PRS-μ

CERN, Tuesday 14 October 2003

New framework for muon reconstruction in DT

and new segment builder for DT

Stefano Lacaprara

Stefano.Lacaprara@pd.infn.it

INFN and Padova University

Outline

- New reconstruction framework,
- new RecHit interface,
- new segment building algo,
- status and release plan,

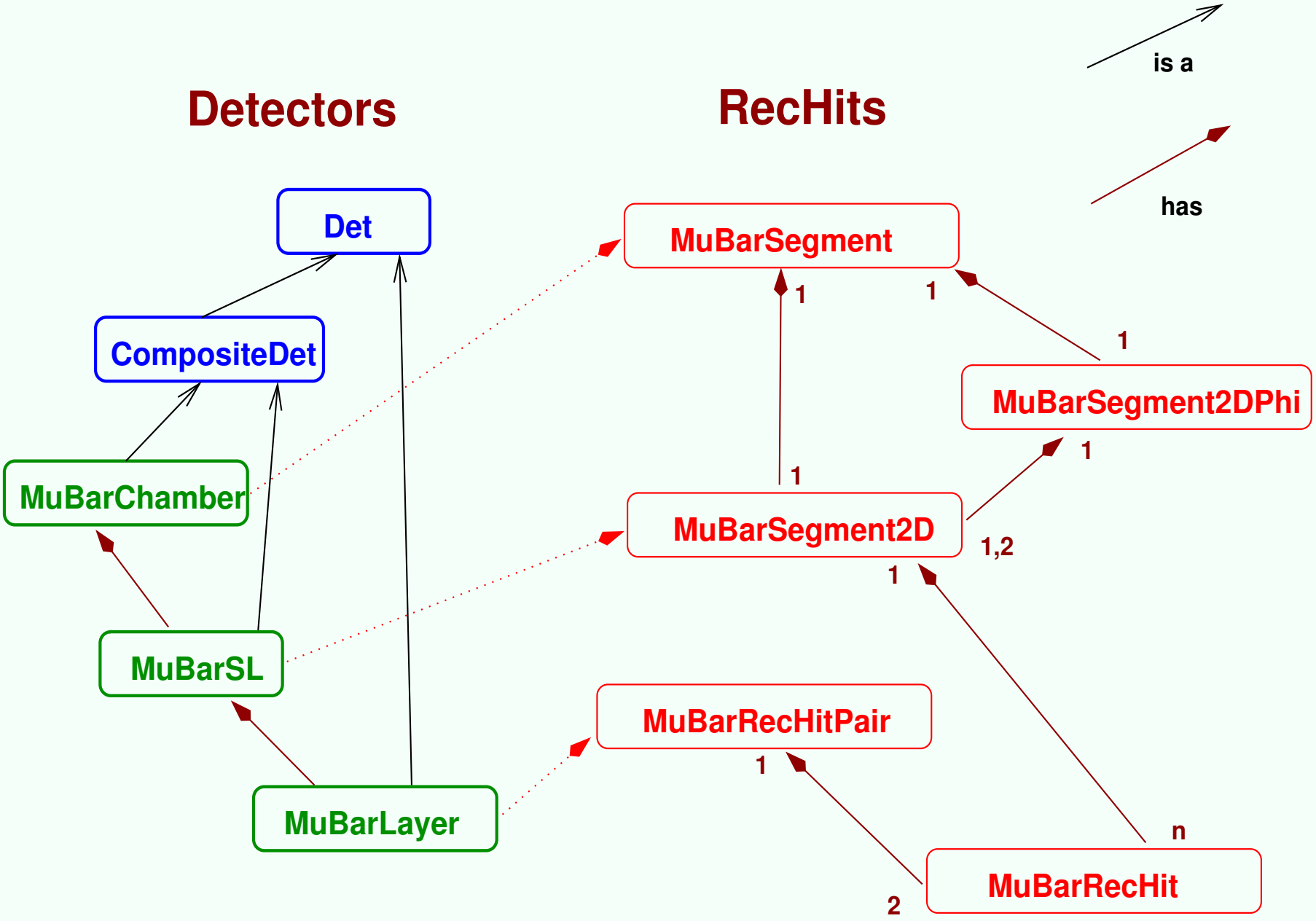
Motivation

- DT geometry fully CommonDet aware
- DT reconstruction is not.
  - CommonDet way: every Det should provide RecHits
  - MuBarChamber, SL and Layer are Det, but only Chamber returns RecHits
  - The Chamber RecHits are 4D segments
  - a 4D segment is built from 2D segments
  - 2D segments are built in zed SL and in ϕ projection (two SLs)
  - 2D segments are built from hits
  - very complex and non CommonDet to access 2D segs and hits from a segment
  - Not possible to get 2D segments w/o reconstructing 4D seg, or in each ϕ SL
  - The hits (associated and not) are accessible via non CommonDet interface

