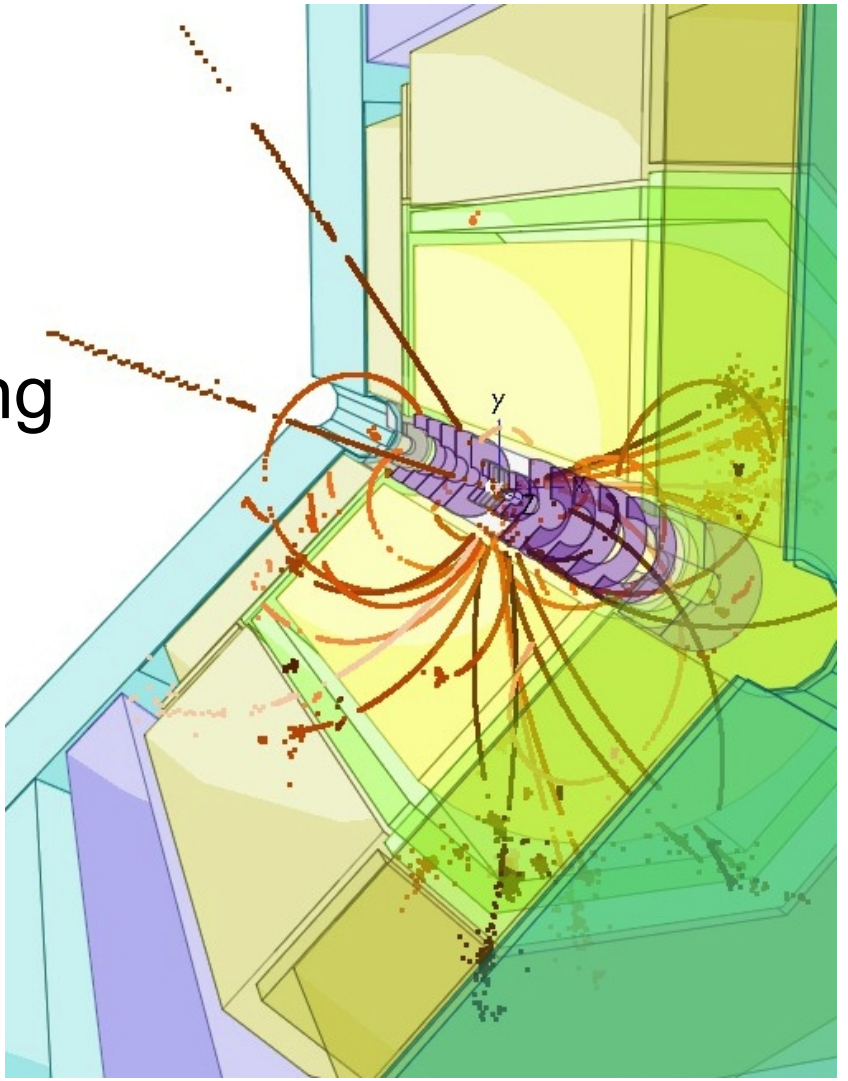
Linear Collider Track Reconstruction Tools

Frank Gaede, DESY
Common Track Reconstruction Software Forum
CERN, 2. Dec. 2015

iLC SOft

Outline

- Introduction
  - ILD tracking and AIDA
- EDM and Geometry for tracking
  - LCIO, DD4hep
- Core tracking tools
  - KalTest, aidaTT, MarlinTrk
- Pattern recognition
- Summary Outlook

