Hall A Analysis Software Status

Ole Hansen

Jefferson Lab

Transversity Collaboration Meeting
July 21, 2008

Podd 1.5

- Expanded database capabilities (arrays, strings)
- Extended detector maps (reference channels)
- Restructured helicity classes
- Support for ROOT 5.18, 5.20, Fedora 9 (gcc 4.3)
- Bugfixes (split runs, formulas, output)
- Speed improvements (output)

Well-working “beta” version in CVS now, on Web shortly

Podd 1.5

- Expanded database capabilities (arrays, strings)
- Extended detector maps (reference channels)
- Restructured helicity classes
- Support for ROOT 5.18, 5.20, Fedora 9 (gcc 4.3)
- Bugfixes (split runs, formulas, output)
- Speed improvements (output)

Well-working “beta” version in CVS now, on Web shortly

Podd 1.5

- Expanded database capabilities (arrays, strings)
- Extended detector maps (reference channels)
- Restructured helicity classes
- Support for ROOT 5.18, 5.20, Fedora 9 (gcc 4.3)
- Bugfixes (split runs, formulas, output)
- Speed improvements (output)

Well-working “beta” version in CVS now, on Web shortly

Podd 1.5

- Expanded database capabilities (arrays, strings)
- Extended detector maps (reference channels)
- Restructured helicity classes
- Support for ROOT 5.18, 5.20, Fedora 9 (gcc 4.3)
- Bugfixes (split runs, formulas, output)
- Speed improvements (output)

Well-working “beta” version in CVS now, on Web shortly

Podd 1.5

- Expanded database capabilities (arrays, strings)
- Extended detector maps (reference channels)
- Restructured helicity classes
- Support for ROOT 5.18, 5.20, Fedora 9 (gcc 4.3)
- Bugfixes (split runs, formulas, output)
- Speed improvements (output)

Well-working “beta” version in CVS now, on Web shortly

Podd 1.5

- Expanded database capabilities (arrays, strings)
- Extended detector maps (reference channels)
- Restructured helicity classes
- Support for ROOT 5.18, 5.20, Fedora 9 (gcc 4.3)
- Bugfixes (split runs, formulas, output)
- Speed improvements (output)

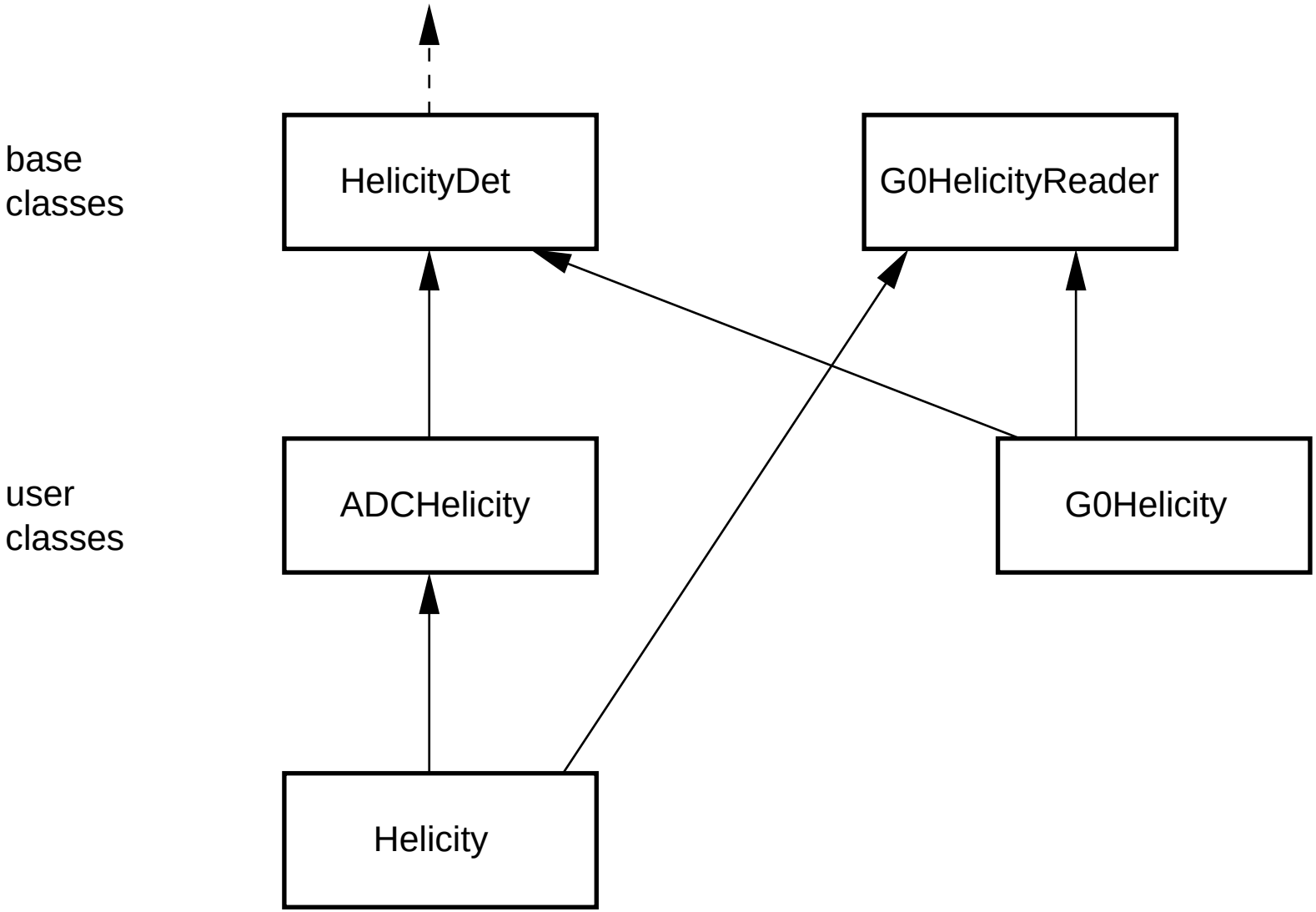
Well-working “beta” version in CVS now, on Web shortly