New reco architecture for DT

- MuBarLayer, MuBarSL and MuBarChamber provides RecHits
- Create 3 different types of RecHits (actually 5, see after)
- Guarantee a uniform access via CommonDet interface
- Allow the possibility to use any kind of RecHits in the track reconstruction, not only 4D segments (as now)
- Address new functionality needs, mostly TestBeam related
- Segments are composite RecHit: easy access to components
- Common interface: all hits/2D segs/4D segs are accessed only via RecHit interface (no specific interface any more), enforcing the by-value semantic

DT RecHit architecture



General features

- all various `MuBar(Segment|RecHit|Pairs...)` *are* `BasicRecHit` (4D,2D,1D)
- Every `MuBar Det` has a `RecDet<RecHit>` which triggers the reconstruction (on-demand) and stores the results in a cache for further access
- only `RecHit` can be accessed, never `BasicRecHit` directly: `RecHit` interface is wide enough to get all possible informations from whatever concrete reconstructed object (hit, segment)
- (technicality) every `RecHit` is a `Proxy` of a `BasicRecHit`, which is `ReferenceCounted`.

MuBarLayer RecHits

- A `MuBarLayer` reconstructs pair of hits, given the Left/Right ambiguity
- Described as `MuBarRecHitPair`
- Each `MuBarRecHitPair` has two `MuBarRecHit`
- a `MuBarRecHit` is built from Digis according to the reco algorithm (time→distance)
- `MuBarRecHit` lives in the `MuBarLayer`
- a `MuBarRecHitPair` is a composite `BasicRecHit1D`, its components are `MuBarRecHit`, which are `BasicRecHit1D` as well
- a `MuBarRecHit` provides position (that measured by the drift cell): the measured coordinate is **always** the **x**, in `MuBarLayer` frame
- is a 1D `BasicRecHit` because it actually measures just one coordinate

MuBarLayer RecHits (2)

- MuBarLayer::recHits() return (by value) a vector of RecHit
- The RecHits are MuBarRecHitPair, namely pairs of hits
- by default is not associated, so the L/R is not solved
- The position of a MuBarRecHitPair is $(pos_L + pos_R)/2 = pos_{wire}$
- the error is $(pos_L - pos_R)/2$
- if L/R solved: position and error are those of the "correct" RH
- `MuBarRecHit` accessed via `RecHit::recHits()`, which returns a `vector<RecHit>` (always by value), with two components (Left & Right RH)
- Long term plan: model pairs as MultiRecHit, and hits as WeightedRecHit; possible use of DAF for segment building

MuBarSL RecHits

- Brand new: not present so far
- Segments built in a SL (just one projection)
- Described as `MuBarSegment2D` *isa* `BasicRecHit2D`
- Built from `MuBarRecHit`, solving L/R ambiguity, and updating the RH position using impact angle knowledge
- Every segment `has` a vector of `MuBarRecHit`
- They are **different** from the ones of the layers, because the position is updated
- the RH of the Layer are **not updated**, only the ones owned by the segments are
- A copy of the original RecHit is updated and saved: the original is not touched
- Consequence: a `MuBarRecHitPair` accessed via a `MuBarLayer` cannot be associated, with L/R solved in the current framework

MuBarChamber RecHits

- Segment built in a chamber (possibly two projections)
- Described as `MuBarSegment` *isa* `BasicRecHit4D`
- actual dimension depends whether built from one or two projections
- Built from a zed SL segment and a ϕ one
- ϕ SL segment is described as `MuBarSegment2DPhi` which *isa* `MuBarSegment2D`
- a MuBarSegment2DPhi is built from
  - a segment (2D) built from hits of the two ϕ SLs (implemented)
  - two segments (2D), each built in one SL, and then matched (not yet implemented)