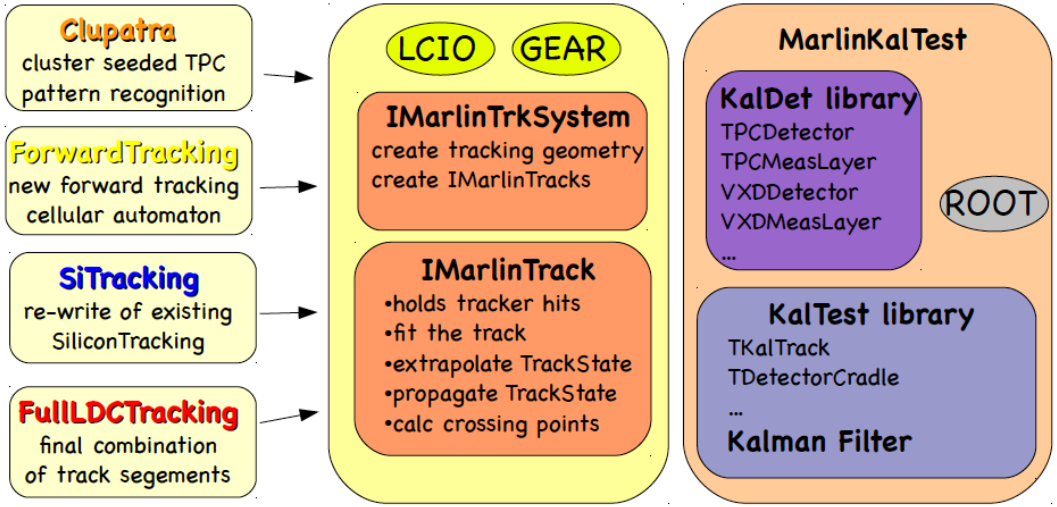


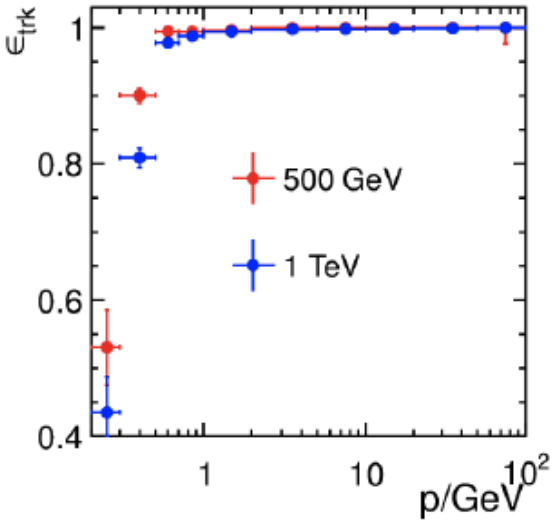
Introduction

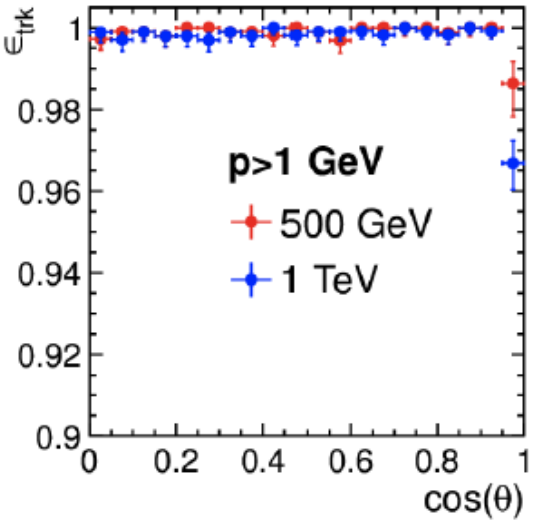
- developed new C++ tracking tools for Linear Colliders
  - to replace old F77 (!) code from LEP
  - successfully used for ILD DB (2011), CLIC CDR (2012)
- partly done in context of AIDA-WP2 project
  - goal: eventually have a generic HEP tracking toolkit that could be shared by all LC detector concept groups (and possibly others)
  - allowing to transparently use different fitting algorithms
  - provide toolkit for pattern recognition
  - have well defined and easy to use interface to detector geometry
- code developed in context of ILD w/ generality in mind

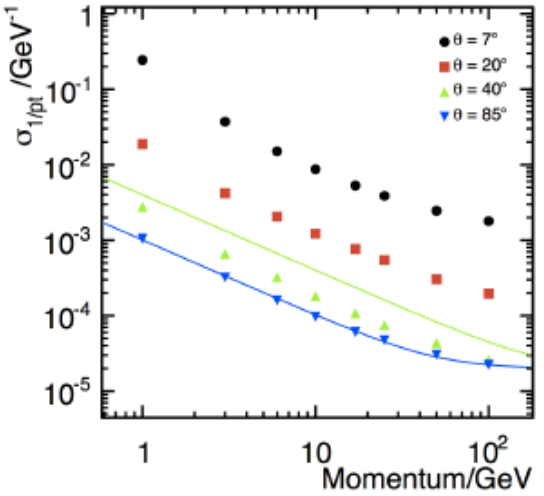
ILD track reconstruction

- independent pattern recognition in TPC, Si, Fwd
- programmed against IMarlinTrk interface
- KalTest Kalman filter
- achieves performance goals for ILC







