New Detector Addition and First Results: FTPC and Active Pixel

Mike Miller, Kai, Fabrice, Maria Mora Corral

Outline

1. New detectors in ITTF: a how to.
2. Integration of Pixel detector
3. Integration of FTPC
   a) Big picture issues
   b) Details
   c) First FTPC results
   d) News from the front: Maria's quotes
4. Conclusions

An Integration How To

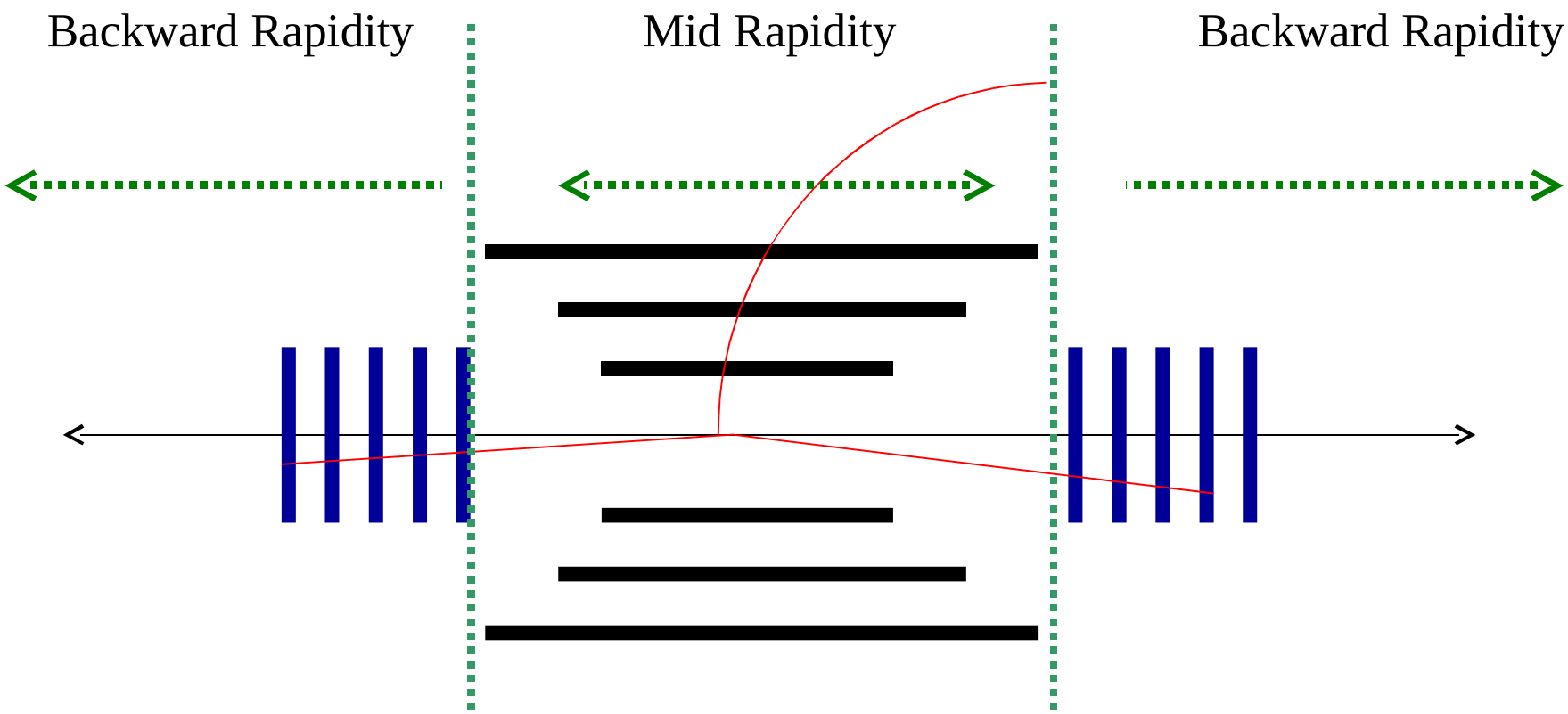
- What's expected:
  - Derive new directory StRoot/StXXXDetectorGroup
- What's for free:
  - Inherit from StiDetectorGroup
  - Base classes take care of c++ specifics
- Code you have to write:
  - Build shapes
  - Build detector objects
  - Load Hits

What Happens “Underneath”?

