Tracking in iLCSoft

Focus on tracking for ILD

F.Gaede

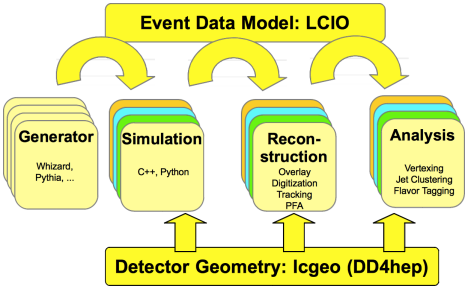
VTX Meeting, Ringberg, May 1-4, 2017

Outline

- Brief overview on iLCSoft
  - LCIO, DD4hep, Marlin
- Tracking tools in Marlin
- Tracking geometry in DD4hep
- Tracking Performance for ILD
- Summary and Outlook
- Open questions for ILD

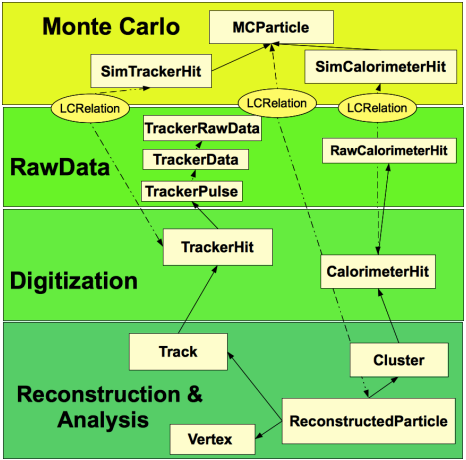
Overview iLCSoft

- iLCSoft is the common software framework for Linear Collider detector studies
  - used by CLIC, ILD, SiD, Calice, LCTPC (and friends: FCC, CEPC, HPS, EIC, . . .)
- key components in iLCSoft:
- **LCIO**
  - the common *event data model (EDM)*
- **DD4hep**
  - the common *detector geometry description*
- **Marlin**
  - the *application framework*



LCIO

- LCIO provides the common *EDM* and *persistency* (i.e. file format for LC studies
- the EDM is hierarchical:
  - you can always get the constituent entities from a higher level object, e.g. the *TrackerHits* that were used to form the *Track*
  - only exception: *you cannot directly go back to the Monte Carlo Truth information*
  - this is possible via dedicated *LCRelation* collections
- everything is stored in *LCCollections*
- collections are retrieved from the *LCEvent* via their **name**
- see: `http://lcio.desy.de`

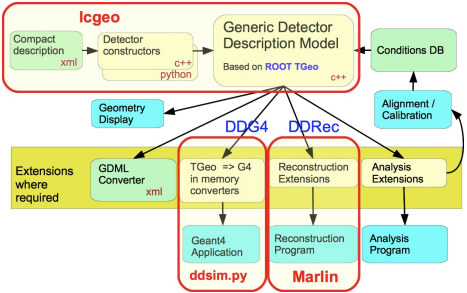


LCIO classes for Tracking

- SimTrackerHit (simulation)
  - position, Edep, link to MCParticle
- TrackerHit (digitization)
  - position, posCov, Edep
- TrackerHitPlane (digitization)
  - position, posCov, Edep
  - measurement directions (u, v, 1D or 2D), errors
- TrackerHitZCylinder (digitization)
  - position, posCov, Edep
  - measurement directions ($r\phi, z$)
- Track
  - *TrackerHits*, χ^2, NDF
  - *TrackStates* at
    - IP
    - first/last hit
    - calo face
- TrackState
  - reference point, (pivot)
  - *perigee parameters*
  - $(\omega, d_0, \phi, tan(\lambda), z_0)$