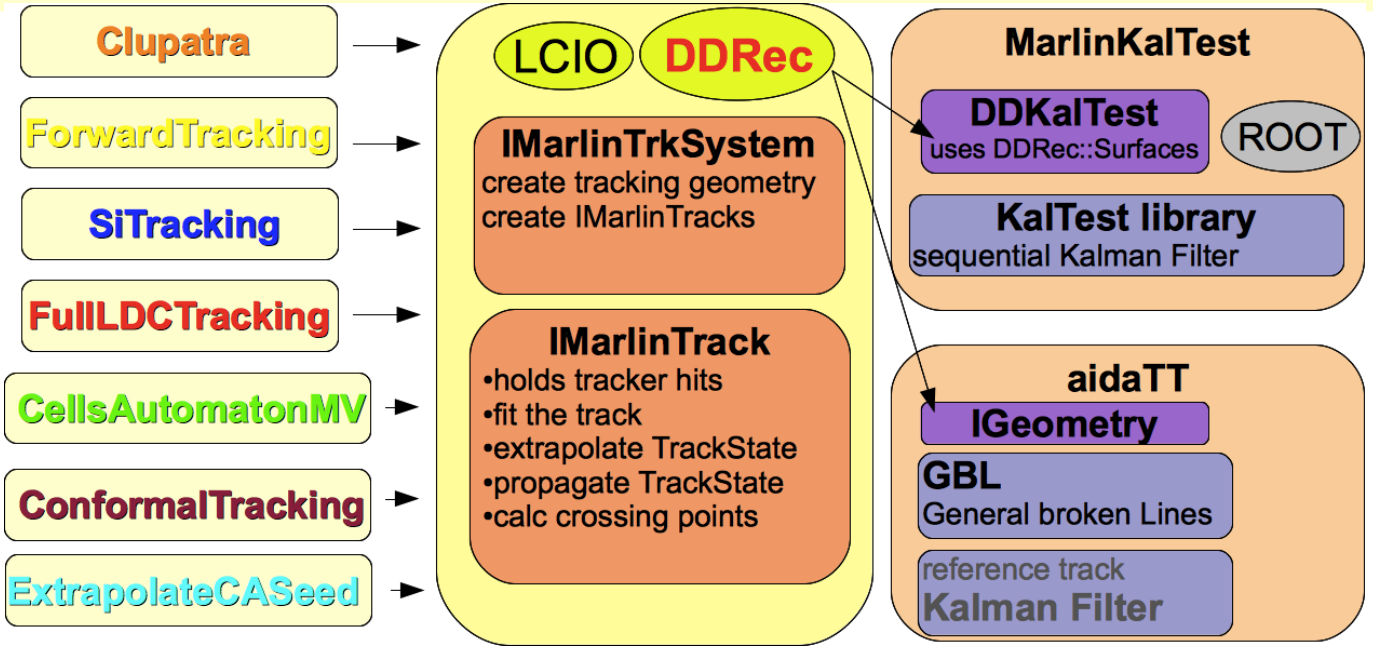


Post DBD tracking code development

- since the DBD we have incorporated the developments from the AIDA project:
  - moved the geometry description to **DD4hep**
    - based on surfaces and detector descriptors
  - introduced a new fitting framework **aidaTT**
    - first implementation using GBL
  - developed new patter recognition algorithms
    - improved Si-Tracking using CA techniques
      - much higher efficiencies for low pt (<~100 MeV)
- adopted code to work for new CLIC simulation model
  - developed pattern recognition for an all Si-Tracker
- recently the code has been adopted by SiD

code now used by all LC detector concepts

IMarlinTrk - LC Tracking Tools



- IMarlinTrk: interface that separates pattern recognition code from actual fitter implementation
  - only dependencies LCIO, DDRec (DD4hep) - NB: no Marlin dependency !
- pattern recognition algorithms have been written to a large extend in plain vanilla C++ (no LCIO, geometry, etc.), e.g.
  - topological clustering, CA libraries, conformal mapping,...
  - currently code lives in iLCSoft libraries - could be extracted to standalone libraries

