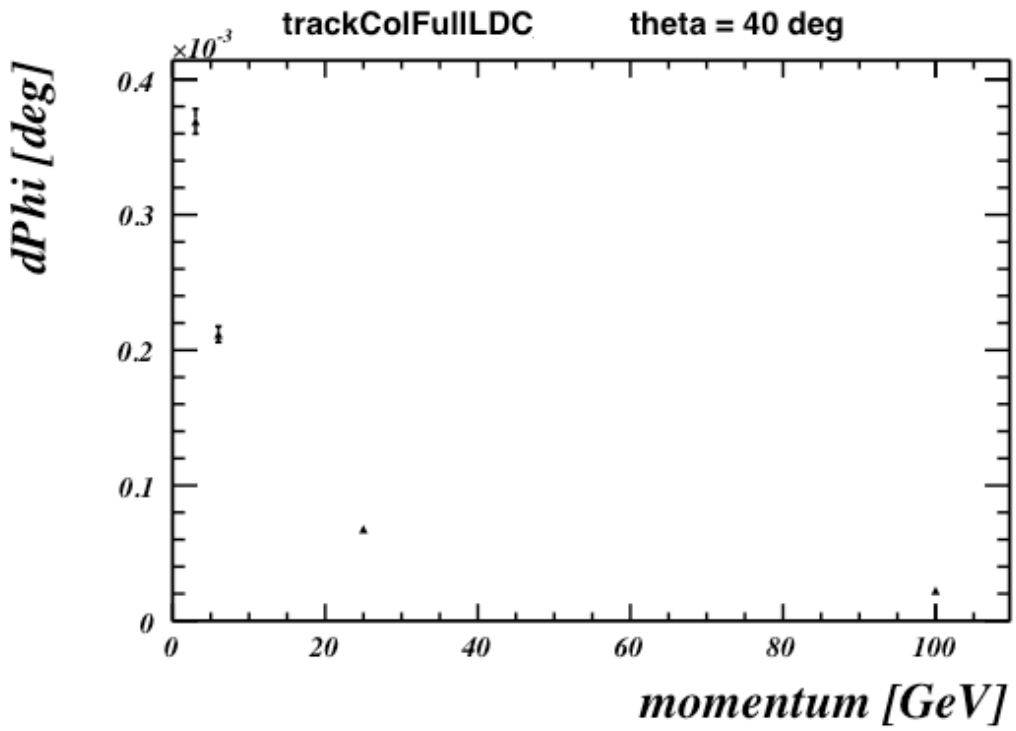


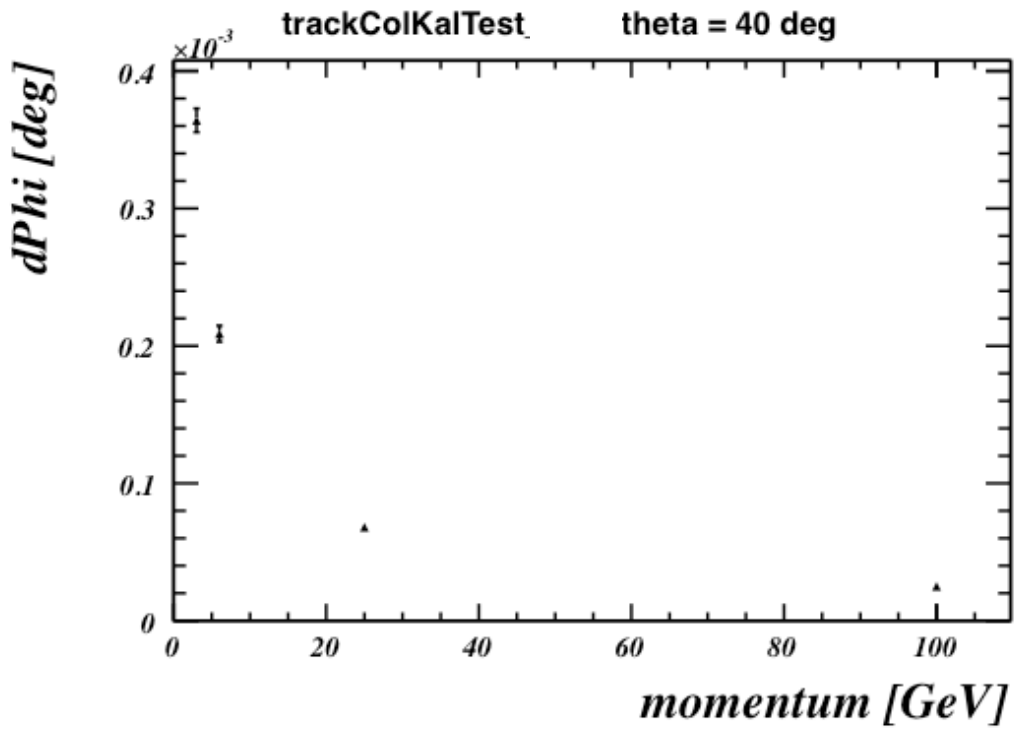
Track Parameter Comparison



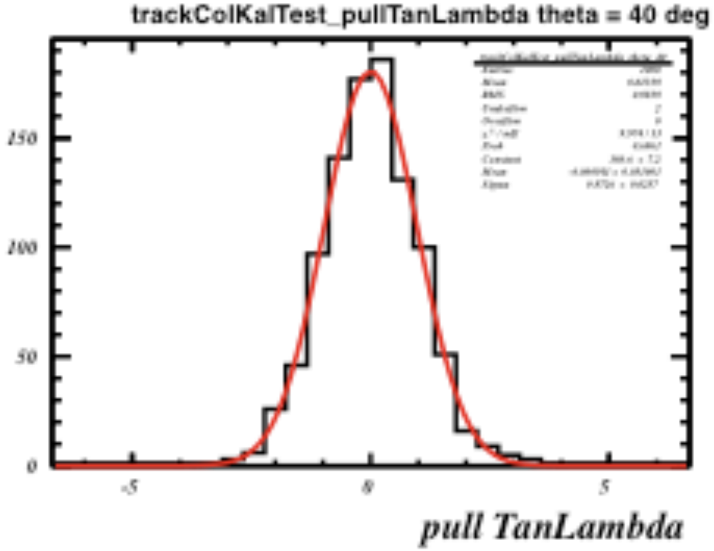


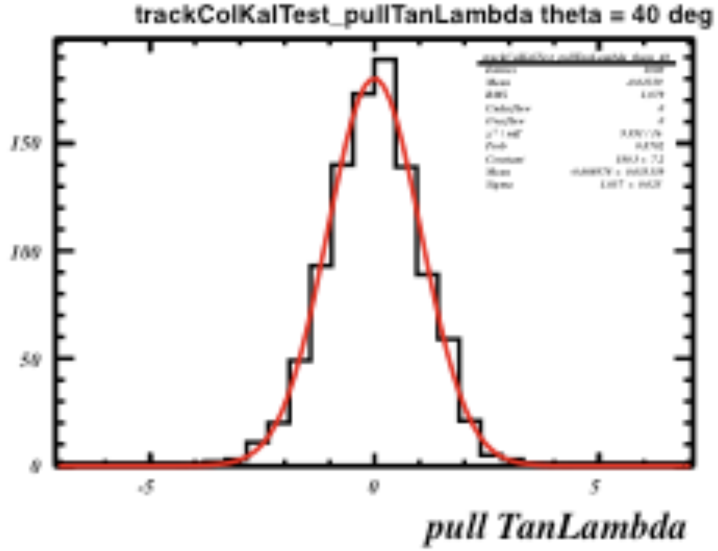
Track Parameter Comparison