MuBarChamber RecHits (2)

- A segment owns two SL segments
- the SL segments own their hits
- the hits position and error are updated using the segment info (position along wire, impact angle)
- the SL segments are updated as well, with updated hits
- As before, copies of all RecHits are made: the original RH (either hits or 2D segs) are NOT updated
- Easy access to all components
  - From a 4D seg: `MuBarSegment::recHits()` returns the 2d segments
  - for ϕ projection can be either the "supersegment" (built from 2 SLs) or the two original SL segs
  - From 2D seg: `MuBarSegment2D::recHits()` return the hits (updated)

Usage example

- Get all hits of a chamber
  - Get the MuBarChamber pointer
  - Loop through its layers (via `MuBarChamber::getLayers()`)
  - Get the RecHits: NB. are MuBarRecHitPair via `vector<RecHit> MuBarLayer::recHits()`
  - Loop through the RecHits and get RH components via `vector<RecHit> RecHit::recHits()`
  - These are the hits (both L/R): position in Layer via `RecHit::localPosition()`

Usage example (2)

- Get all segments 2D in a chamber, and access the hits of these segs
  - Get the MuBarChamber pointer
  - Loop through its SLs (via `MuBarChamber::getSLs()`)
  - Get the RecHits: NB. are MuBarSegment2D via `vector<RecHit> MuBarSL::recHits()`
  - These are 2D segments: **in SL frame**
  - Loop through the RecHits of the segment and get RH components via `vector<RecHit> RecHit::recHits()`
  - These are the hits, **in layer frame**: ONLY the hit actually used to built the segment is returned (L or R), NOT the pair

Usage example (3)

- Get all segments 4D in a chamber, and access the hits of these segs
  - Get the MuBarChamber pointer
  - Get the RecHits: NB. are MuBarSegment via `vector<RecHit> MuBarSL::recHits()`
  - These are 4D segments: **in Chamber frame**
  - Get the components via `vector<RecHit> RecHit::recHits()`
  - These are 2D segment **in SL frame**: zed and "superϕ" or zed and two ϕ's
  - get again componets via `vector<RecHit> RecHit::recHits()`
  - These are the hits, **in layer frame** updated for position **NB. are DIFFERENT from the ones got via the Layer (which are NOt updated**

New segment building algo

- Not only new architecture but also a brand new segment building algo

- First step: collect hit (MuBarRecHit) from layers
  - Segments2D are built in SL frame
  - Hits are defined in Layer frame: move to SL one
  - Collect hits from:
    - a SL: to build ordinary 2D segment in a SL,
    - two SLs (the first one define the frame): to build a "superϕsegment" to be used by a 4D segment in chamber
    - or a collection of layers (to be implemented): for user case (e.g. want to check the mis-alignment of a Layer inside a SL)

New segment building algo (2)

- Take two hits in different layers compatible with (loose) IP hypothesis
  - Collect all hit compatible with this seed (according to hit pos and error): Can be both L/R or just one
  - more than one hit per layer allowed (happens for inclined tracks)
  - If enough hits ($n > 3$), then build a different segment for each L/R hypothesis of each ambiguous RH: full combinatorics
  - Fit (linear) hits: compute χ^2, pos and dir (use `CommonDet/StatUtilities/LinearFit`)
  - Save only the segment with most hits and best χ^2
- Go on with other hits pairs
- Check for double segments

New segment building algo (3)

- Got a collection of segments candidates: check for conflict and ghosts
  - Hit sharing is allowed (max configurable, default 2)
  - Eventual shared hits must agree on L/R! if not, remove the conflicting hits from the worst segment (delete it if too bad).
  - If more shared hits than allowed, remove the worst segment
- return the result: usually just one segment
- Build a `MuBarSegment2D` for each segment candidate: update the constituents hits with impact angle correction and refit the segment
- Build a `RecHit` from the `BasicRecHit` and store it in the cache of the RecDet