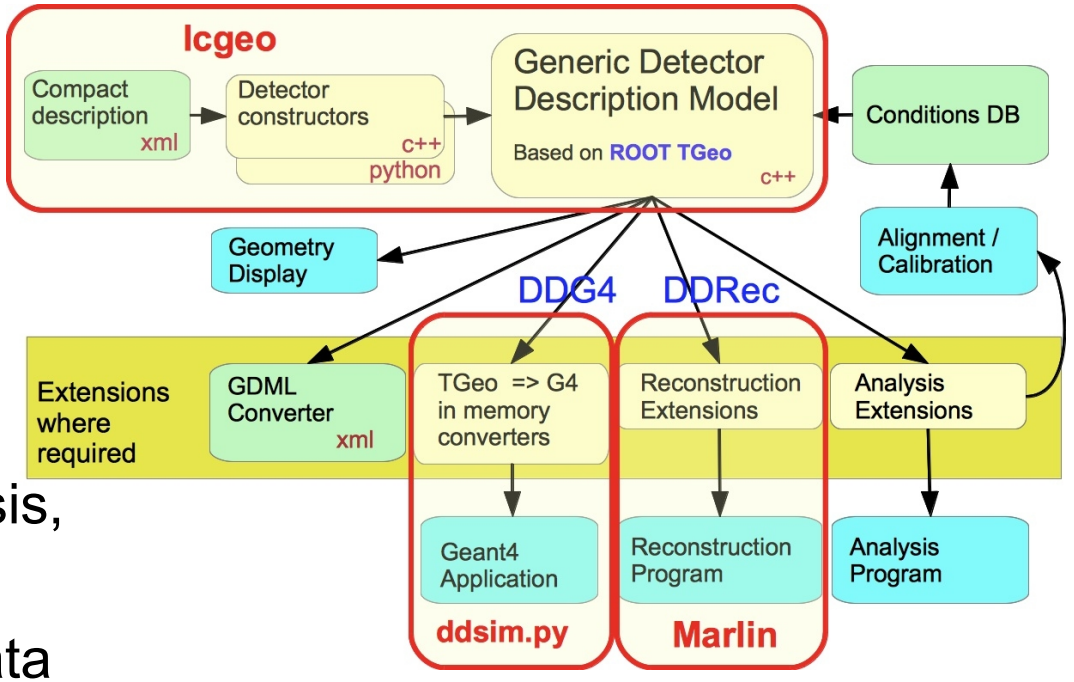
LCIO Tracking related classes

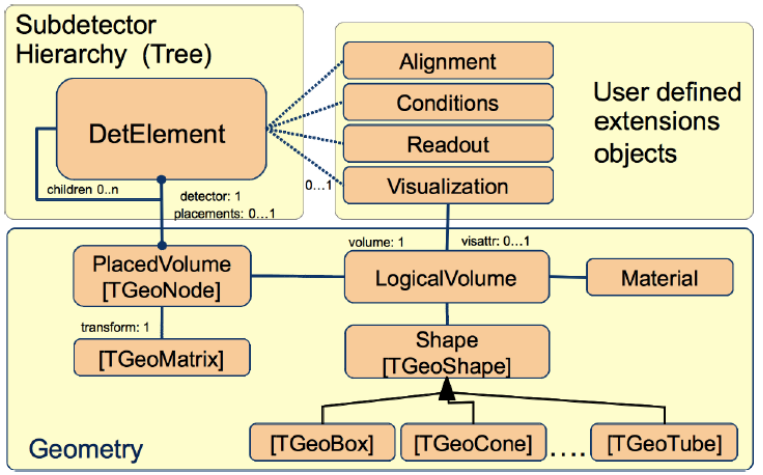
- **LCIO**: lightweight EDM and persistency for Linear Colliders more general for lepton colliders
- obects stored in named collections hold by the Event
- TrackerHits:
  - SimTrackerHit/TrackerHit
    - pos (x,y,z), cov(x,y,z), Edep, cellID, (MCTruth-link)
  - specialization for planar and cylindrical 1D/2D measurements
- Track:
  - vector of TrackerHits
  - ndf, chi2
  - arbitrary number of TrackStates, typically: IP, first/last hit, calo face
- TrackState
  - perigee helix parameters: omega, phi, tan(lambda), d0, z0 (adopted from L3)
  - covariance matrix , reference point