- StiXXXDetectorBuilder.cxx
  - Create c++ objects representing detector material and location.
  - Add these to internal container (handled by a call to base class add(StiDetector*)
  - StiDetectorTreeBuilder organizes these objects into a unique, extremely sorted tree structure.
  - This “sort” relies on:
    1. Region: kBackwardRapidity, kMidRapidity, kForwardRapidity
    2. Radial position
    3. Azimuth

“sort” happens only once/reco and defines “rules” of navigation

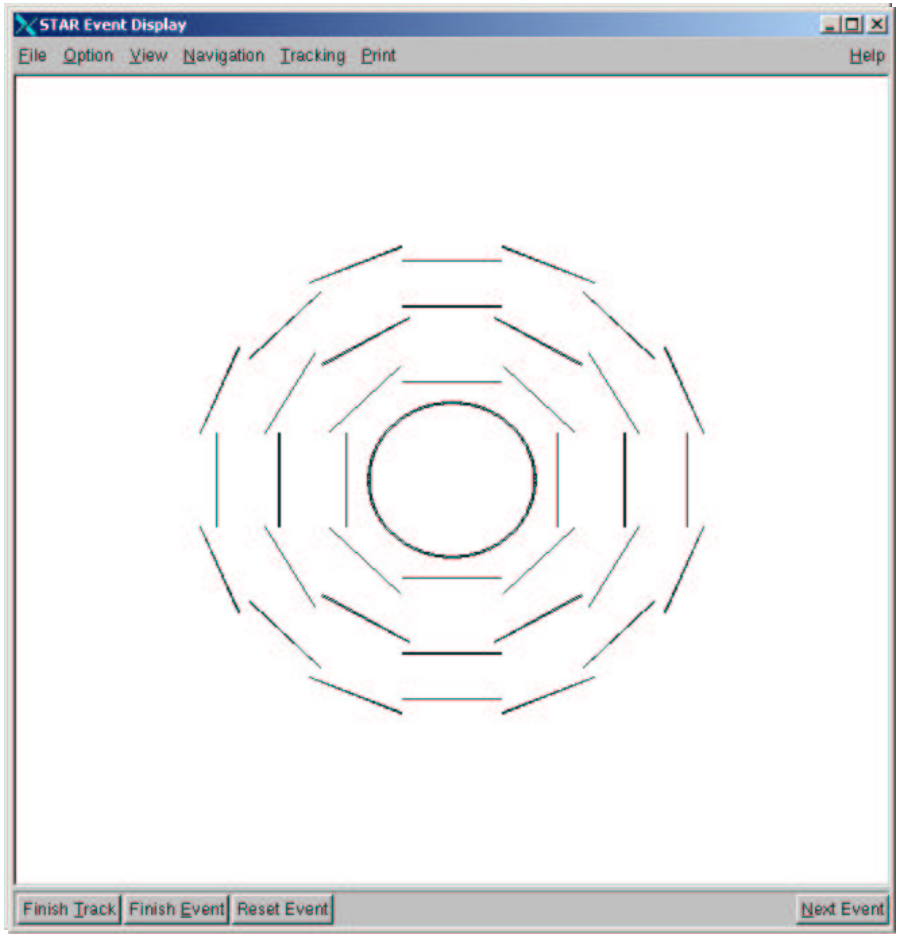
Why do we care about the sort?