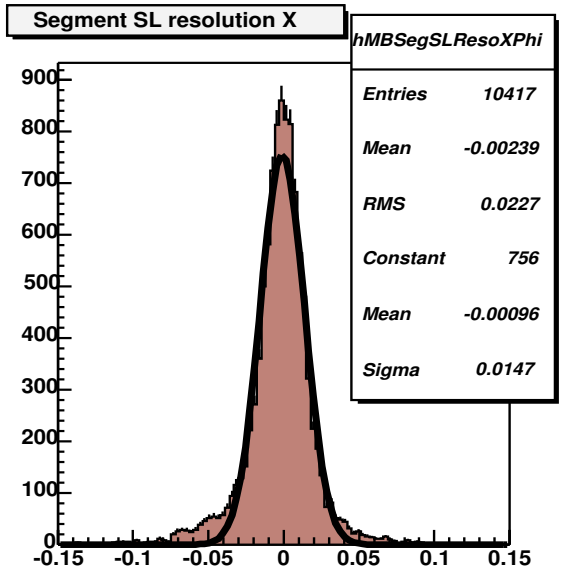
New segment building algo (4)

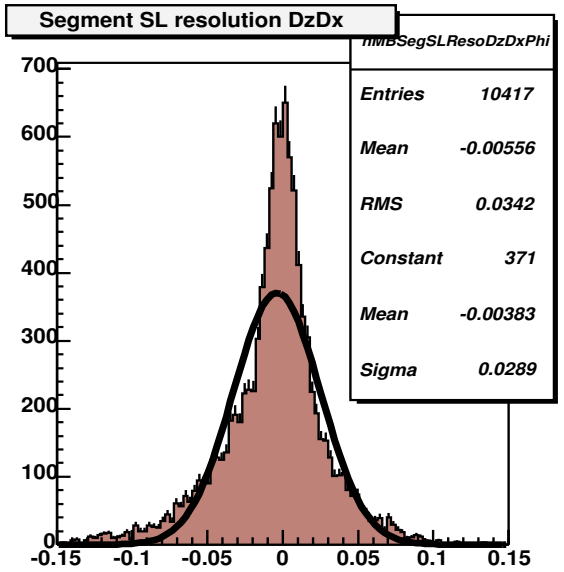
- Building of 4D segment in chamber starts from 2D segments in SL
  - Just take zed SL segment (if any)
  - for ϕ projection two options:
    - Build a "superϕ segment" from hits of the two SLs (same algo)
    - Use the 2D segments of the two SLs, match them, build a "superϕ segment" out of them (to be implmented)
  - Build a `MuBarSegment` from the two projection (or from just one)
  - update the 2d segs and their hits, then update also the 4d seg
  - save it as a `RecHit` into the `RecDet` cache