http://lcio.desy.de/

DD4hep - DDRec for Tracking

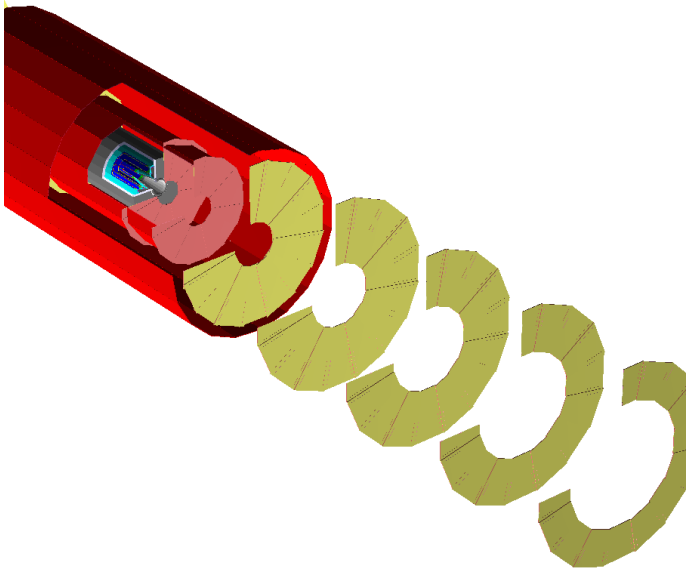
- **DD4hep**: detector geometry description for HEP
  - AIDA project (CERN/DESY)
- support full experiment life cycle
- one source of geometry for
  - simulation, reconstruction, analysis, event displays,...
  - extension mechanism for user data → **DDRec**
- simple detector description classes: extend, layout, #layers,..
- cellID ↔ position
- material properties (point, line)
- tracking Surfaces

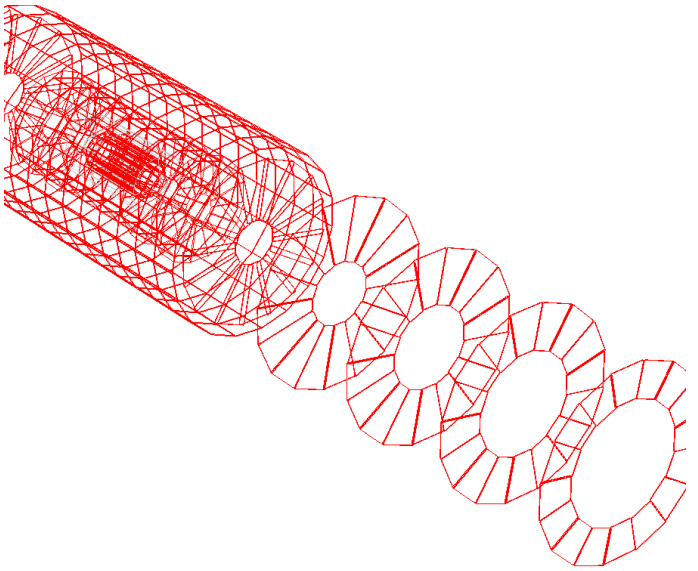




DDRec surfaces for tracking

- tracking needs special interface to geometry
- measurement and dead material surfaces (planar, cylindrical, conical)
- surfaces attached to volumes in detailed geometry model





- u,v, origin and normal
- inner and outer thicknesses and material properties
- local to global and global to local coordinate transforms:
- (x,y,z) ↔ (u,v)

automatic material averaging for surfaces

- material properties are averaged along normal of the surface
- along given thicknesses