Podd 1.5

- Expanded database capabilities (arrays, strings)
- Extended detector maps (reference channels)
- Restructured helicity classes
- Support for ROOT 5.18, 5.20, Fedora 9 (gcc 4.3)
- Bugfixes (split runs, formulas, output)
- Speed improvements (output)

Well-working “beta” version in CVS now, on Web shortly

New Helicity Class Hierarchy



Split Runs

Split Run Example

```
THaRun* r1 = new THaRun( "/data1/e01001_1000.dat.0" );
THaRun* r2 = new THaRun( "/data2/e01001_1000.dat.1" );
analyzer->Process( r1 );
analyzer->Process( r2 );
```

- Directories must be same or end with `dataN`
- Old/alternative methods still work
- Could be futher improved (single object for group of runs)

Unfinshed in Podd 1.5

- Global beam helicity (`fEvtHdr.fHelicity`)
- Incorporate Bob's scaler and NormAna updates
- Add patches for ROOT 5.20 and gcc 4.3

Unfinshed in Podd 1.5

- Global beam helicity (`fEvtHdr.fHelicity`)
- Incorporate Bob's scaler and NormAna updates
- Add patches for ROOT 5.20 and gcc 4.3

Unfinshed in Podd 1.5

- Global beam helicity (`fEvtHdr.fHelicity`)
- Incorporate Bob's scaler and NormAna updates
- Add patches for ROOT 5.20 and gcc 4.3

BigBite Tracking: Tree Search Algorithm

- Suggested by Dell'orso *et al.*, NIM, 1990
- Recursive template matching
- Fast and efficient (speed and memory)
- Suitable for BigBite
  - simple geometry
  - field-free tracking region
- Proven at HERMES with chambers similar to BigBite's

BigBite Tracking: Tree Search Algorithm

- Suggested by Dell'orso *et al.*, NIM, 1990
- Recursive template matching
- Fast and efficient (speed and memory)
- Suitable for BigBite
  - simple geometry
  - field-free tracking region
- Proven at HERMES with chambers similar to BigBite's

BigBite Tracking: Tree Search Algorithm

- Suggested by Dell'orso *et al.*, NIM, 1990
- Recursive template matching
- Fast and efficient (speed and memory)
- Suitable for BigBite
  - simple geometry
  - field-free tracking region
- Proven at HERMES with chambers similar to BigBite's