Navigation Choices: In/Out, +Phi/-Phi, +Region/-Region

Pixel Integration

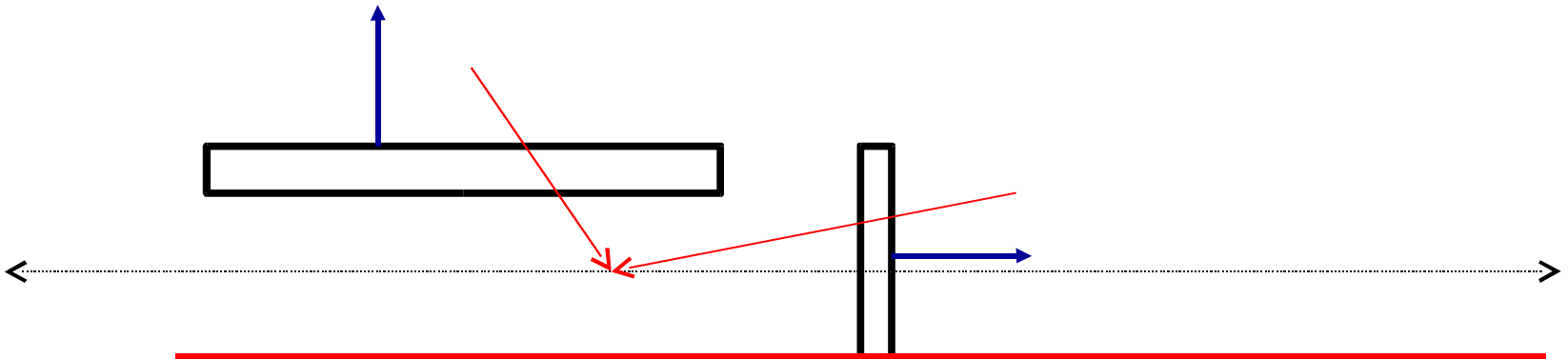
- Mid Rapidity region already existed
- Currently 1 cylindrical layer split into 12 arc-lengths.
  - Material properly built, sorted, and navigable
- Hits:
  - StEvent doesn't include Pixel
  - Local StMcEvent version does (Fabrice)
  - No tracking possible w/o hits!
- Tracking will start when the hits (and hit errors) show up



Pixel integration is easy, natural

Why Isn't FTPC so Natural?

1. Forward/Backward sorting was not yet fully implemented, tested
2. FTPC tracking requires a "new" seed-finder (you don't start in the TPC!)
3. FTPC tracking requires a modification of track-volume intersection routines



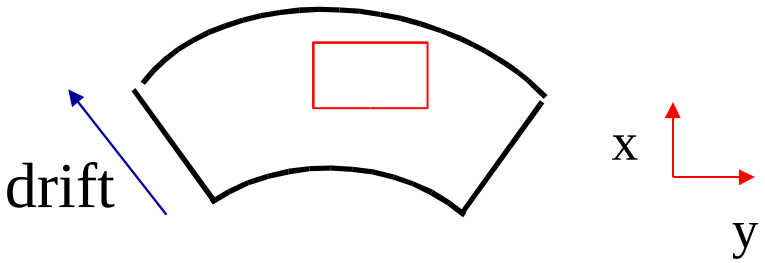
Need to cope with different outward pointing normal directions

What has been done

- Activated, tested, debugged different regions
  - Had to touch both Detector-model and Hit-container
- Created StiFbLocalTrackSeedFinder
  - Start outside-in tracking in FTPCE, then FTPCW
  - Copied most of the logic from TPC track finder, changed starting point
  - Generalized intersection routines to handle FTPC
  - Perform fast circle-line fit and pass off to Kalman
- Intersection routines in Kalman Tracker: t.b.d.
- Keeping the error matrix diagonal in Kalman Tracker: t.b.d.

Hit Sorting, Retrieval

- Hit Sorting:
  - Sort groups of hits by sector, padrow.
  - Within group, sort first by “distance along pad”, then by “radius” (y, then x)
  - Should be phi and radius to minimize bias…



- Retrieval
  - Define window in (y,x) = (deltaD, deltaZ)
  - Names are relic of TPC design bias

Pattern Recognition

Current Algorithm

- Two Point Seed
  - Two closest points on neighboring FTPC planes
- Extension
  - Straight line extrapolation to next plane.
  - Choose point closest to intersection