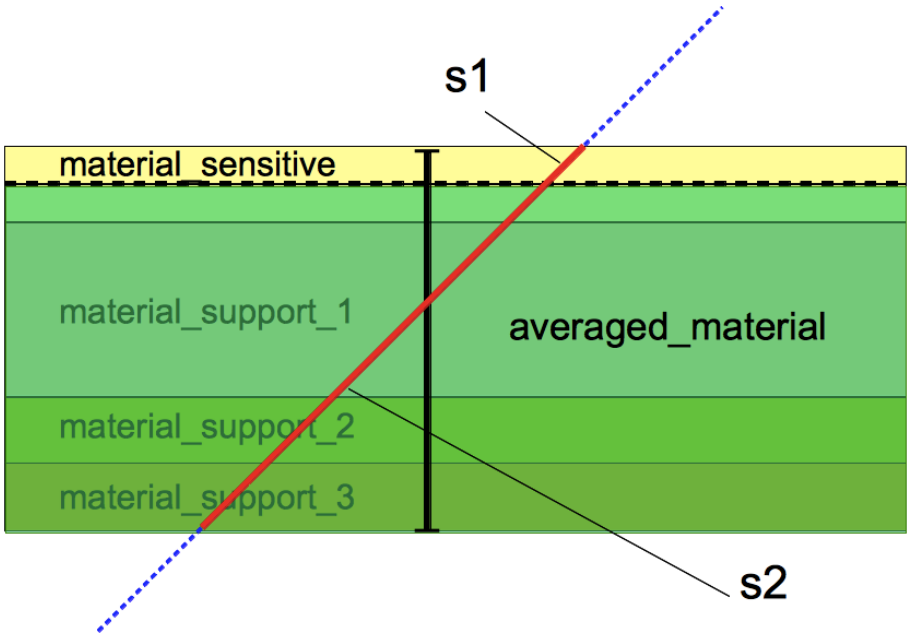
$$< A > = \left(\sum_i^N \rho_i t_i \right) / \left(\sum_i^N \rho_i \frac{t_i}{A_i} \right)$$

$$< Z > = \left(\sum_i^N \rho_i \frac{t_i Z_i}{A_i} \right) / \left(\sum_i^N \rho_i \frac{t_i}{A_i} \right)$$

$$< \rho > = \left(\sum_i^N \rho_i t_i \right) / \left(\sum_i^N t_i \right)$$

$$< X_0 > = \left(\sum_i^N t_i \right) / \left(\sum_i^N \frac{t_i}{X_{0i}} \right)$$

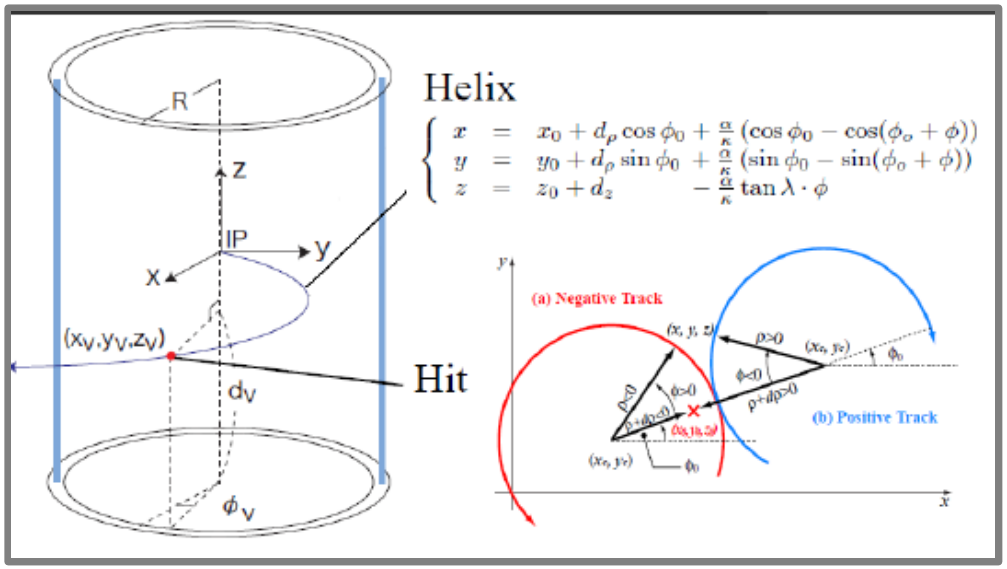
$$< \lambda > = \left(\sum_i^N t_i \right) / \left(\sum_i^N \frac{t_i}{\lambda} \right)$$



- roughly equivalent to individual materials for Bethe-Bloch
- identical for multiple scattering