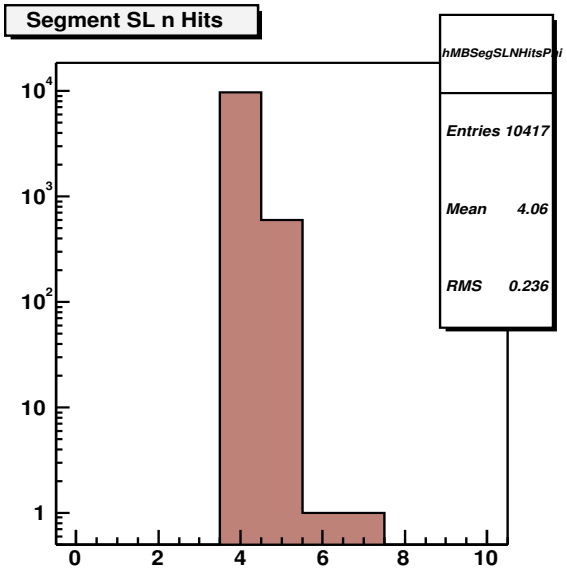
Status

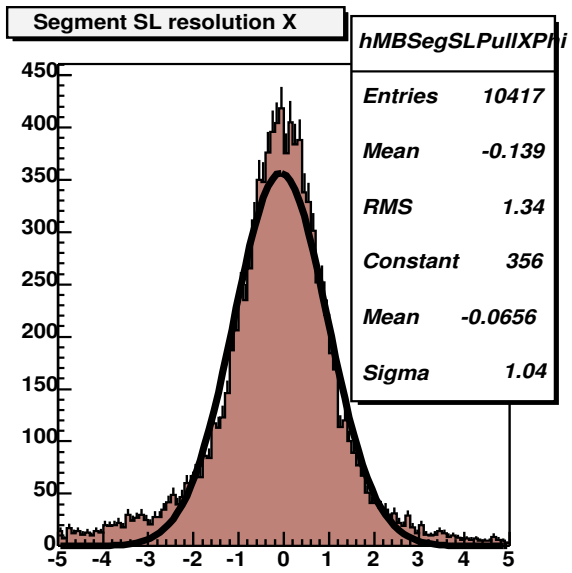
- Code is written, but not yet ready
- Test on functionality and interface is ok
- Performances still under study
- Want to test the algo on TestBeam data!!
- Results (**very preliminary**) shown here using old digitization and hit building
- Lack of a visualization tool is a serious limit in development
- How does the algo perform in presence of showering? Do we get the correct segment? Hard to say w/o looking at the event
- Iguana shows only a fraction of needed information: can be improved, but the use off-CERN is basically impossible

