HPS Tracking Overview

Matt Graham

HPS Collaboration Meeting

June 6, 2013

We need YOU!

- This talk is mainly a plea for help!
- alignment
- track finding optimization
- include hit times in the tracking fit
- vertexing in varying magnetic field
- modified/dedicated tracking for vertexing analysis?
- Kalman Filter and/or broken lines for track fitting

Review track finding & fitting

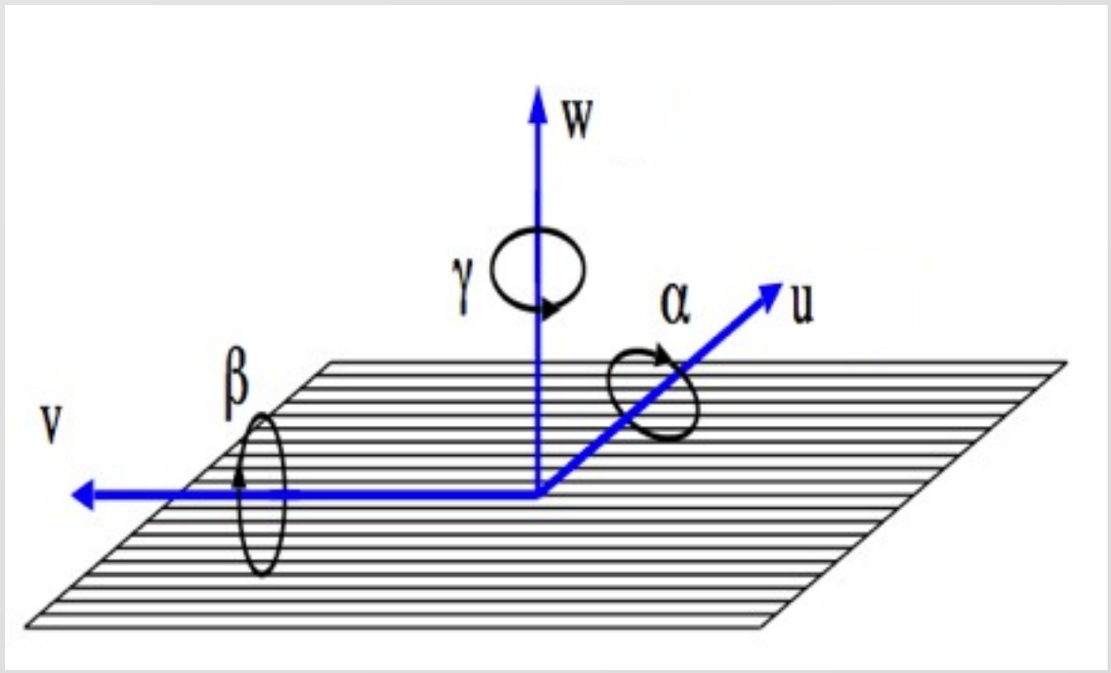
- HelicalTrackHits
  - combine silicon tracker hits from adjacent layers into 3d space-points (10 2d hits $\Longrightarrow$ 5 3d hits / track)
- SeedTracker/HelicalTrackFit
  - take all combinations of hits in first three layers as seeds
  - helix fit as combo of an xz-circle fit and sy-linear fit
  - if seed passes sanity checks, add downstream layers
  - require track has hit in all “5” layers, pass chi^2 and (loose) pointing cuts
- Multiple scattering handled by blowing up errors for downstream hits...not correlated

Track-based Alignment

- offline alignment of tracker planes will be vital to get the vertex resolution we need
- we should have plenty of tracks to do this...
  - use 2-and 3-track tridents to simultaneously determine the beam position (and size if ~ resolution)

problem comes down to solving large system of linear equations with constraints...number of ways to do this and some of these have been implemented in MILLEPEDE II

  - this program was used for CMS and Alice tracker alignment
  - optimized for speed and is parallelized