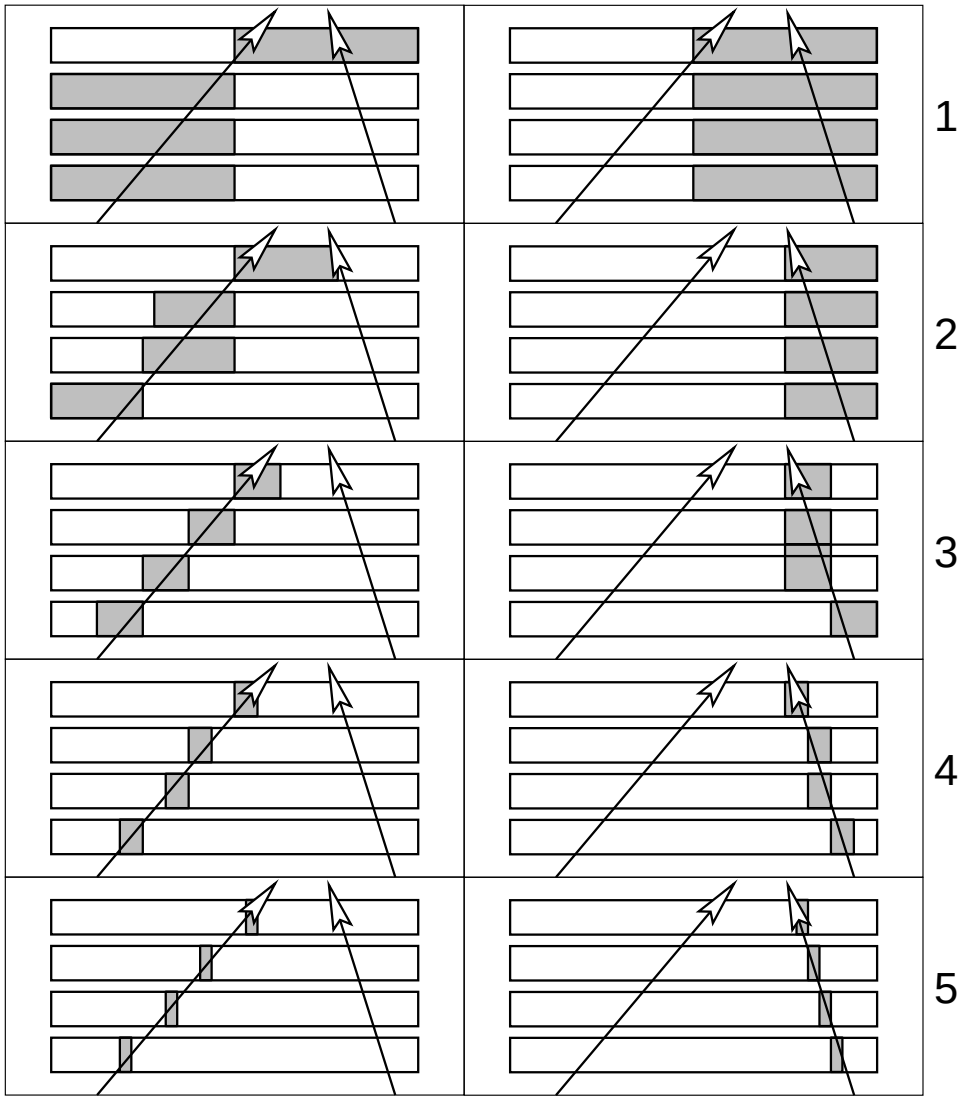
BigBite Tracking: Tree Search Algorithm

- Suggested by Dell'orso *et al.*, NIM, 1990
- Recursive template matching
- Fast and efficient (speed and memory)
- Suitable for BigBite
  - simple geometry
  - field-free tracking region
- Proven at HERMES with chambers similar to BigBite's

BigBite Tracking: Tree Search Algorithm

- Suggested by Dell'orso *et al.*, NIM, 1990
- Recursive template matching
- Fast and efficient (speed and memory)
- Suitable for BigBite
  - simple geometry
  - field-free tracking region
- Proven at HERMES with chambers similar to BigBite's

Successive Approximation Method



Key Advantages

- Tree-like structure allows fast template lookup ($\mathcal{O}(\log N_{\text{bins}})$)
- Symmetry considerations allow efficient template storage ($\mathcal{O}(1\text{MB})$): only "base patterns" stored

Key Advantages

- Tree-like structure allows fast template lookup ($\mathcal{O}(\log N_{\text{bins}})$)
- Symmetry considerations allow efficient template storage ($\mathcal{O}(1\text{MB})$): only "base patterns" stored

Finished Since Last Meeting

- 2D pattern "de-cloning" algorithm ($\rightarrow$ "roads")
- Track fitting within roads
- Combination of roads in 3D
- 3D track fitting
- 3D track de-cloning/de-ghosting algorithm
- Testing & debugging with online E04-007 data
- Adapted to new event display
- 2D processing parallelized
- Profiling & speed optimization ($\times 2$)

Finished Since Last Meeting

- 2D pattern "de-cloning" algorithm ($\rightarrow$ "roads")
- Track fitting within roads
- Combination of roads in 3D
- 3D track fitting
- 3D track de-cloning/de-ghosting algorithm
- Testing & debugging with online E04-007 data
- Adapted to new event display
- 2D processing parallelized
- Profiling & speed optimization ($\times 2$)

Finished Since Last Meeting

- 2D pattern "de-cloning" algorithm ($\rightarrow$ "roads")
- Track fitting within roads
- Combination of roads in 3D
- 3D track fitting
- 3D track de-cloning/de-ghosting algorithm
- Testing & debugging with online E04-007 data
- Adapted to new event display
- 2D processing parallelized
- Profiling & speed optimization ($\times 2$)