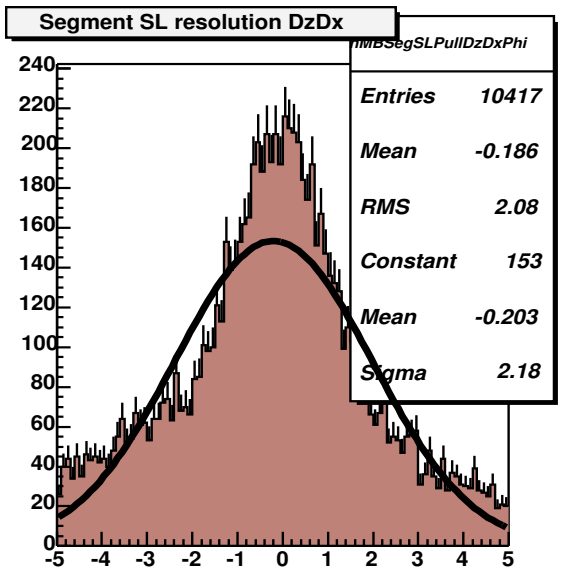
Segment in ϕ SL

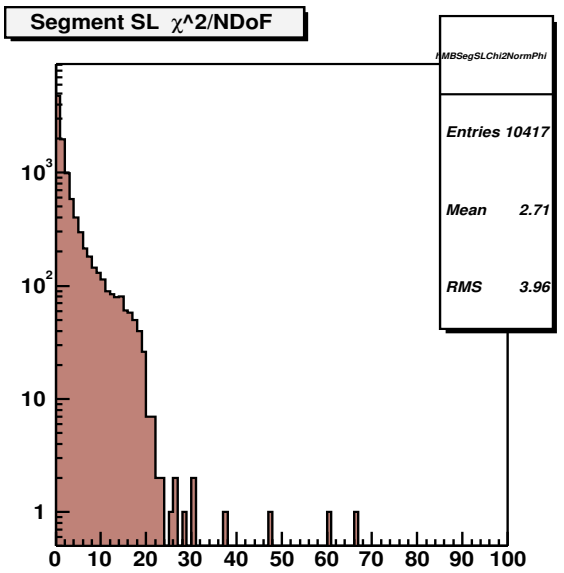






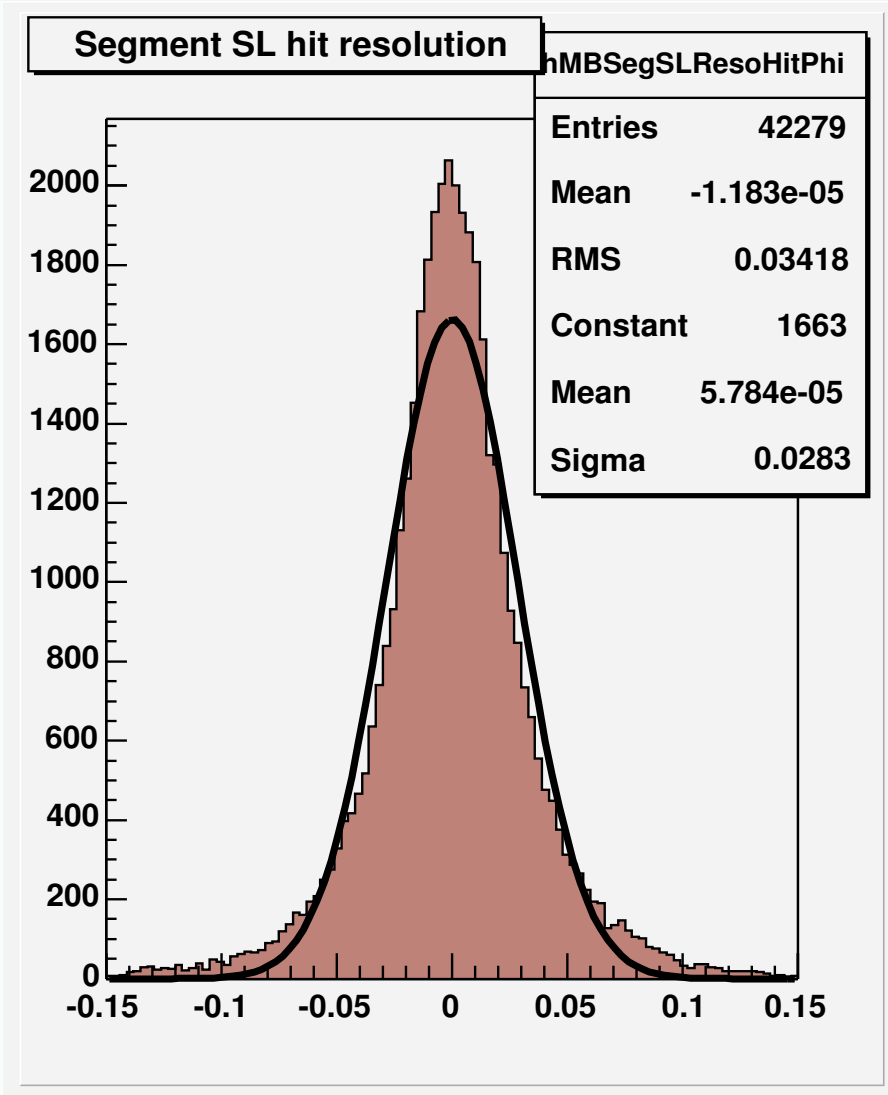