DD4hep

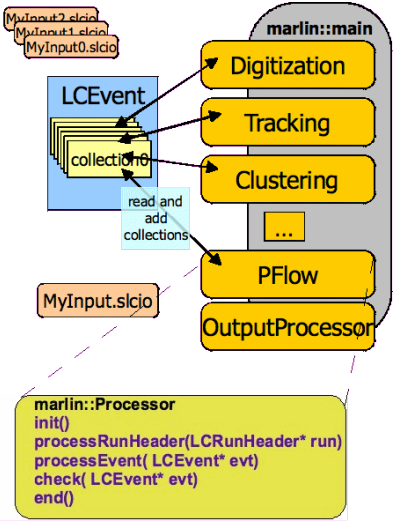
- DD4hep (*Detector Description for HEP*) is the common detector geometry description for iLCSoft
- the **same** detector model is used for:
  - simulation, reconstruction, visualization and analysis
- the detector is fully described via a set of:
  - C++ detector constructors
  - XML files (*compact files*)
- DD4hep is component based, e.d.
  - **DDG4** full simulation with Geant4
  - **DDRec** interface for reconstruction
- **lcgeo**: sub-package with LC detector models
- **ddsim**: python program to run a full simulation



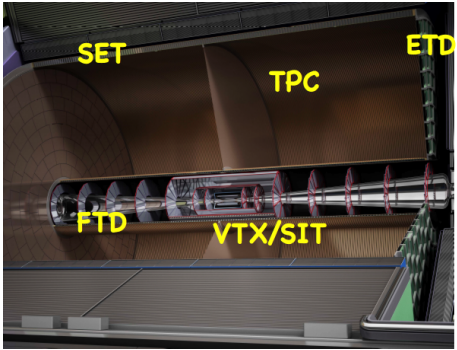
`http://aidasoft.web.cern.ch/DD4hep`

Marlin

- application framework used throughout iLCSoft
- every task is implemented in a *Processor*
  - task can be as trivial as digitizing a hit collection or as complex as running the full *PFA*
- Marlin appplications are fully configured via XML files, defining:
  - global parameters
  - the chain of processors to run
  - per processor parameters
- xml files created with *editor* or via *MarlinGUI*
- more: `http://ilcsoft.desy.de/Marlin/current/doc/html/index.html`