Finished Since Last Meeting

- 2D pattern "de-cloning" algorithm ($\rightarrow$ "roads")
- Track fitting within roads
- Combination of roads in 3D
- 3D track fitting
- 3D track de-cloning/de-ghosting algorithm
- Testing & debugging with online E04-007 data
- Adapted to new event display
- 2D processing parallelized
- Profiling & speed optimization ($\times 2$)

Finished Since Last Meeting

- 2D pattern "de-cloning" algorithm ($\rightarrow$ "roads")
- Track fitting within roads
- Combination of roads in 3D
- 3D track fitting
- 3D track de-cloning/de-ghosting algorithm
- Testing & debugging with online E04-007 data
- Adapted to new event display
- 2D processing parallelized
- Profiling & speed optimization ($\times 2$)

Finished Since Last Meeting

- 2D pattern "de-cloning" algorithm ($\rightarrow$ "roads")
- Track fitting within roads
- Combination of roads in 3D
- 3D track fitting
- 3D track de-cloning/de-ghosting algorithm
- Testing & debugging with online E04-007 data
- Adapted to new event display
- 2D processing parallelized
- Profiling & speed optimization ($\times 2$)

Finished Since Last Meeting

- 2D pattern "de-cloning" algorithm ($\rightarrow$ "roads")
- Track fitting within roads
- Combination of roads in 3D
- 3D track fitting
- 3D track de-cloning/de-ghosting algorithm
- Testing & debugging with online E04-007 data
- Adapted to new event display
- 2D processing parallelized
- Profiling & speed optimization ($\times 2$)