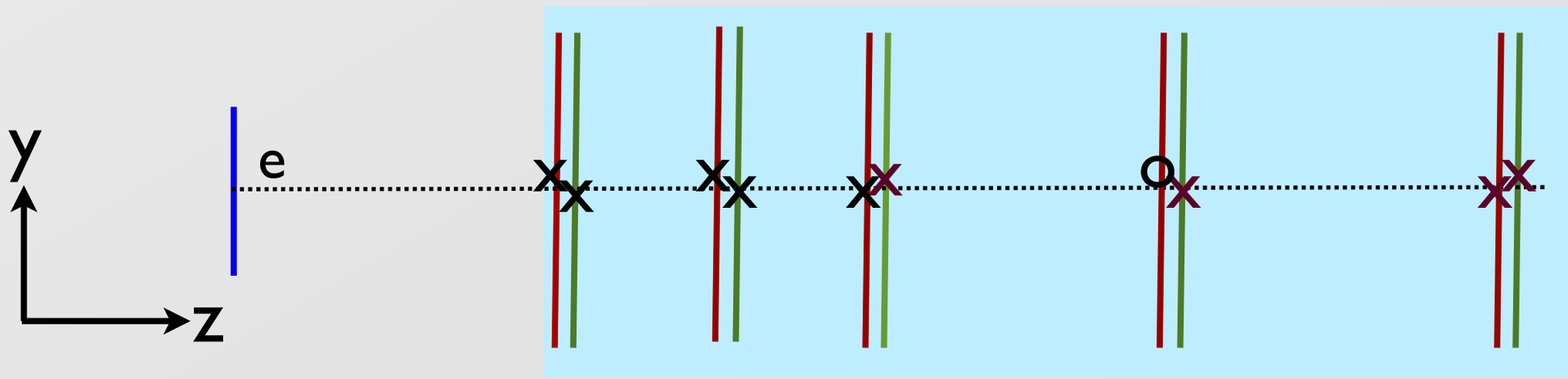
in general, each sensor has 6 alignment parameters ... some are probably insensitive (v, α,β?)

constraints between sensors and planes due to the rigid structure of the tracker also decrease number of parameters

Track-based Alignment

- As I said, we've had trouble getting this to work...
- Maybe we should re-think the approach?
  - start with alignment of axial layers (measure non-bend, tracks linear)?
  - align three-hit sets by hand?
  - write our own, simplified and specific, alignment code?
  - use toy MC to develop and test...go to realistic MC & test run data only after toy is understood?
- A volunteer to take on this task would be great (as I have also said already)!
- This is number one priority...

Tracking finding optimization (efficiency)



the current way we use seed tracker/helical track fitter requires 3d space points

→ if any of the first 10 Si layers are inefficient, we loose tracks

→ ideally, we'd build our tracks from the 2d strip hits directly...but this will require some work

→ helical track fitter does allow for 2d hits (strip_hits)...do I fully trust it to work out of the box? Not too much; needs to be checked

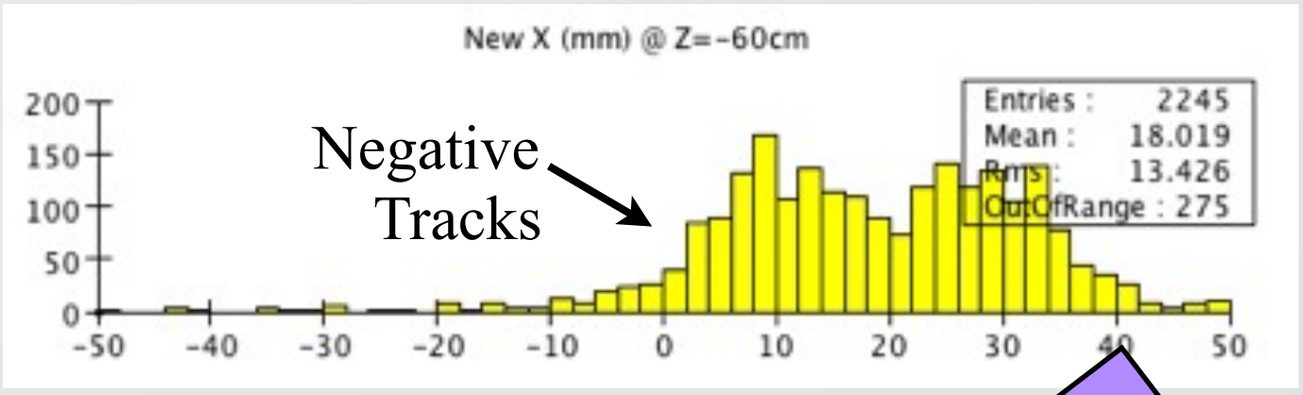
Tracking finding optimization (speed)