KalTest Kalman Filter

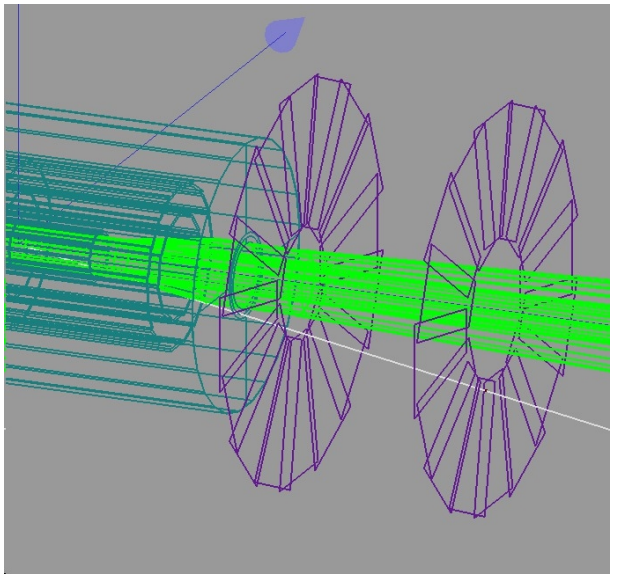
- standalone Kalman Filter
  - developed at KEK (K.Fujii etal)
  - implemented in ROOT
- iterative Kalman Filter
  - follow the track state through the detector
- needs simplified geometry description in terms of surfaces
  - one surface at every material boundary
  - manually programmed based on actual detector geometry (GEAR)
  - provide material information
  - projection of measurements onto surfaces



- uses perigee track parameters similar to L3/LCIO:
  - d0, z0,
  - phi0, tan(Lambda)
  - omega

DDKalTest

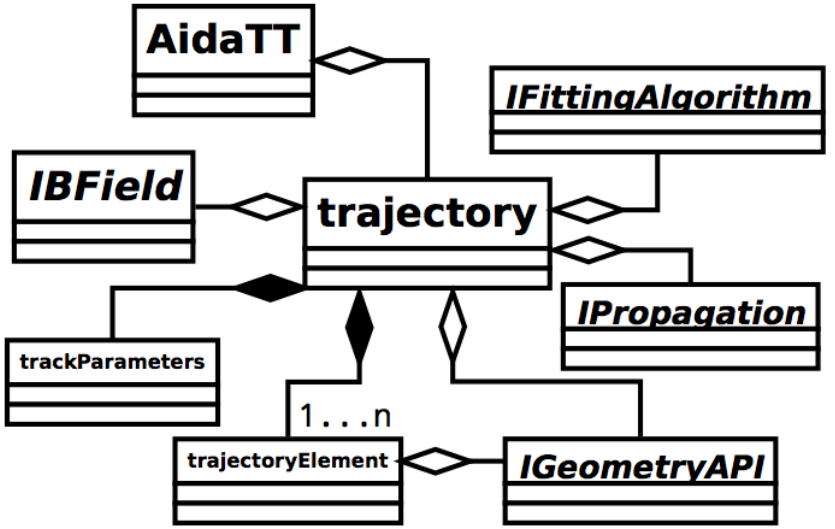
- new package that provides measurement surfaces needed by KalTest using DDRec::Surfaces:
- DDPlanarMeasLayer
  - 1D,2D Si-tracker - barrel/endcap
  - dead materials (endcaps)
- DDCylinderMeasLayer
  - 2D hits in TPC
  - supports (cryostat, field cage,...)
- DDConeMeasLayer
  - conical sections of beam pipe



with DDKalTest we can run the track fitting for every detector that has a DD4hep geometry description (and the surfaces added) !

aidaTT

- generic tracking toolkit developed in AIDA-WP2
- can transparently use a Kalman Filter or the GeneralBrokenLines **GBL**
  - Kalman Filter to be done
- GBL provides interface to Millepede alignment tool
- IGeometry interface uses DDRec::Surface