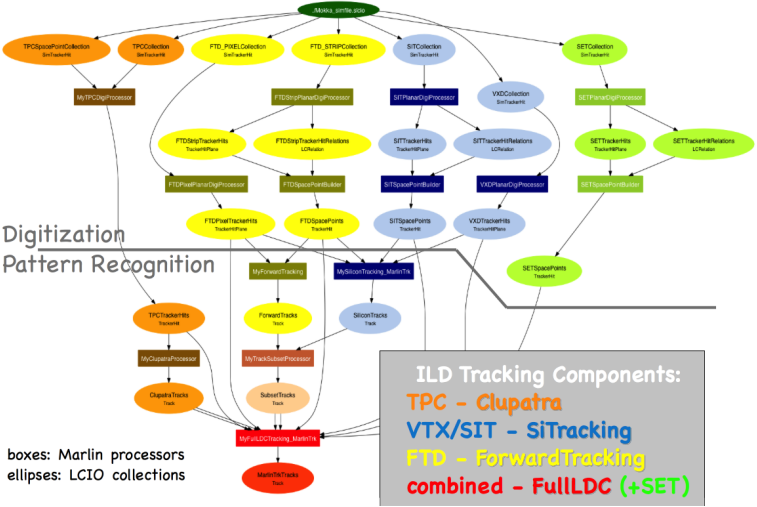
The ILD tracking system



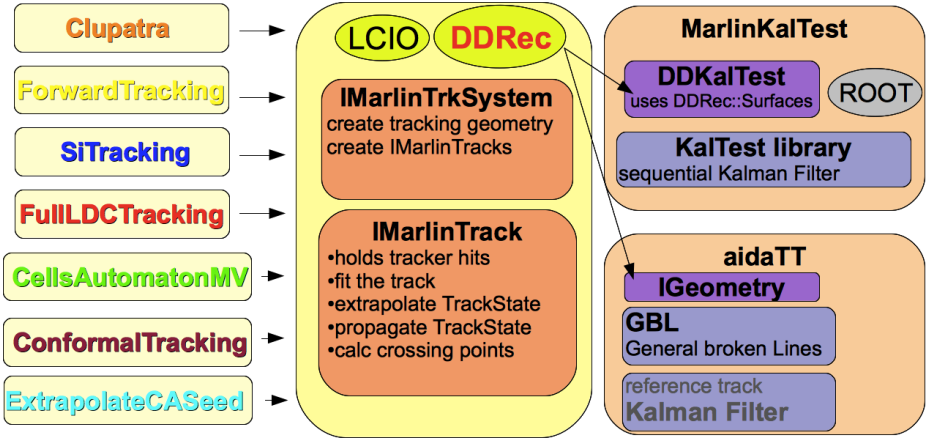
Detector	Point Resolution		
VTX	$\sigma_{r\phi,z}$	=	2.8μm (layer 1)
	$\sigma_{r\phi,z}$	=	6.0μm (layer 2)
	$\sigma_{r\phi,z}$	=	4.0μm (layers 3-6)
SIT	σ_{α_z}	=	7.0μm
	α_z	=	$\pm 7.0^\circ$ (angle with z-axis)
SET	σ_{α_z}	=	7.0μm
	α_z	=	$\pm 7.0^\circ$ (angle with z-axis)
FTD *Pixel*	σ_r	=	3.0μm
	$\sigma_{r_\perp}$	=	3.0μm
FTD *Strip*	σ_{α_r}	=	7.0μm
	α_r	=	$\pm 5.0^\circ$ (angle with radial direction)
TPC	$\sigma^2_{r\phi}$	=	$(50^2 + 900^2 \sin^2\phi + \left((25^2/22) \times (4T/B)^2 \sin\theta\right)(z/\mathrm{cm}))\,\mu\mathrm{m}^2$
	σ^2_z	=	$(400^2 + 80^2 \times (z/\mathrm{cm}))\,\mu\mathrm{m}^2$
	where ϕ and θ are the azimuthal and polar angle of the track direction		

tracking resolutions used in DBD - can we do better now ?

ILD tracking processors in Marlin



MarlinTrk



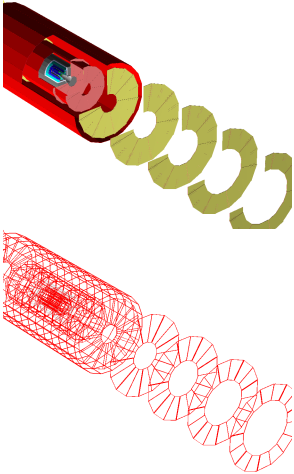
- pattern recognition uses *IMarlinTrk* interface
  - can choose actual track fitter w/o code change

tracking geometry: surfaces

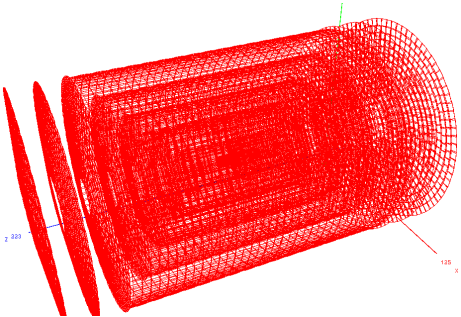
- tracking needs special interface to geometry
- measurement and dead material surfaces (planar, cylindrical, conical)
- surfaces attached to volumes in detailed geometry model

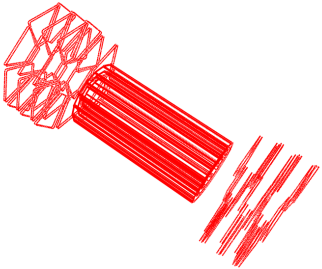
surfaces:

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



tracking surfaces for CLIC all Si-tracker

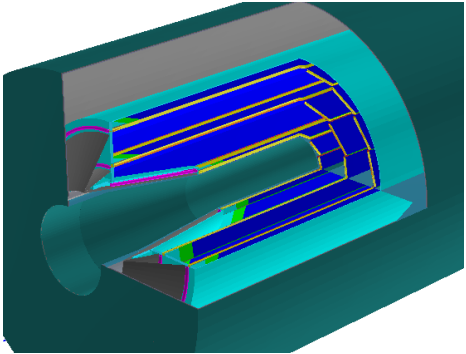




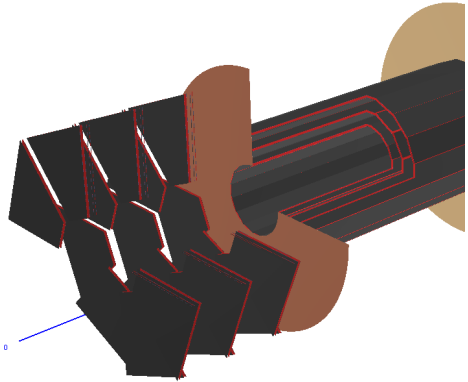
- O(10^4) surfaces[1]: ideally need fast navigation in layer geometry

[1] pictures show older design of CLIC tracker

Vertex detector models in lcgeo/DD4hep



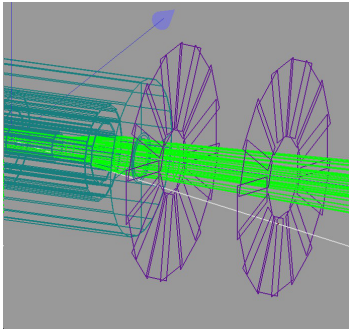
ILD vertex detector



CLIC vertex detector

DDKalTest

- provides measurement 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



Generic track fitting

- can run track fitting for any detector that defines these surfaces in DD4hep

surfaces and materials

- material properties are **automatically averaged**
  - from detailed model
  - along normal of the surface along given thicknesses

averaging materials

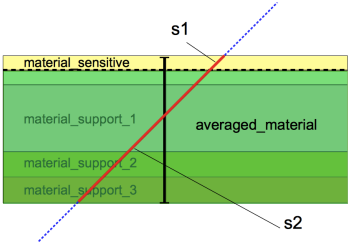
$$< A >= \left(\sum_i^N \rho_i t_i\right) / \left(\sum_i^N \rho_i \frac{t_i}{A_i}\right) \qquad t_i \textit{ thickness}$$

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

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

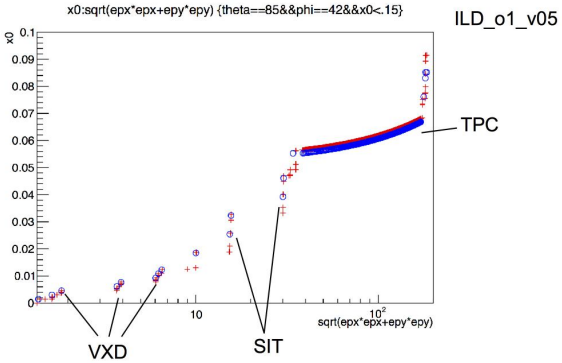
$$< X_0 >= \left(\sum_i^N t_i\right) / \left(\sum_i^N \frac{t_i}{X_{0i}}\right)$$

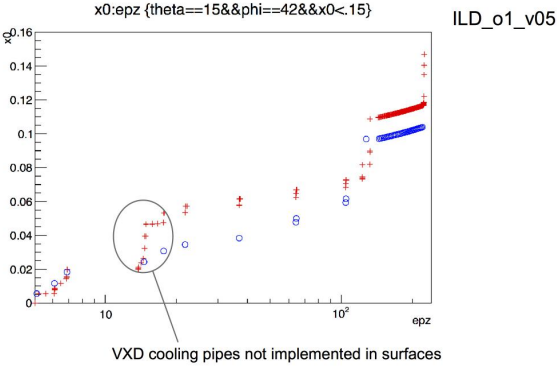
$$< \lambda >= \left(\sum_i^N t_i\right) / \left(\sum_i^N \frac{t_i}{\lambda}\right)$$