- Tracking is the most time consuming part of reconstruction...
- Track fitting is actually fairly fast (the "fit" is analytical...non-iterative)
- What takes the time is the combinatorics...because of ghosting, number of space points/layer is N_{hits}^2, track finding goes as N!
- a couple of ideas:
  - apply filters that (efficiently) rejects bad hit combinations early
  - use axial layers to find combinations consistent with (linear) track....only use those hits to make stereo pair/find helical tracks
  - sort hits in a way that allows us to cut off iteration early
  - some of these ideas would also be good for track-based L3 trigger

Hit times in tracking fit

- The SVT hits have a resolution of ~8ns...we should use this information in the track fitting
- Currently we simply make a cut on the hit time (relative to trigger) and reject all out of time hits
- It should be fairly simple to add the time of the hits in the chi^2 definition of the tracks...
- how useful this is, if this is the best way to use this information, etc...needs to be studied

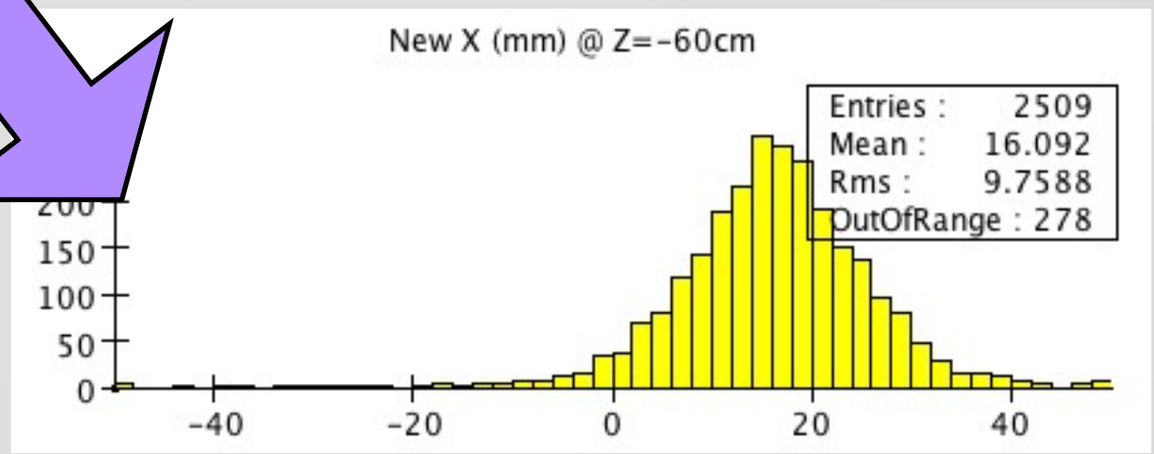
Using real B-field in vertexing



...for test run, this was a huge effect.

In full run, target will sit where B ~ 1/2 B_{FULL} ...will need to account for this. First pass:

Add fringe field



- swim "tracks" (positions & covariace matrices), through the varying B-field, to the approximate vertex position
- in region near vertex, assume B-field constant...can use current Billoir vertexer
- check with MC, which includes real B-field
- if it doesn't perform well enough, do something better.