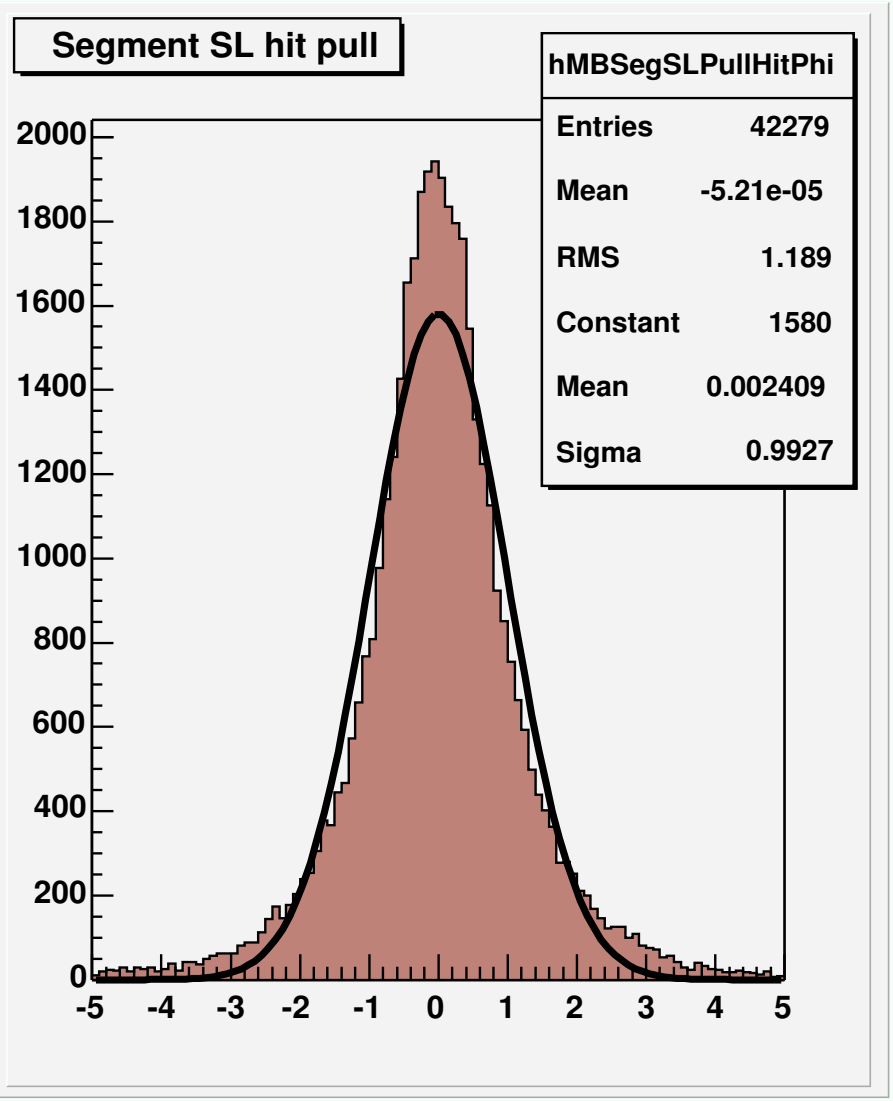




Hits in ϕ SL segments

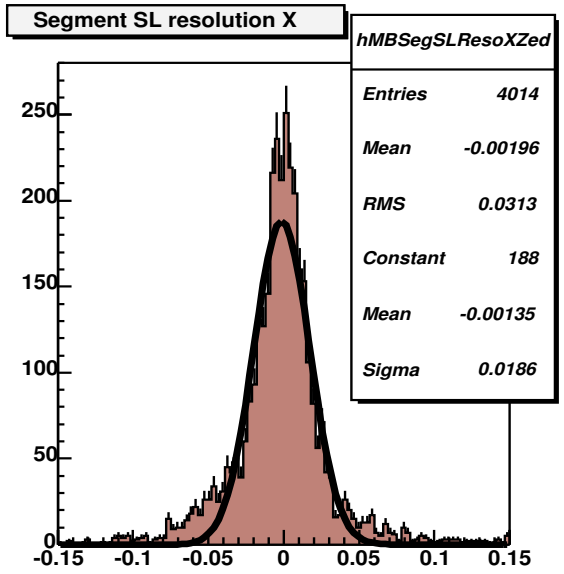
Resolution and pull of Hits in a SL segment wrt segment