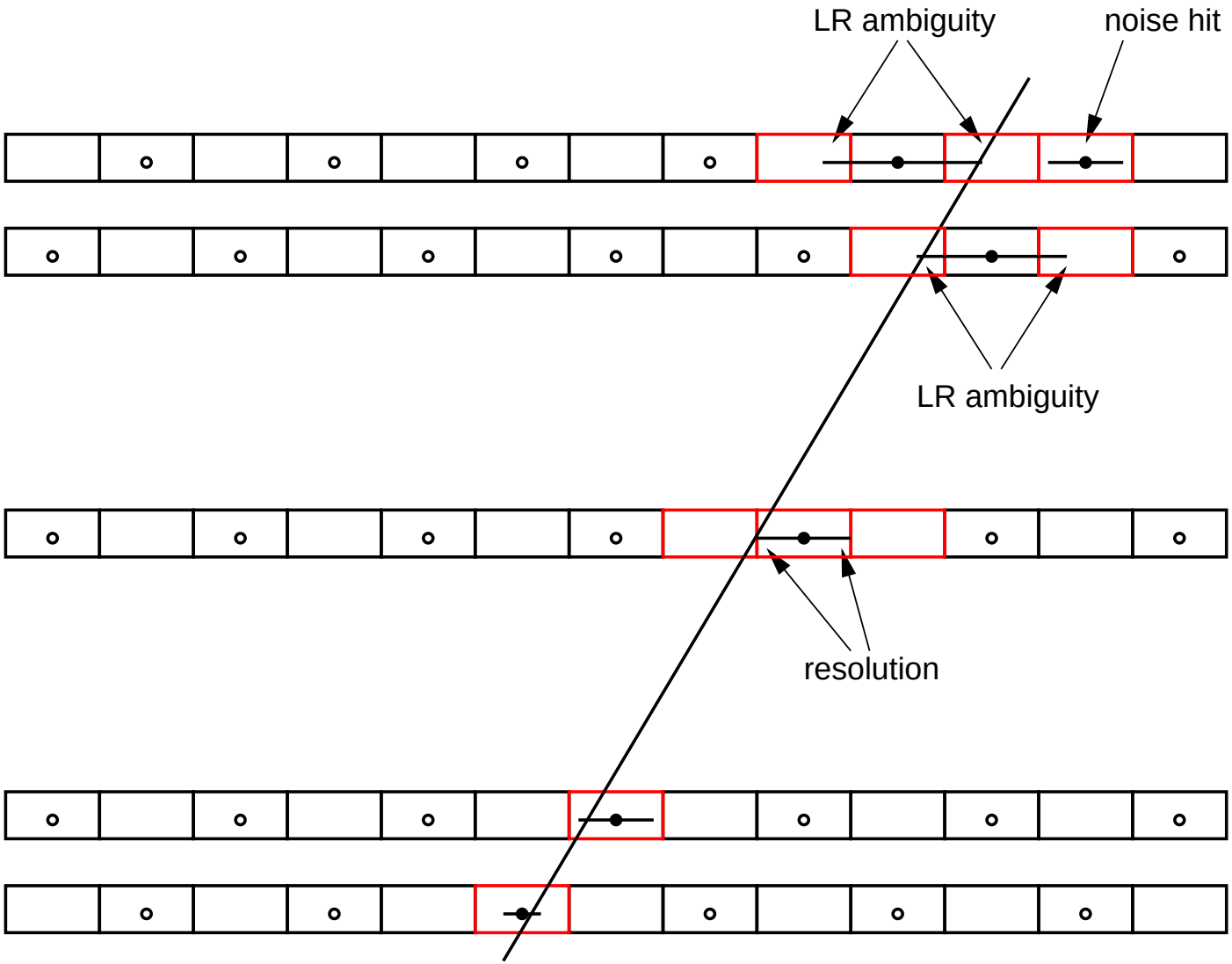
Finished Since Last Meeting

- 2D pattern "de-cloning" algorithm ($\rightarrow$ "roads")
- Track fitting within roads
- Combination of roads in 3D
- 3D track fitting
- 3D track de-cloning/de-ghosting algorithm
- Testing & debugging with online E04-007 data
- Adapted to new event display
- 2D processing parallelized
- Profiling & speed optimization ($\times 2$)

Finished Since Last Meeting

- 2D pattern "de-cloning" algorithm ($\rightarrow$ "roads")
- Track fitting within roads
- Combination of roads in 3D
- 3D track fitting
- 3D track de-cloning/de-ghosting algorithm
- Testing & debugging with online E04-007 data
- Adapted to new event display
- 2D processing parallelized
- Profiling & speed optimization ($\times 2$)

Pattern Cloning (multiple patterns for single track)



2D De-Cloning Algorithm

- "Clustering" problem. Not entirely straightforward . . .
- Many clone patterns may occur $\rightarrow$ time-critical algorithm
- Current implementation runs in $\mathcal{O}(N \log N)$ time.
- *Can* be improved: $\approx \mathcal{O}(N \cdot \alpha(N)) \rightarrow$ summer student project

2D De-Cloning Algorithm

- "Clustering" problem. Not entirely straightforward . . .
- Many clone patterns may occur $\longrightarrow$ time-critical algorithm
- Current implementation runs in $\mathcal{O}(N \log N)$ time.
- *Can* be improved: $\approx \mathcal{O}(N \cdot \alpha(N)) \rightarrow$ summer student project