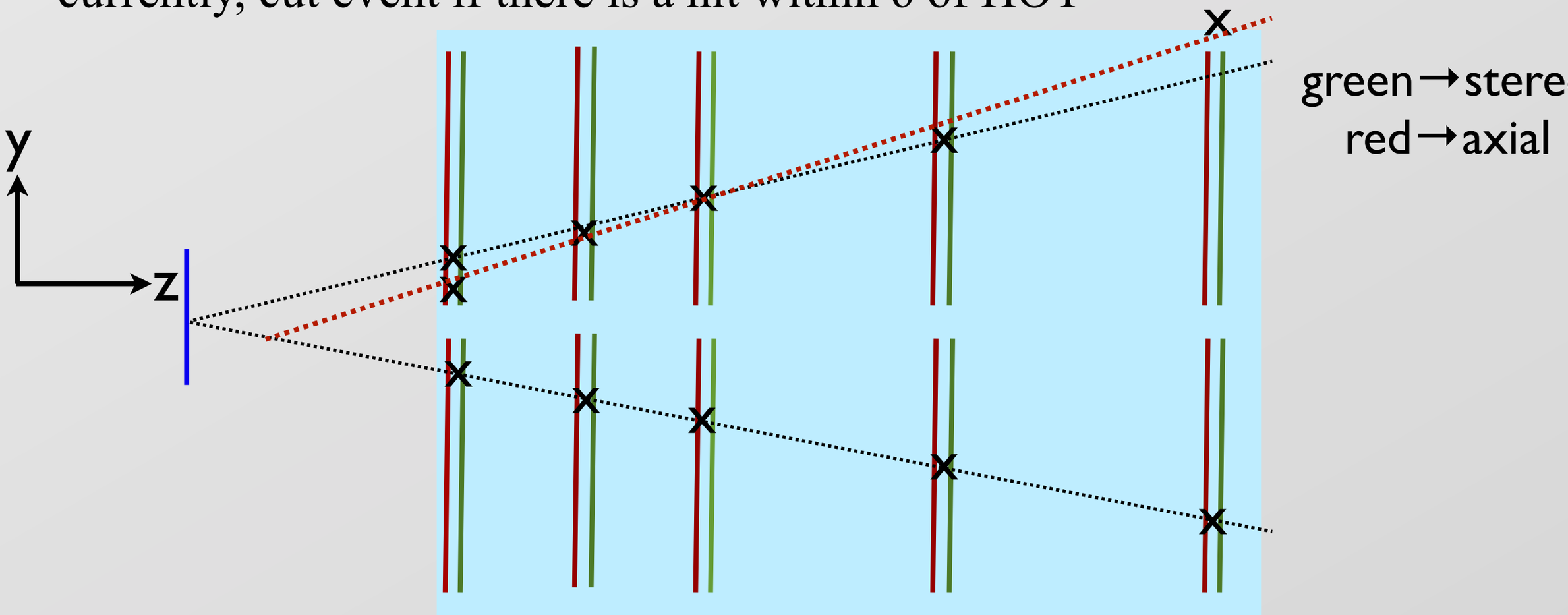
Do we need to modify tracking for displaced vertex search?

Miss-assigned hits: hit not associated with the true electron is included in track...typically this hit is closer to beam, pulling vertex to +z

→currently, tracks can only have 1 overlapping hit...otherwise, best chi^2 taken

→good to save both tracks for further analysis; maybe cut event if there is a decent track where vertex is near target