Planned Algorithm

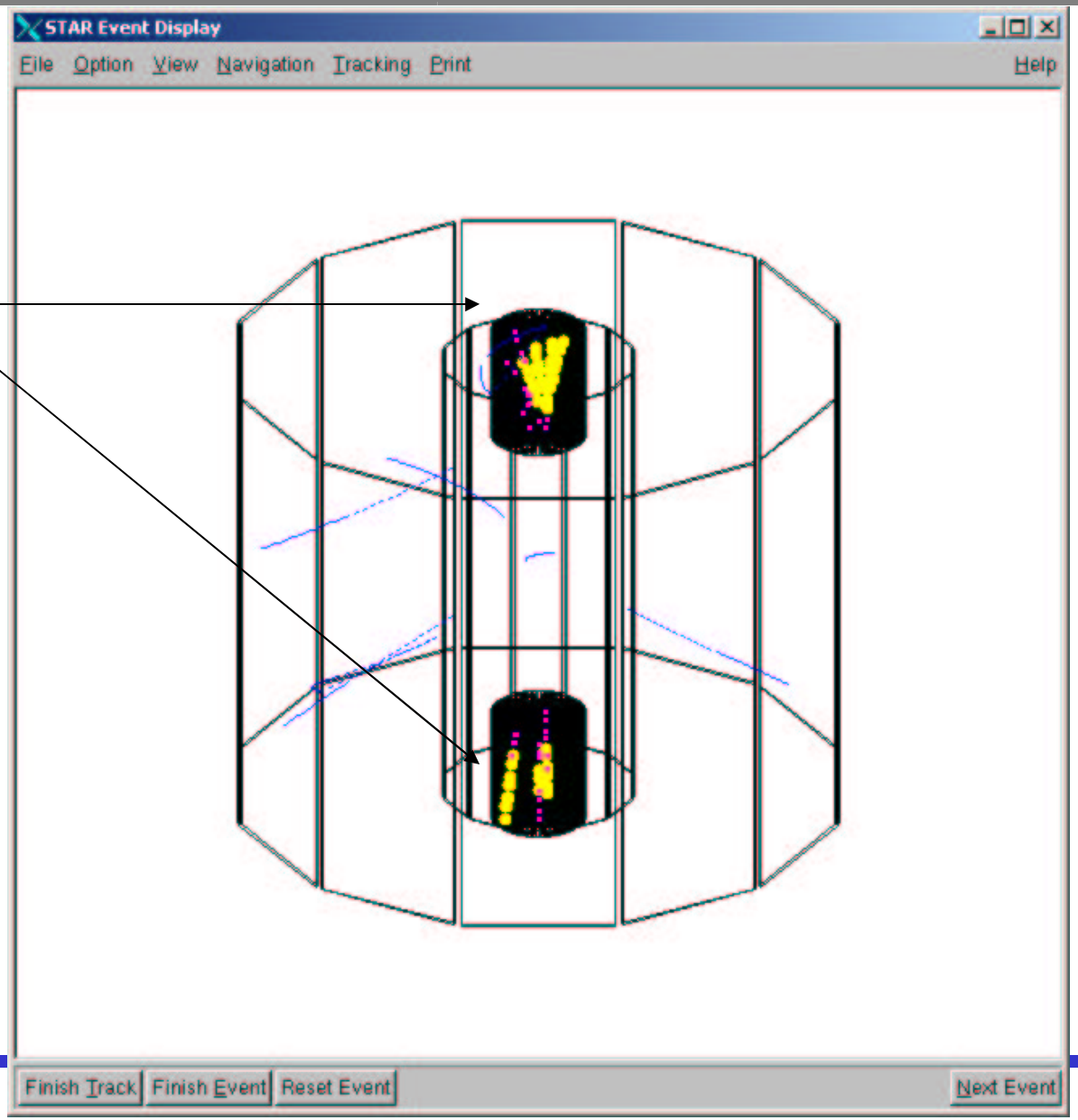
- Two Point Seed
  - –Straight line through point, primary vertex/beamline
  - –Choose point closest to intersection of this line and next FTPC plane
- Extension
  - –Probably stay with straight line extrapolation to next plane.