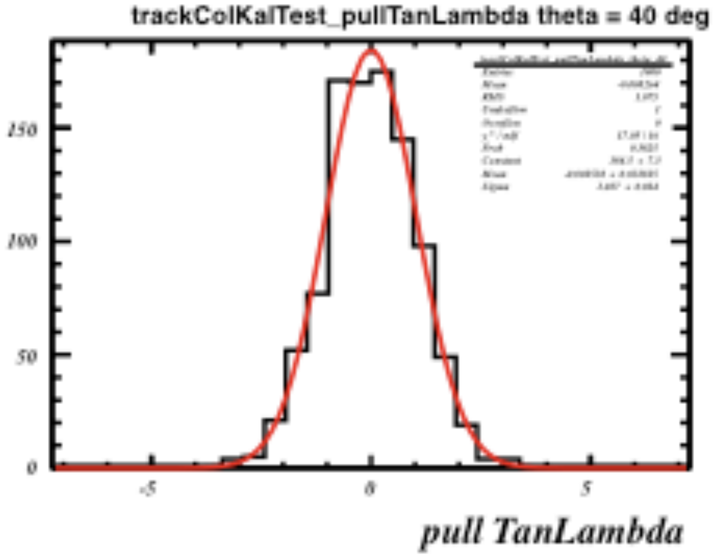


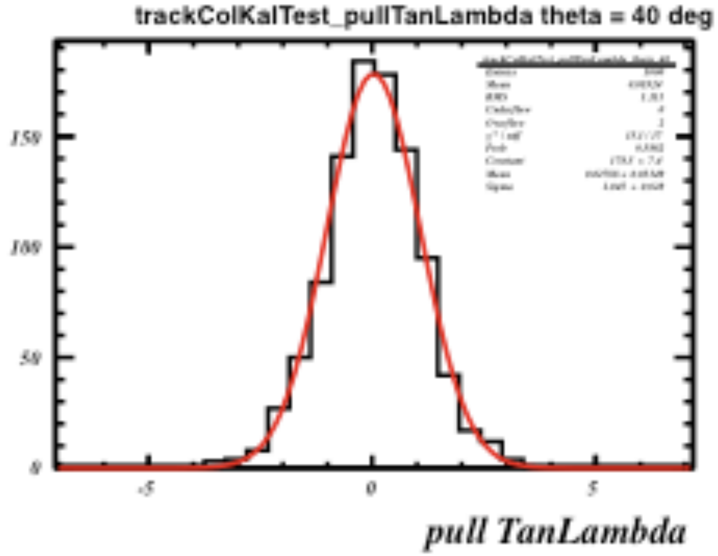


Track Parameter Pull Distributions

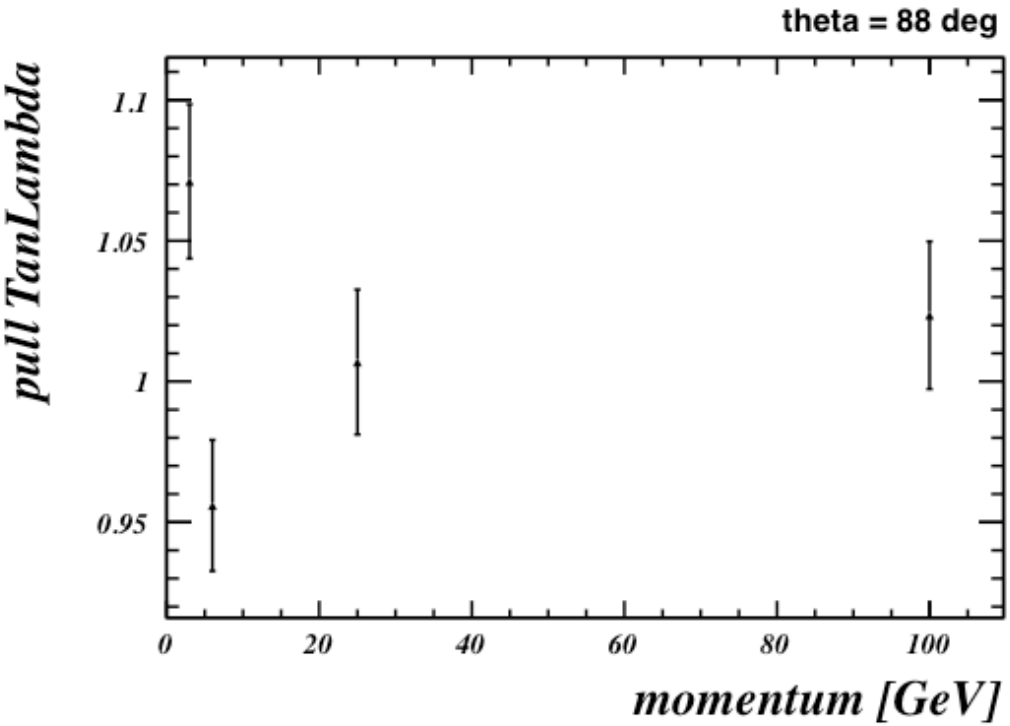


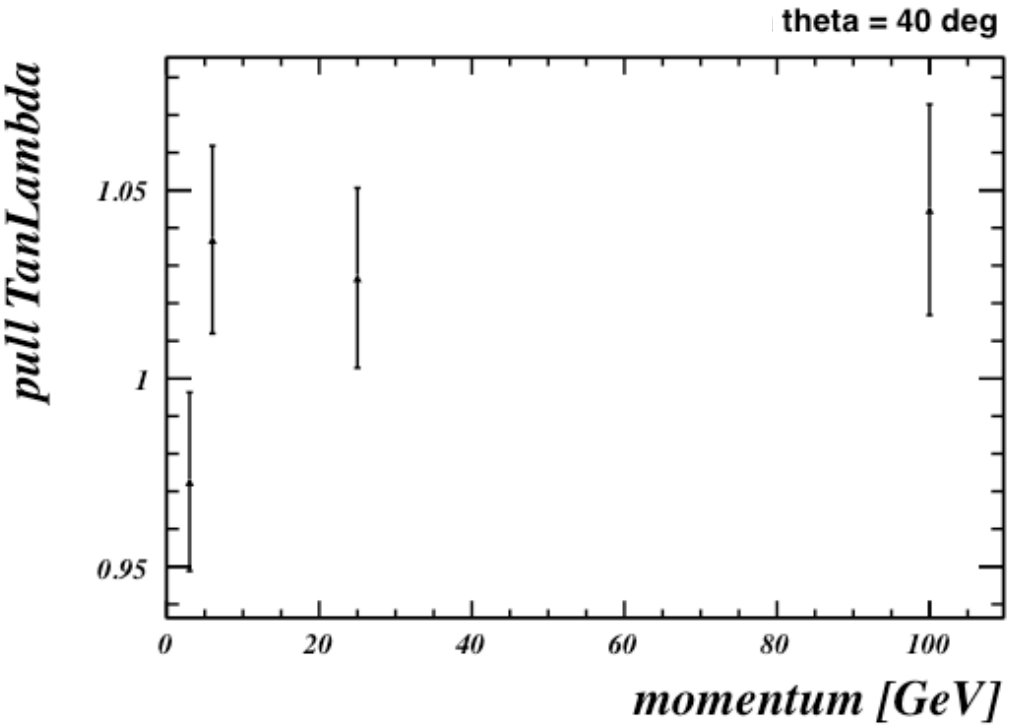




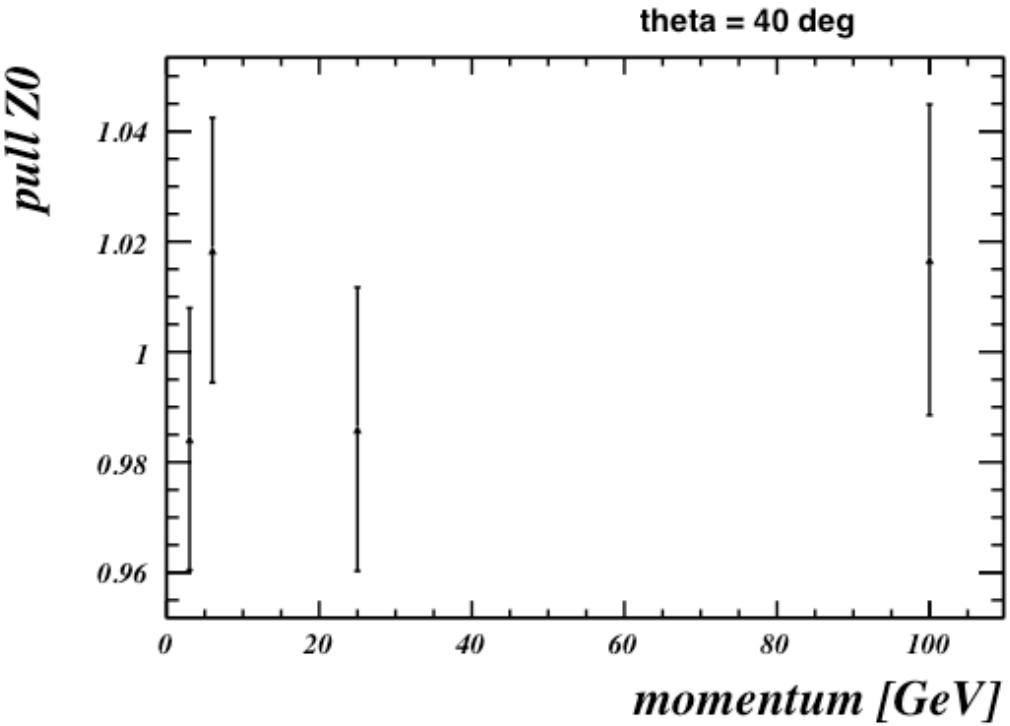