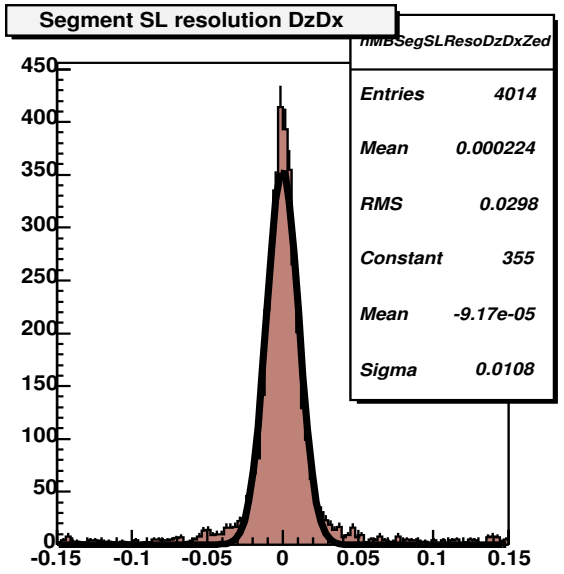


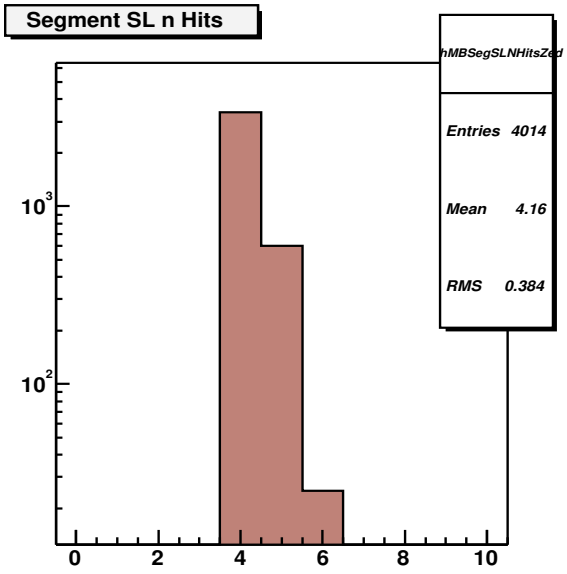


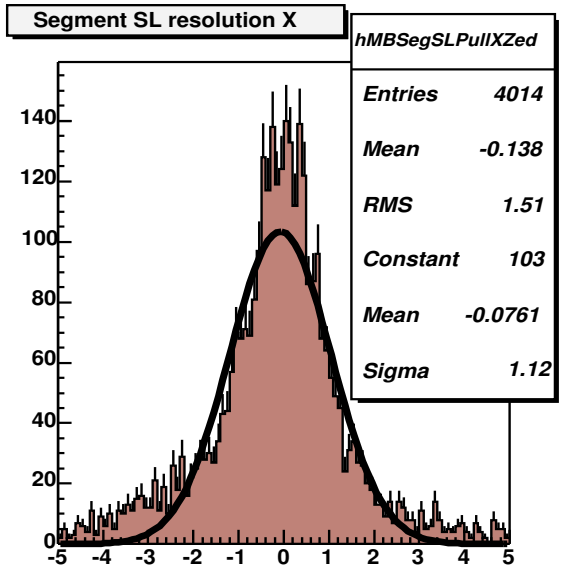
Plot done in 15“ !!!

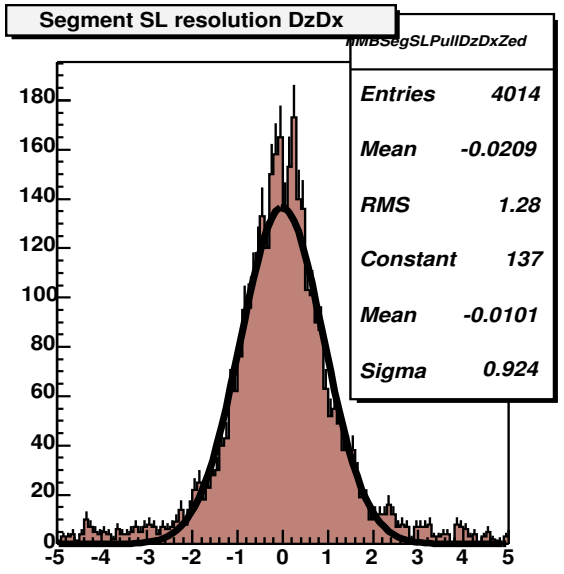
Segment in Zed SL

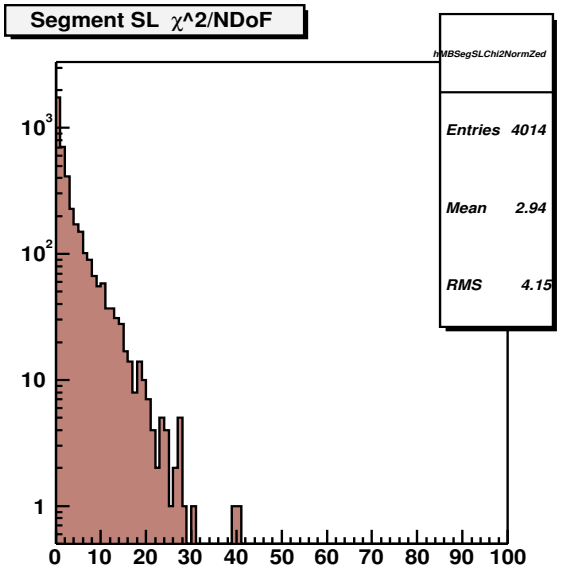












Release plan

- Not for ORCA 750
- need further test and developing before release
- Impact on track reconstruction (easy to solve)
- Allow for more sophisticated pattern recognition and fit (using not only chamber segments but also 2d segs or even hits)
- Documentation to be written (no release without it!)
- Timescale: $2 \div 3\ weeks \oplus$ time spent to build ORCA (pre)release: $= ??$
- Take chance to decouple Hit reconstruction from geometry in DT

Many thanks to Teddy for many hints and fruitful discussion