Proof Of Principle

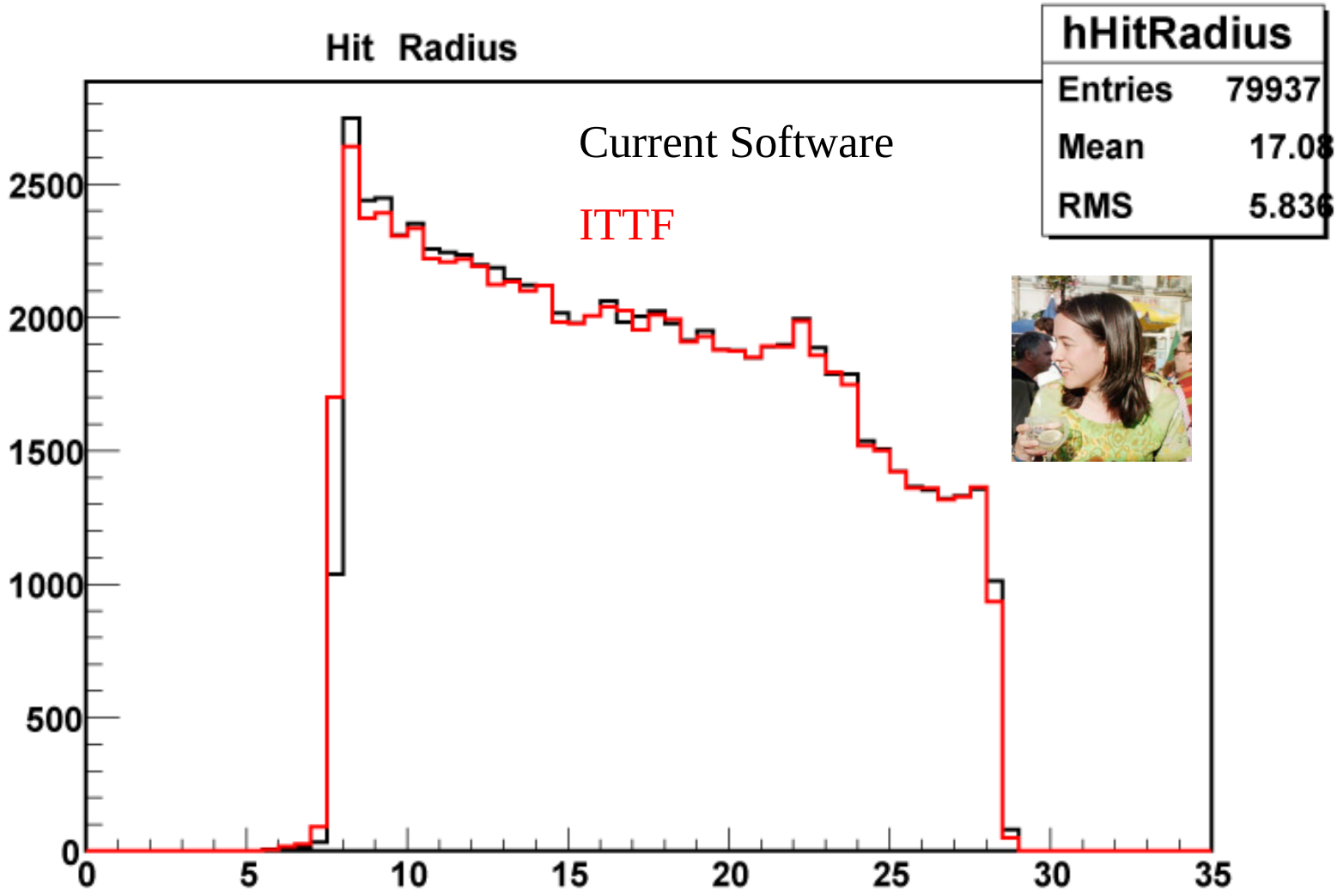
FTPC Volumes

FTPC Hits

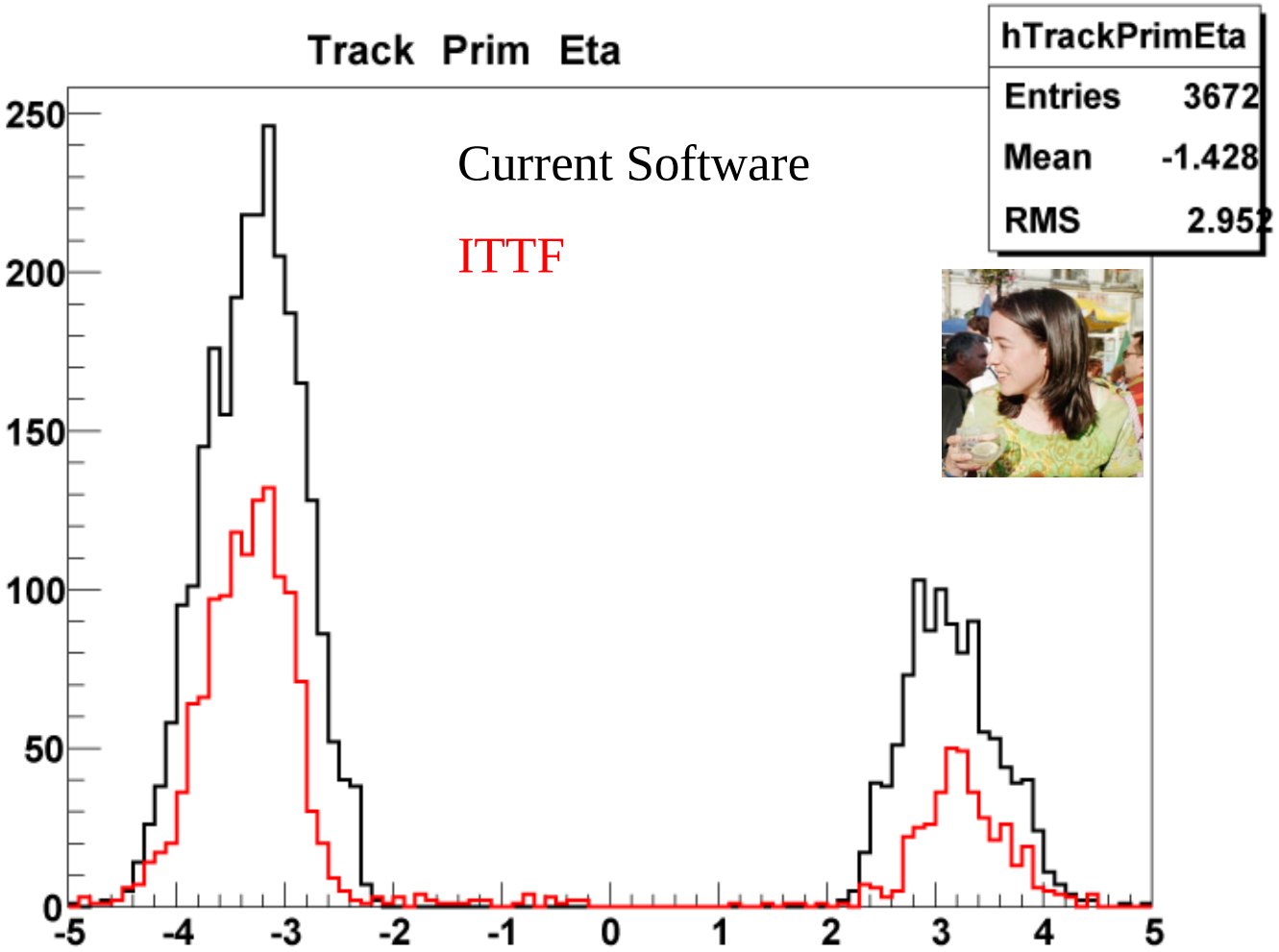
FTPC Hits Assigned to Track



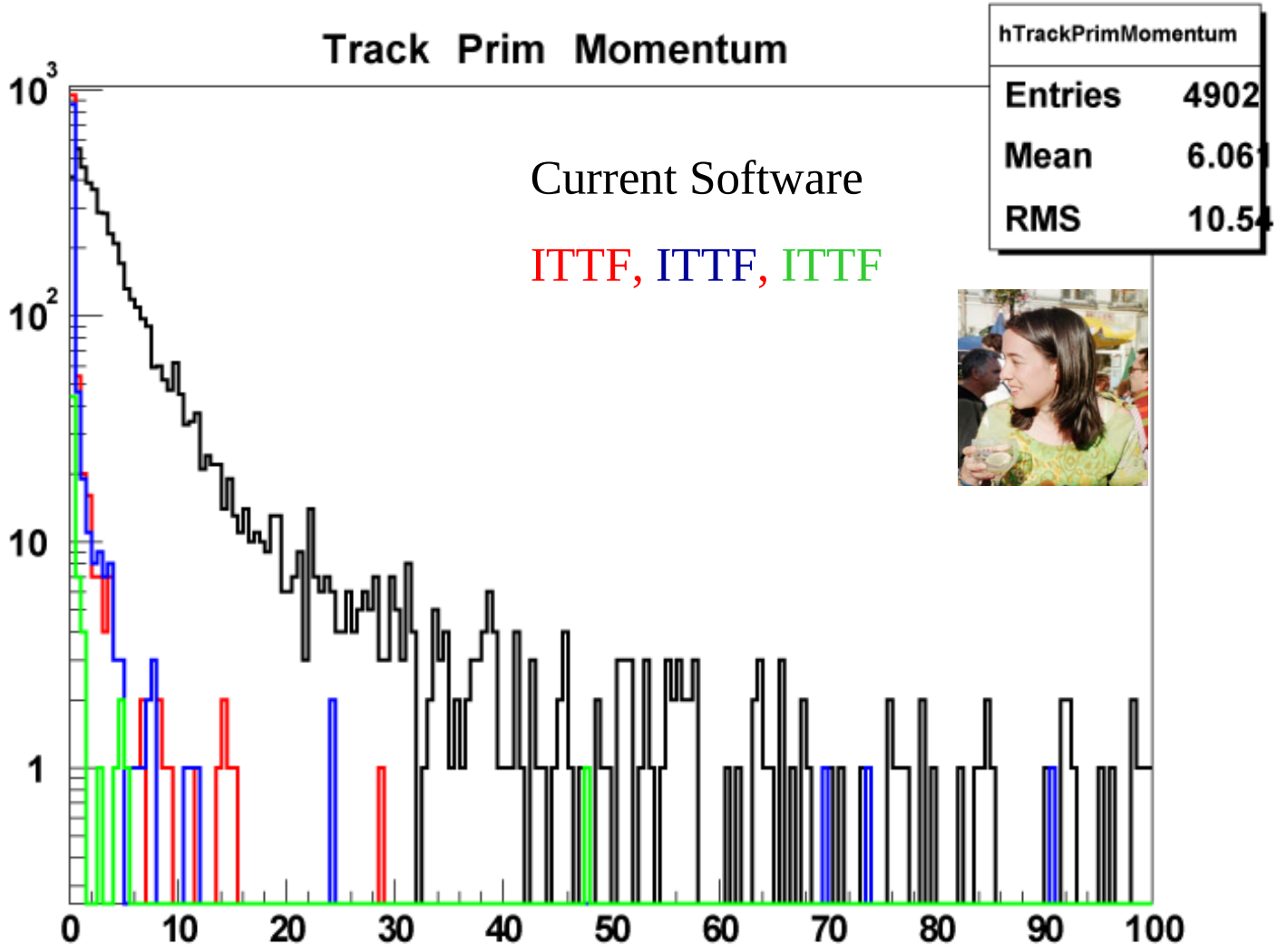
The hits look good…position



And we find some tracks…eta



But not so good…p_T



Comments From Maria

- “Personally, I would say that the code is flexible and easy to understand (only need C++ knowledge).”
- “My only personal problem was trying to understand the different kind of coordinates used for placing the StiShapes, this was not straight forward and a better description should be provided to the users.”

Questions From Maria

- “Does the ITTF code allow for the use of different seed finder parameters for every subsystem?” **yes**
- “Does the ITTF code allow for the use of different seed finder algorithms for every subsystem?” **yes**
- “Does the ITTF code allow for the use of different track finder parameters for every subsystem?” **Not yet, but it should**

Requirements From Maria

- "…we need the ability of vertex finding using only FTPC tracks."

yes

- B-Field inhomogeneous: "The IITF code must be able to include in his fit code a correction for this modification."

Not yet, but it should

FTPC: Two Main Issues

1. Use real track finding logic.
   a) Use tricks from current FTPC code
   b) Restrict searches to beam line candidates
   c) Move from line to circle extrapolation
2. Need to feed these tracks through Kalman
   a) Modify intersection routines
   b) Uncover any subtle assumptions about tracks being from mid rapidity

Conclusions

1. Detector Integration is now well defined
2. Integration of Pixel (or any other mid-rapidity detector) is natural.
3. Integration of FTPC work in progress.
4. Once FTPC integration is debugged, BEMC, FPD, etc… will become “natural” tasks.