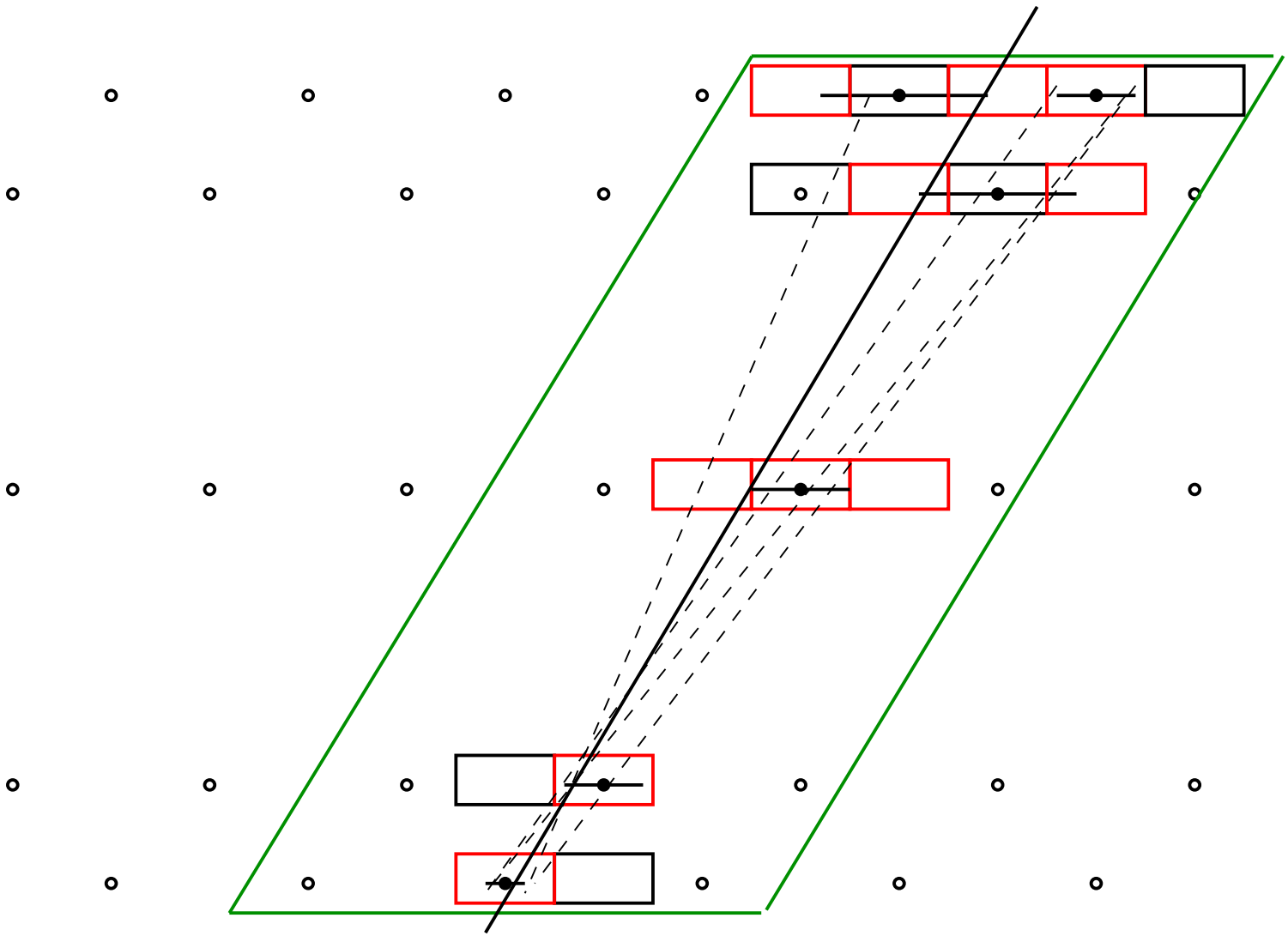
2D De-Cloning Algorithm

- "Clustering" problem. Not entirely straightforward . . .
- Many clone patterns may occur $\longrightarrow$ time-critical algorithm
- Current implementation runs in $\mathcal{O}(N \log N)$ time.
- *Can* be improved: $\approx \mathcal{O}(N \cdot \alpha(N)) \rightarrow$ summer student project

2D De-Cloning Algorithm

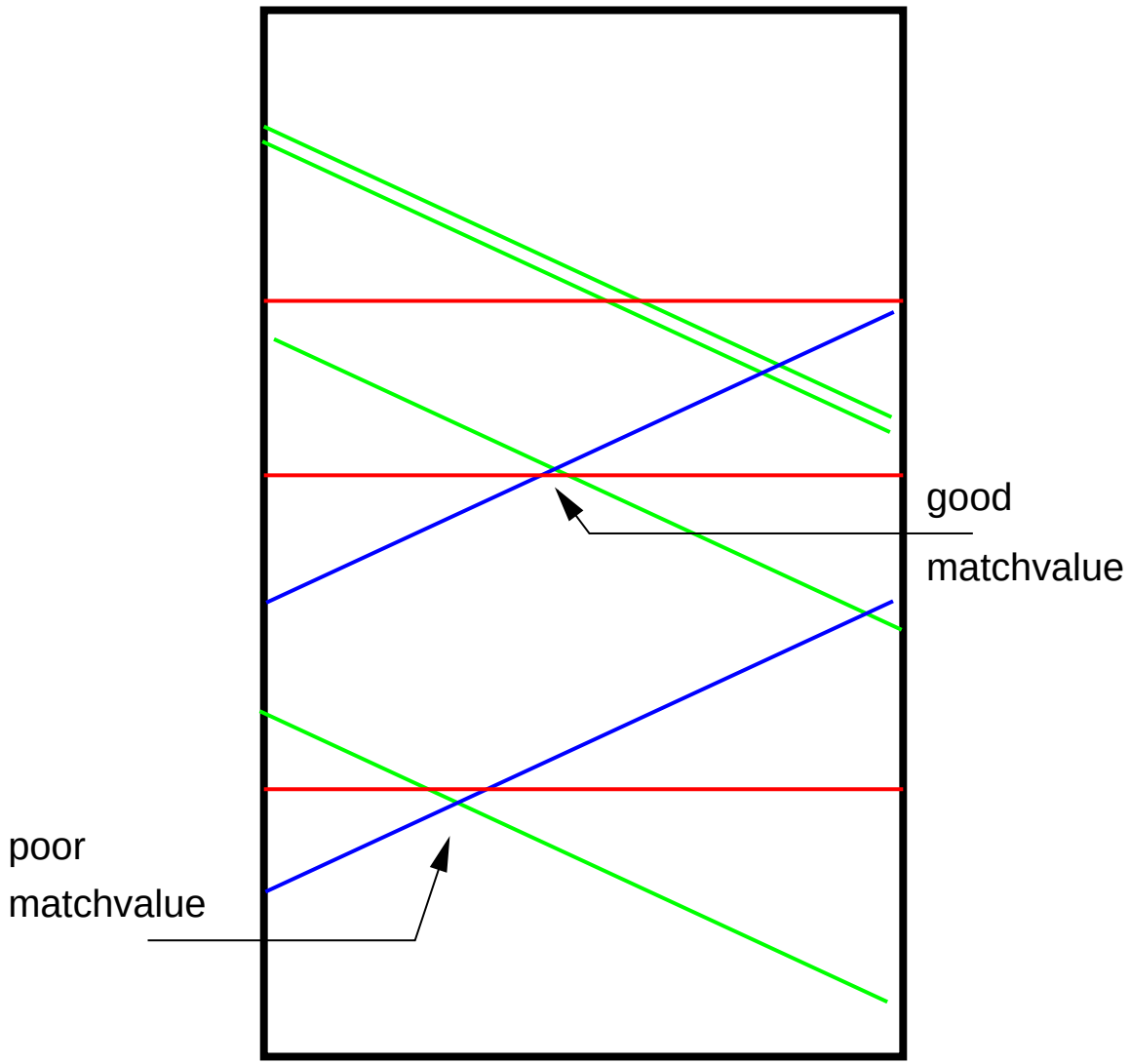
- "Clustering" problem. Not entirely straightforward . . .
- Many clone patterns may occur $\longrightarrow$ time-critical algorithm
- Current implementation runs in $\mathcal{O}(N \log N)$ time.
- *Can* be improved: $\approx \mathcal{O}(N \cdot \alpha(N)) \longrightarrow$ summer student project