roughly equivalent for Bethe-Bloch - identical for multiple scattering

material: comparison surfaces vs. detailed model

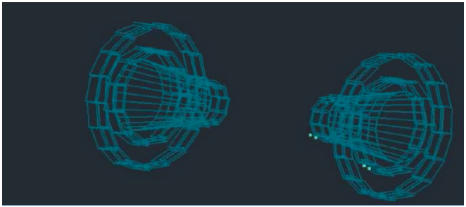




- surface describe material well in most regions - but not everywhere . . .

improving the material description

- correct material description is crucuial for correct track fit
  - pull distributions
- in general approach with surfaces works well
- need to add more surface in *"complicated"* regions like the VXD



VXD support surfaces

- Question: when is it good enough ?

Navigation tracking geometry

- aidaTT:
  - compute all intersections of reference track with the surfaces and sort wrt. s (path length)
- KalTest/DDKalTest:
  - compute a *global ordering* of all surfaces in the geometry wrt. *one single ordering parameter*:
    - radius of planes and cylinders parallel to z
    - outermost radius of disk like measurement surfaces
    - potentially a source of errors and *not fast*
- ideally we would like to have a smart topological pre-ordering of the surfaces (a la ATLAS)
  - not *strictly needed* but would be *nice to have* for large MC production
  - trade-off: manpower vs CPU power ...

Track Digitizers

DDPlanarDigiProcessor

- Gaussian smearing of mean hit position from particle's energy depositions
- along u,v measurement directions (2D,1D)
- taken from DDRec::Surface
- used for **all Si-Trackers** (pixel, strips)

TPCDigiProcessor

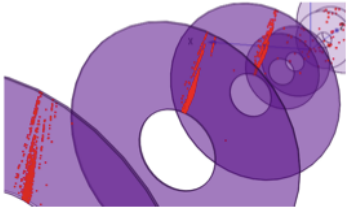
- dedicated TPC digitizer:
- parameterized point resolution as function of $(\phi_{track}, z_{drift})$
- parameterized double hit resolution

detailed, realistic Si-Digitizers

- so far no need to use these for MC mass producion

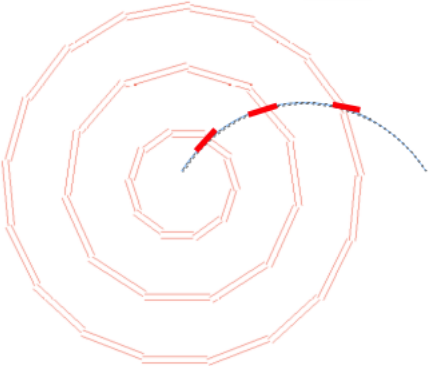
SpacePointBuilder

- introduce realistic treatment for Si-strip detectors
- combine pairs of digitized 1d *TrackerHitPlanes* from double layers with strip stereo angle into *TrackerHits* with 3d space points
  - incl. correct covariance matrix
- try to avoid parallax problem by projecting to the IP
- all possible hit pairs that result in hits laying within the bounded surface of the wafer (rectangle/trapezoid) are used
- including ghost hits
- space points are used for pattern recognition
- final track fit uses 1d measurements and errors



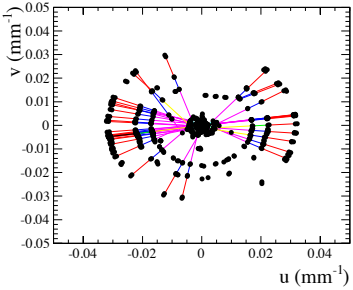
example patrec: CellularAutomatonMV (Y.Voutsinas)