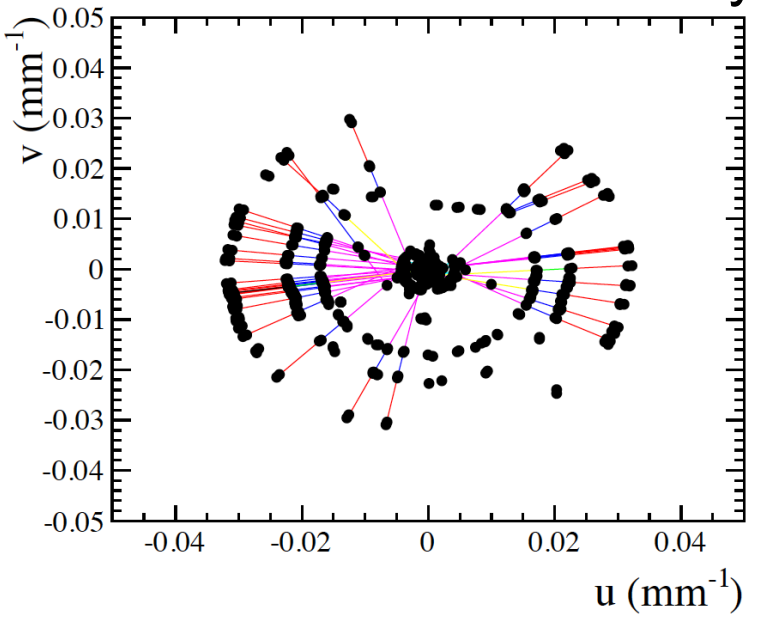
- utility functions:
  - parameter conversions
  - intersection calculations
  - material effects (QMS, dE/dx)
- internally uses curvilinear parameters and analytic Jacobians (CMS)

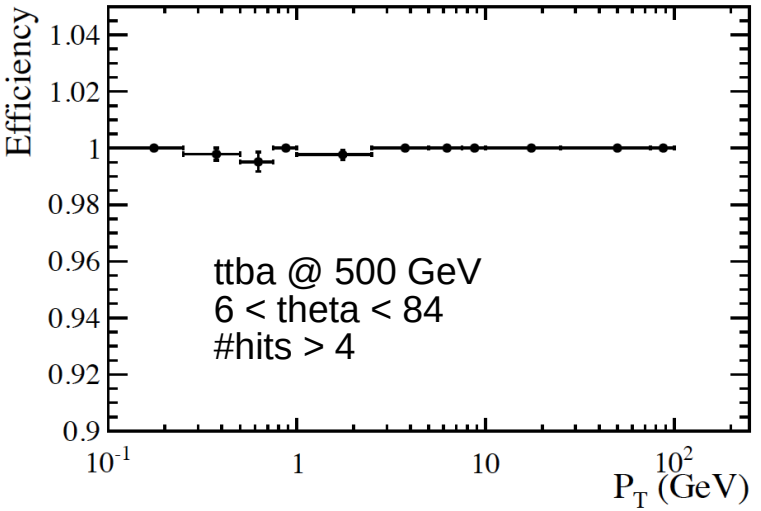
- work continues in AIDA2020
  - additional fitters
  - more pattern recognition tools
  - replace GSL with Eigen
  - parallelization

pattrec example: CLIC ConformalTracking

D.Hynds

- apply conformal mapping to CLIC all Si-tracking
  - map x,y plane to u,v plane:
  - $u = x/r^2 , v = y/r^2 , r^2 = x^2 + y^2$
  - tracks (circles) from IP are mapped to straight lines
- run CA to find tracks in complete detector
  - consistency criterion in z





- global method
- no geometry used !
- could be provided as standalone library/tool

Summary - what we can offer

- the linear collider community has a complete set of tracking tools that are
  - lightweight compared to LHC
  - many pattern recognition algorithms based on
    - topological clustering, Cellular Automatons, conformal mapping
    - partly depending on LCIO and DD4hep - partly standalone
  - track fitting tools that
    - use DD4hep Surfaces as geometry model
    - simple interfaces for tracker hits and tracks
    - framework independent
  - used currently by three detector concepts (ILD, CLICdp, SiD)
  - flexible for adaptation to new detector models
    - in particular if they are described in DD4hep

Outlook - what we like to get

- we are continuously trying to improve our tracking code, e.g.
  - currently navigation is somewhat simplistic and brut-force
    - → would like to benefit from ATLAS code for geometry navigation
  - treatment for non-homogeneous B-fields is not yet optimal
    - → eventually we need a Runge-Kutta solver for arbitrary B-fields
  - implement parallelization where possible
    - → want to benefit from work done for LHC
  - use this forum for exchange of ideas

- would like to see more **common HEP tracking software tools**
- ideally under the umbrella of the **HSF**