2D Track Fitting



Combination in 3D

Chamber front view (x down, y left):



3D Track Fitting

Fit the linear equations

$$\mathbf{A}\hat{\beta} = \mathbf{y}$$

using the coordinates y_i of the best 2D fits in all planes i,

$$y_i = \begin{pmatrix} x + m_x z_i \\ y + m_y z_i \end{pmatrix} \cdot \begin{pmatrix} \cos\alpha_i \\ \sin\alpha_i \end{pmatrix},$$

where x, m_x, y, m_y are the parameters to be fitted (β_k).
The fit is done by Cholesky decomposition of the normal equation

$$(\mathbf{A^T W A})\hat{\beta} = (\mathbf{A^T W})\mathbf{y},$$

where **W** is the weight matrix (cf. ROOT's `TLinearFitter`).

3D De-Cloning/De-Ghosting

- Pick set of 3D tracks with the lowest χ^2 that do not share any roads and hits.
- Fast and effective (nearly complete elimination of obvious clones)
- Clones may survive if 2D clustering incomplete (split clusters); *e.g.* due to crosstalk between wires
- Ghosts may survive if χ^2 cuts too loose or χ^2 too good; *e.g.* due to poor alignment and calibration

Experience with E04-007 data

- 12-plane configuration much slower to analyze than 15 planes. Chambers do not provide enough information to reject arbitrary combinations of front and back hits.
- Speed about 100 Hz with very noisy data on Intel Core2 6600.
- Quite sensitive to quality of geometry information

Experience with E04-007 data

- 12-plane configuration much slower to analyze than 15 planes. Chambers do not provide enough information to reject arbitrary combinations of front and back hits.
- Speed about 100 Hz with very noisy data on Intel Core2 6600.
- Quite sensitive to quality of geometry information

Experience with E04-007 data

- 12-plane configuration much slower to analyze than 15 planes. Chambers do not provide enough information to reject arbitrary combinations of front and back hits.
- Speed about 100 Hz with very noisy data on Intel Core2 6600.
- Quite sensitive to quality of geometry information

Gprof Results (E04-007, noisy data)

Printed by Ole Hansen

Jun 12, 08 0:02 **gprof−example−1k−ev.txt** Page 1/1

```
index % time    self  children    called     name
[1]     88.9    0.77    3.98   34001+19184037 <cycle 1 as a whole> [1]
                0.11    1.42    2948            TreeSearch::Projection::MakeRoads() <cycle 1> [2]
                0.00    1.12    2997            TreeSearch::Projection::Track() <cycle 1> [4]
                0.21    0.06  156394            TreeSearch::Road::Fit() <cycle 1> [21]
                0.07    0.17  156394            TreeSearch::Road::CollectCoordinates() <cycle 1> [24]
                0.02    0.16  570429            TBranch::Fill() <cycle 1> [35]
                0.00    0.18     999            THaAnalyzer::PhysicsAnalysis(int) <cycle 1> [36]
                0.04    0.08 2486050            THaVar::GetObjArrayLenPtr() const <cycle 1> [44]
```

Thursday June 12, 2008 gprof−example−1k−ev.txt 1/1

Multi-Threading of 2D Processing

- Multi-threading of `Projection::Track()` implemented (all 2D processing)
- Standard ROOT threads (pthreads)
- Maximum 3 threads (configurable)
- With noisy E04-007 data, results are disappointing (10-15% gain)
- On some older machines, multi-threaded code actually runs slower than single-threaded code! L2 cache?

Multi-Threading of 2D Processing

- Multi-threading of `Projection::Track()` implemented (all 2D processing)
- Standard ROOT threads (pthreads)
- Maximum 3 threads (configurable)
- With noisy E04-007 data, results are disappointing (10-15% gain)
- On some older machines, multi-threaded code actually runs slower than single-threaded code! L2 cache?