- combine hits from double layers into mini-vectors
  - based on $dist(h_1, h_2)$ or $\Delta_\phi, \Delta_\theta$ criterion
  - removes unwanted combinations (pair-bg)
- apply *CellularAutomaton (CA)* for combining MVs
  - based on ϕ, θ
- improves tracking efficiency for low p_t considerably
- issue: so far only in VXD - extension to SIT difficult due to 1d strip hits



example patrec: 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 *CellularAutomaton (CA)* to find tracks in complete detector
  - consistency criterion in z



global pat rec method

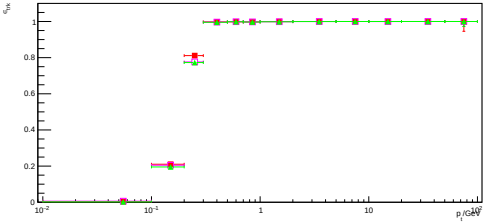
- no geometry used !
- detector independent...

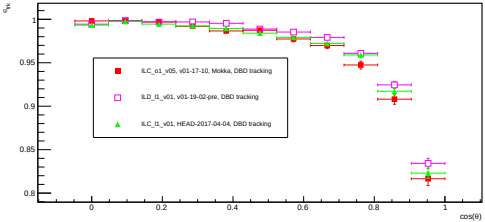
conformal mapping used at Star for CLIC

Performance of ILD Tracking

- DD4hep was introduced only about two years ago into iLCSoft
  - DBD tracking based on Mokka simulation and Gear geometry
- ILD is planning a large MC production for detector optimization and physics benchmarks
- neede to demonstrate performance of *new* ILD tracking
  - as comparison with DBD-like Mokka based tracking

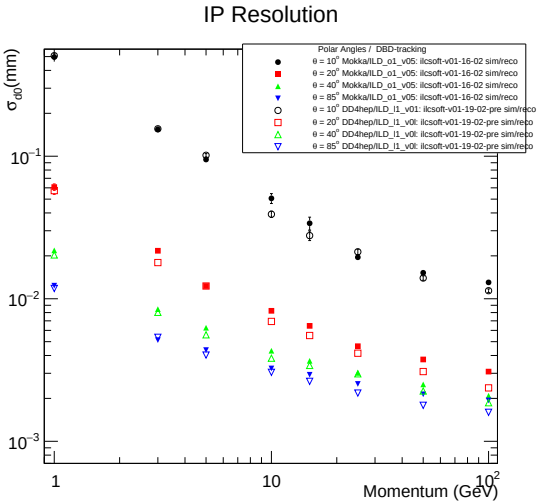
Tracking efficiency





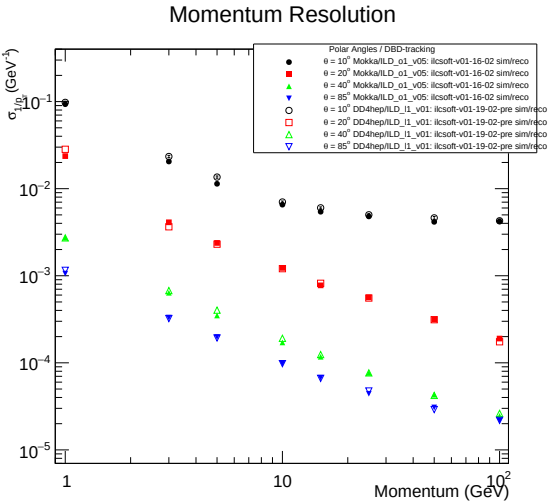
- observe better tracking efficiency than before
- potential reasons:
  - bugs in pattern recognition fixed in the meantime
  - bugs fixed in having more consistent geometry description