***currently, cut event if there is a hit within δ of HOT



small←←←hit errors→→→big

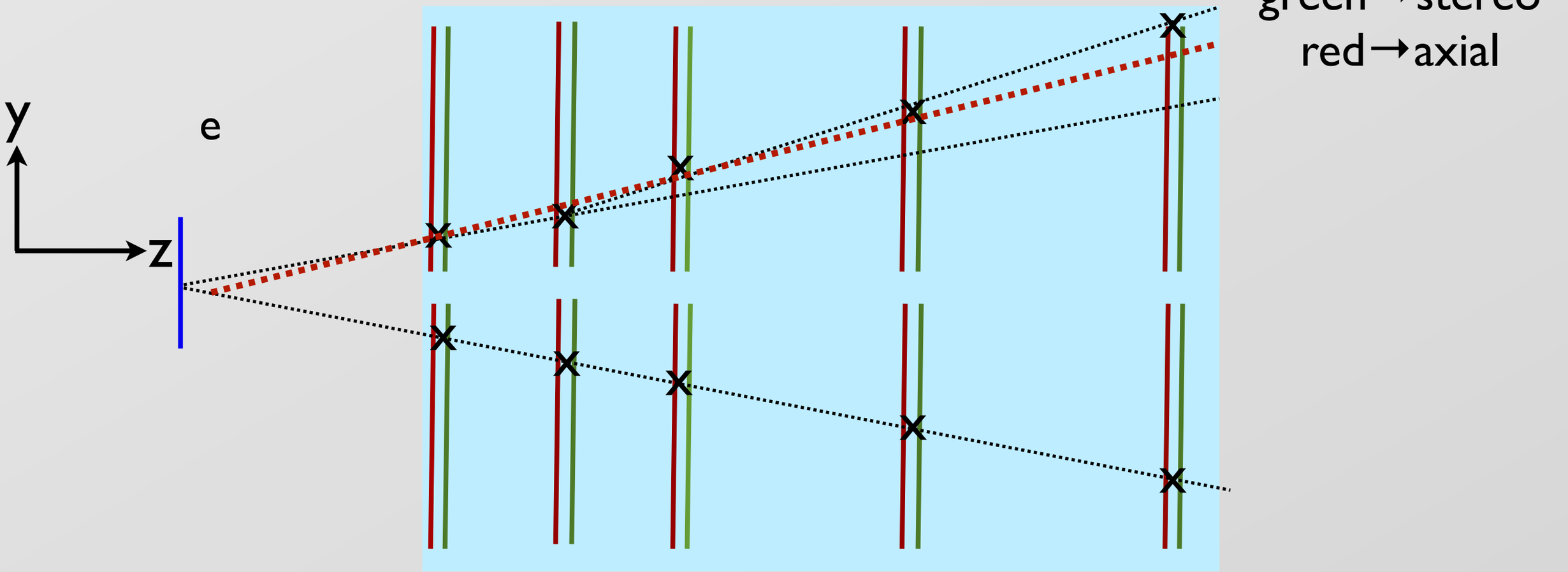
Do we need separate tracking for displaced vertex search?

Hard scatter in first layers: if there is a scatter that kicks track to larger angle, pulls vertex to +z

→ much trickier than mishits! (on the upside this doesn't increase with lumi)