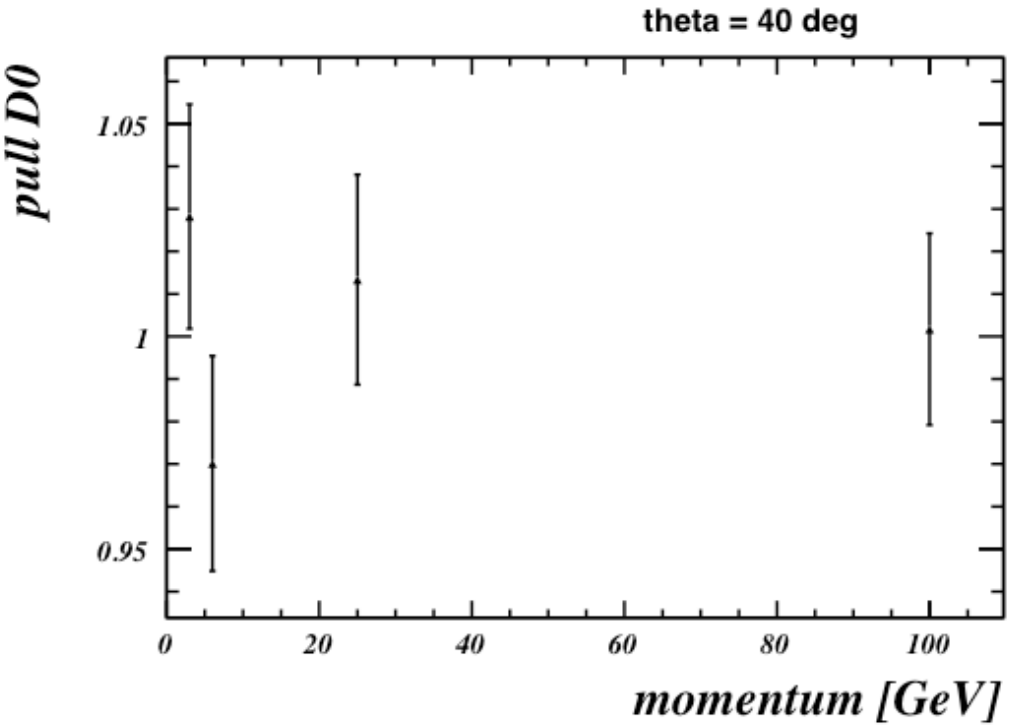
Track Parameter Pull Distributions



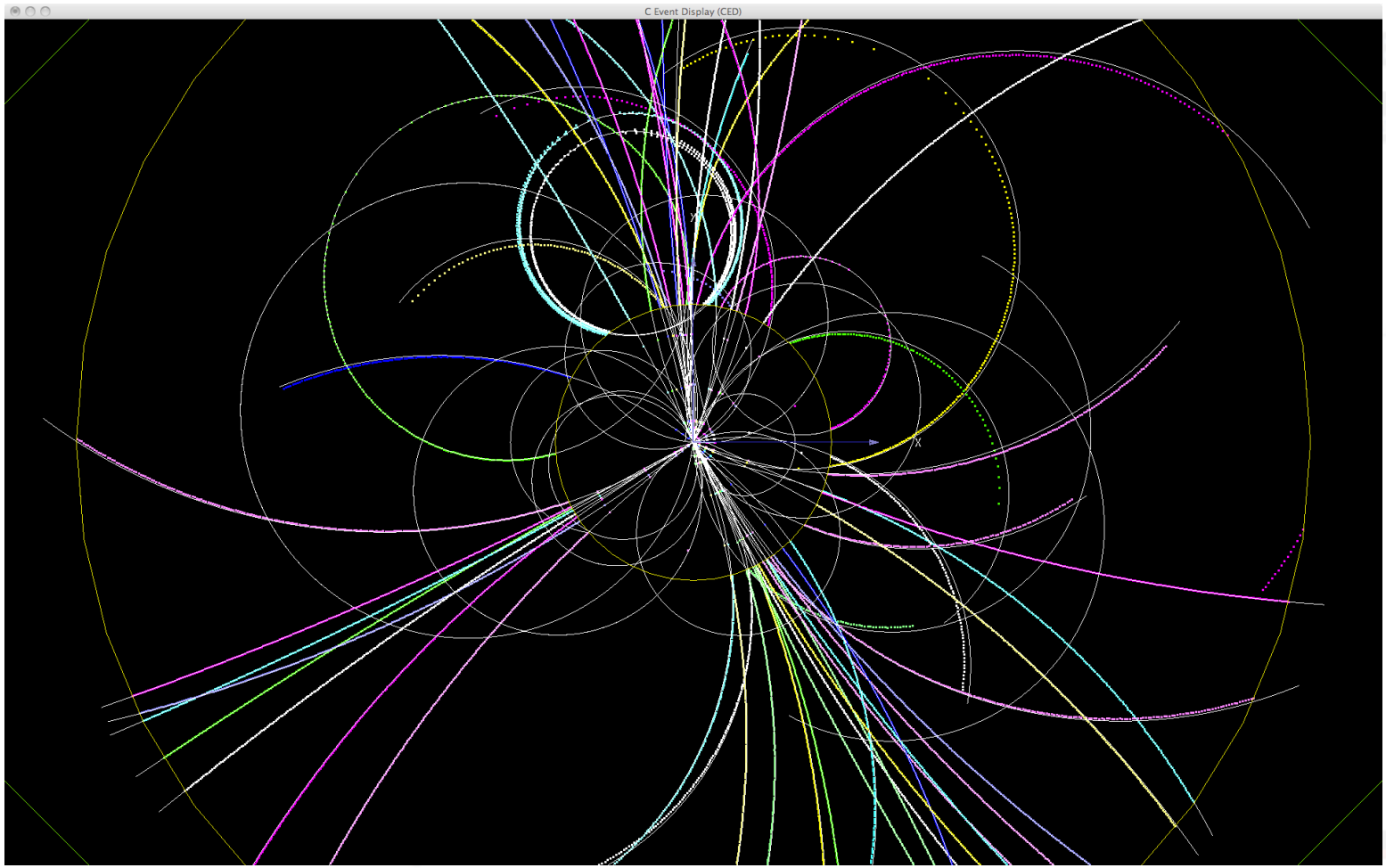


Track Parameter Pull Distributions



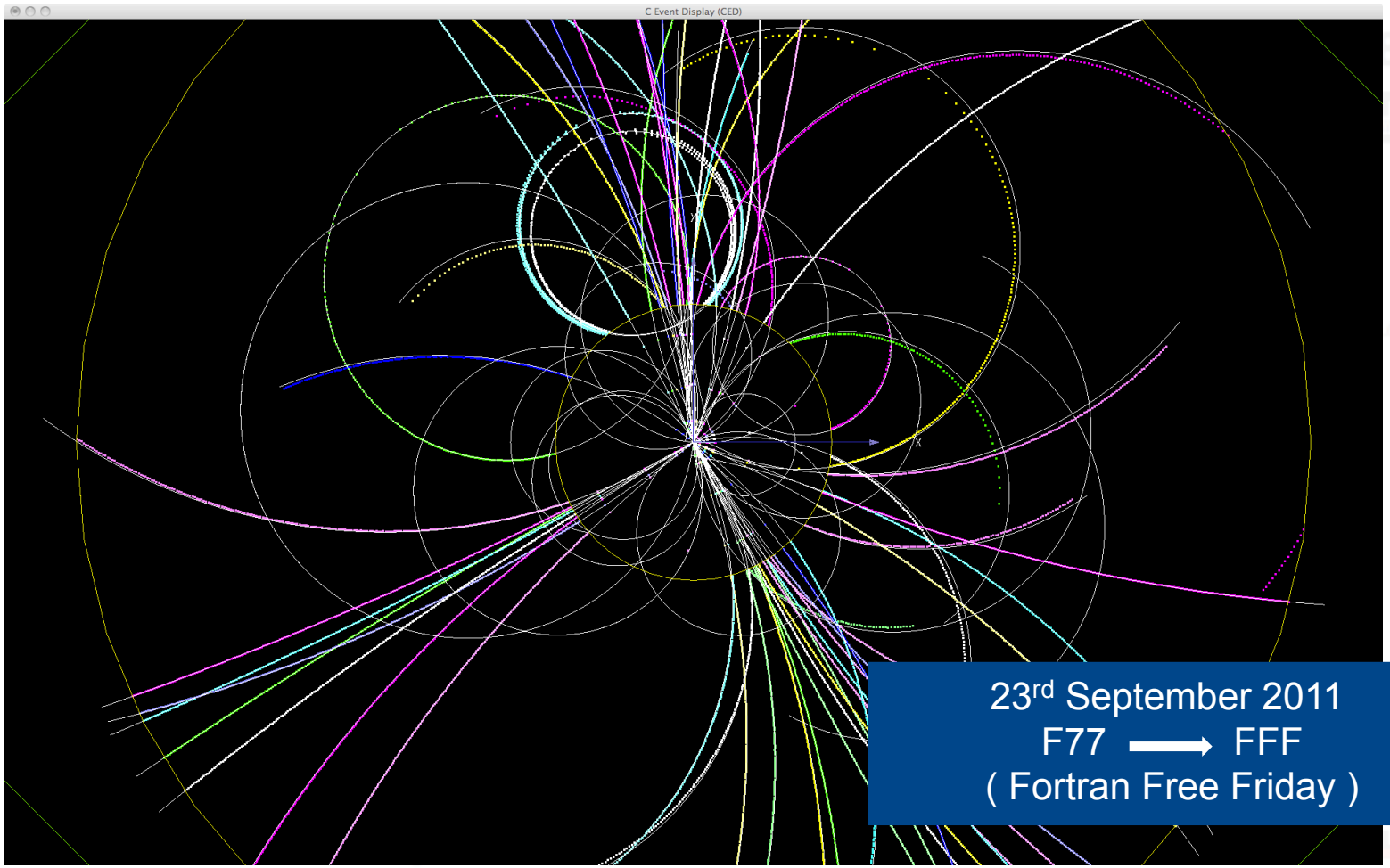


In the mean time …



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

In the mean time …



23rd September 2011
F77 ⟶ FFF
(Fortran Free Friday)

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

Summary

- Detector Models for the DBD well consolidated in Mokka.
- Digitisation development work progressing well, first versions available.
- Released version of **MarlinTrk** and **MarlinTrkProcessors** in svn:
  - https://svnsrv.desy.de/public/marlinreco/MarlinTrk/tags/v01-00
  - https://svnsrv.desy.de/public/marlinreco/MarlinTrkProcessors/trunk
- Need to press forward with production versions of the code.
- MarlinTrk already adopted by the new Clupatra and ForwardTracking packages, as well as reengineered versions of SiliconTracking and FullLDCTracking.
- Need to focus on incorporating the hits from strips detectors into the pattern recognition as well as push forward with the global track reconstruction.