Impact parameter resolution



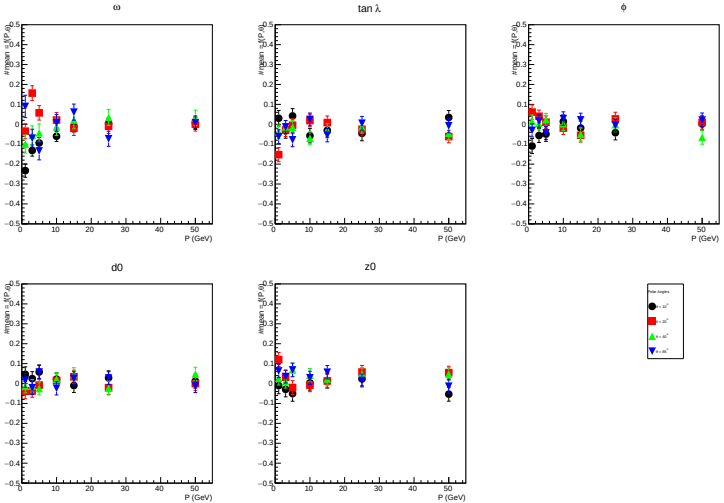
- observe compatible impact parameter resolution
- in same regions slightly better

Momentum resolution



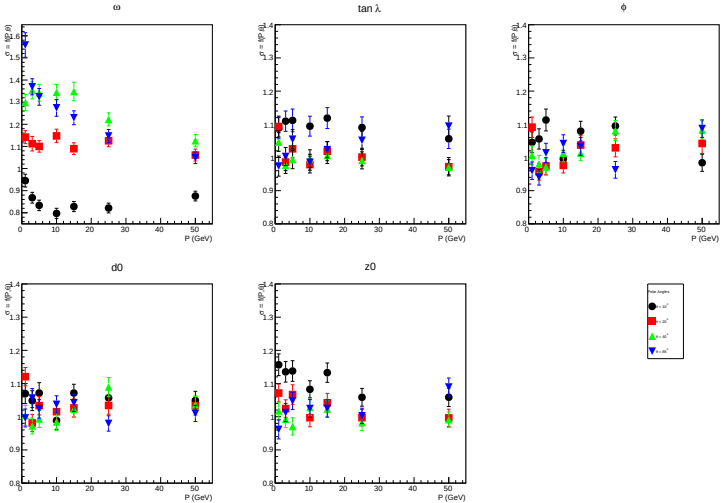
- observe compatible momentum resolution
- in same regions slightly better, worse in others

Pulls - mean value



- comparable to Mokka based (not shown)

Pulls - sigma



- comparable to Mokka based (not shown)

Summary and Outlook

- iLCSoft has detecor independent tracking tools (*IMarlinTrk*)
  - based on *DDRec::Surfaces* geometry and material description
  - used by CLICdp, ILD and SiD (and others)
- provide *flexibility* to study tracking detector layouts in full simulation rather easily
- demonstrated that for ILD we reach comparable or better tracking performance compared to the old software tools used for the DBD
- Next steps:
  - finalize transition from *Gear* to *DD4hep*
  - improve **CPU performance of tracking code - e.g introduce smarter navigation**
- Open questions for ILD tracking: *next slide*

Questions for ILD Tracking and MC mass production

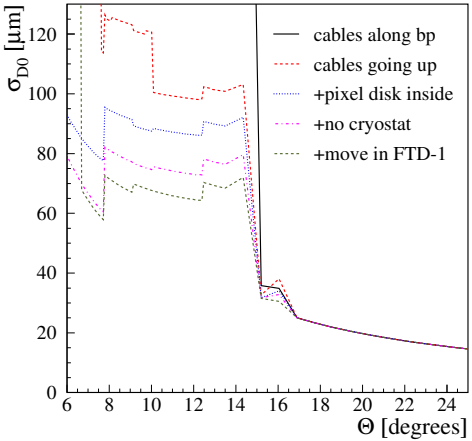
can we introduce a *pixel* technology for the SIT ?

- same material budget and layout (use existing model)
- same or better point resolution in r, ϕ
- single BX tagging (readout time $1\mu s$)
- have two double layers (mini vectors)

which performance numbers to use for the VXD (et al) ?

- point resolution ($3\mu m$?)
- readout time ($1\mu s$?)

can we move the first FTD disk closer to the IP ?

- improve impact parameter resolution
- size of cryostat and VXD services

ILD forward tracking options with SGV