→ can look for slope changes between pairs of hits..needs to be studied more

→ fitter that includes process noise should deal with this well



Dealing with MS Correctly

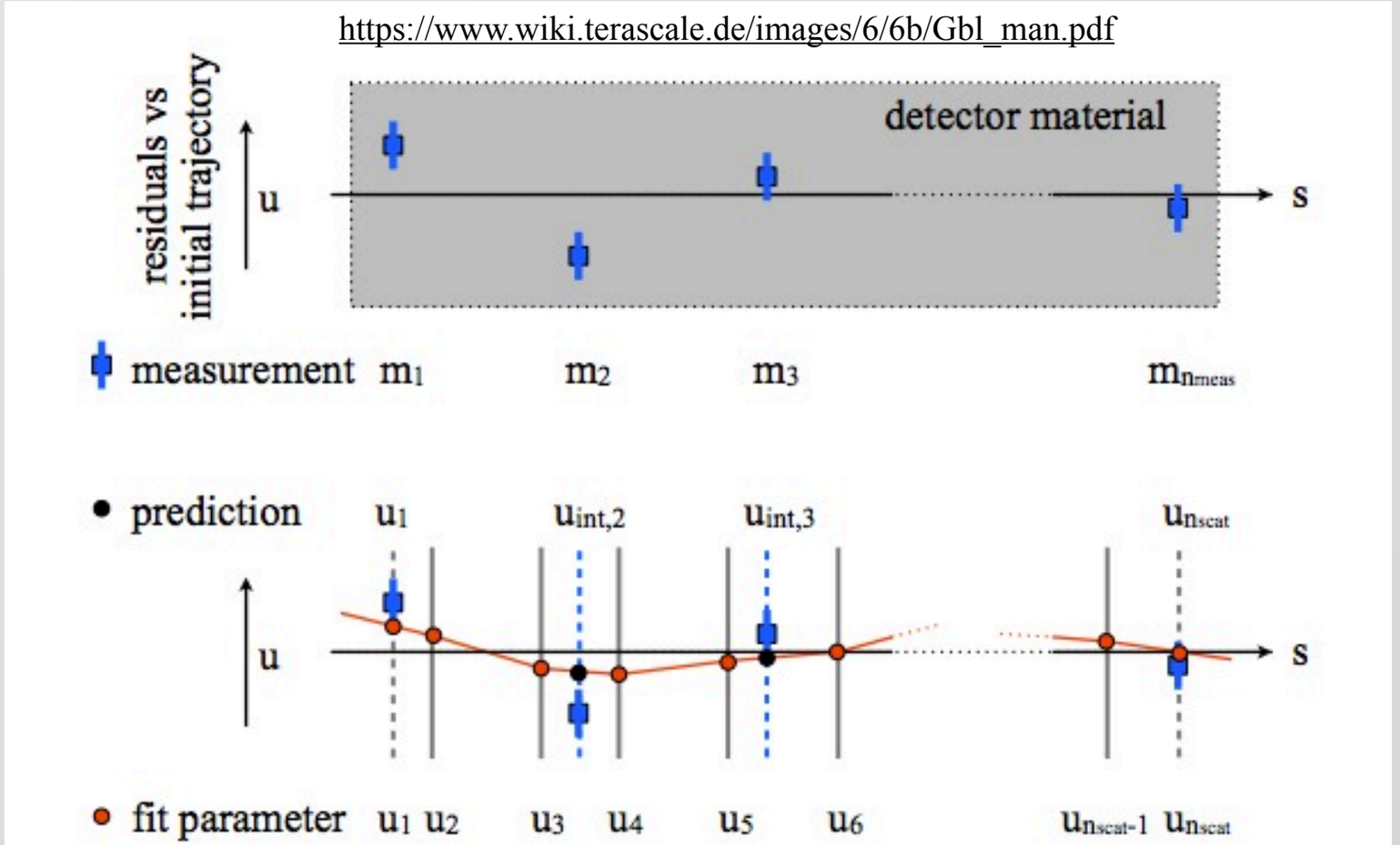
Issue on previous slide wouldn't be as bad if we had track fitter that dealt with MS scattering as process noise ... Kalman Filters (used forever) or Generalized Broken Line Fits (used more recently on some expts)

Kalman Filter: included in TRF; need to get it working in our geometry

GBL: We'd have to roll our own...there is an implementation in C++ available

My Estimate: a dedicated person ~1-2 months to get this working/tested

https://www.wiki.terascale.de/images/6/6b/Gbl_man.pdf



Should we do this before alignment?

Should we write our own tracking code?

- I've suggested many changes we'd like to do in tracking...some of them require some modifications to SeedTracker & HelicalTrackFitter
- Should we scrap these are write our own finder/ fitter?
  - this is not as big a job as it sounds...we can still use many of the same classes, including linear and circular fitters
    - may make it easier to incorporate Kalman/GBE
  - still a big job...
- So....maybe!

What do we need ready on day 1?

- Some sort of track finding & fitting code
  - this we have right now! (efficiency, speed improvements would be nice)
- track-based alignment
  - number one priority
- Everything I'd say would be "nice to have" but not crucial for day 1 running...but will be important for optimizing our physics reach
- We really need manpower...this stuff is very interesting, useful, transferable to other applications & experiments