Multi-Threading of 2D Processing

- Multi-threading of `Projection::Track()` implemented (all 2D processing)
- Standard ROOT threads (pthreads)
- Maximum 3 threads (configurable)
- With noisy E04-007 data, results are disappointing (10-15% gain)
- On some older machines, multi-threaded code actually runs slower than single-threaded code! L2 cache?

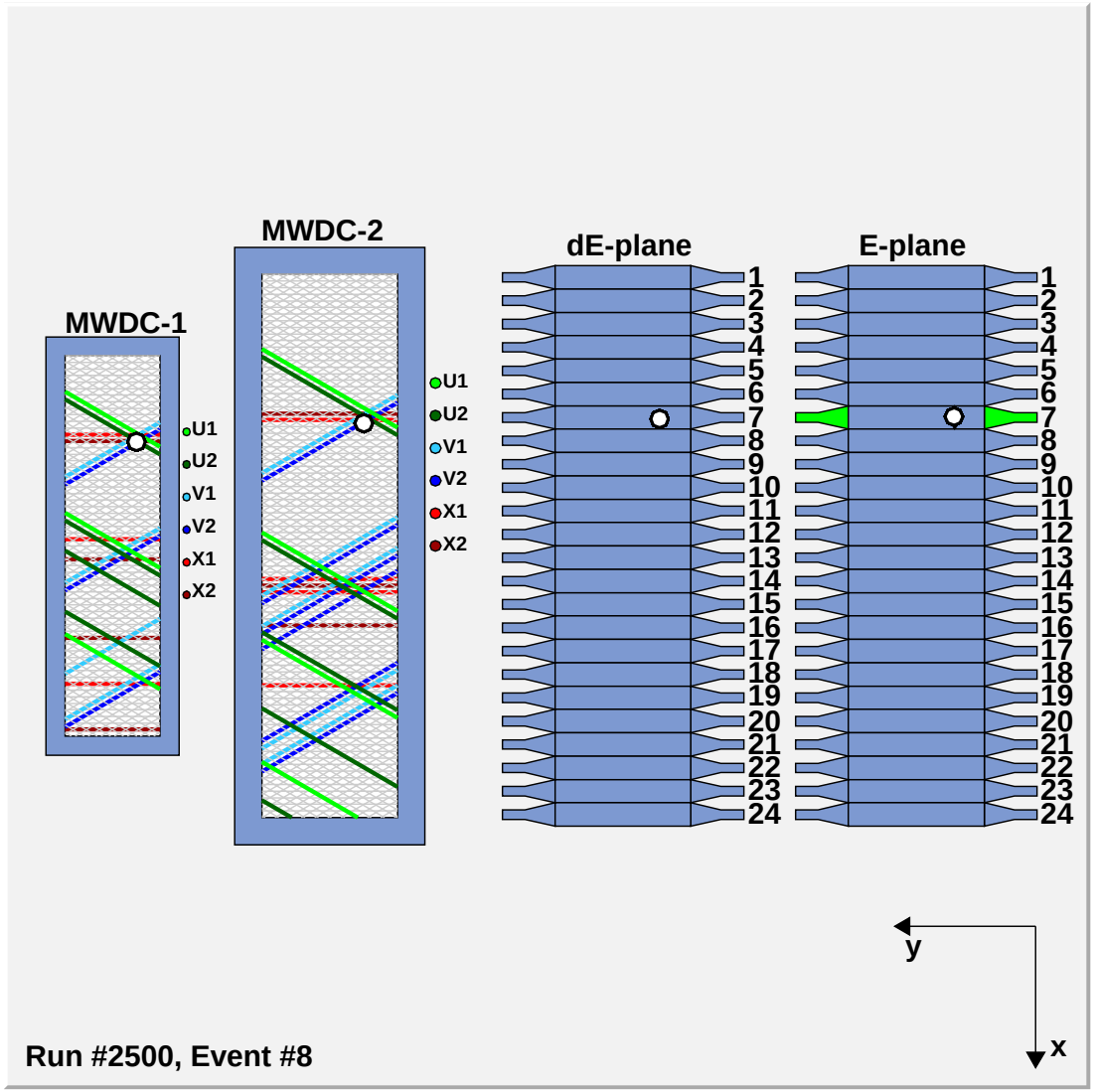
Multi-Threading of 2D Processing

- Multi-threading of `Projection::Track()` implemented (all 2D processing)
- Standard ROOT threads (pthreads)
- Maximum 3 threads (configurable)
- With noisy E04-007 data, results are disappointing (10-15% gain)
- On some older machines, multi-threaded code actually runs slower than single-threaded code! L2 cache?

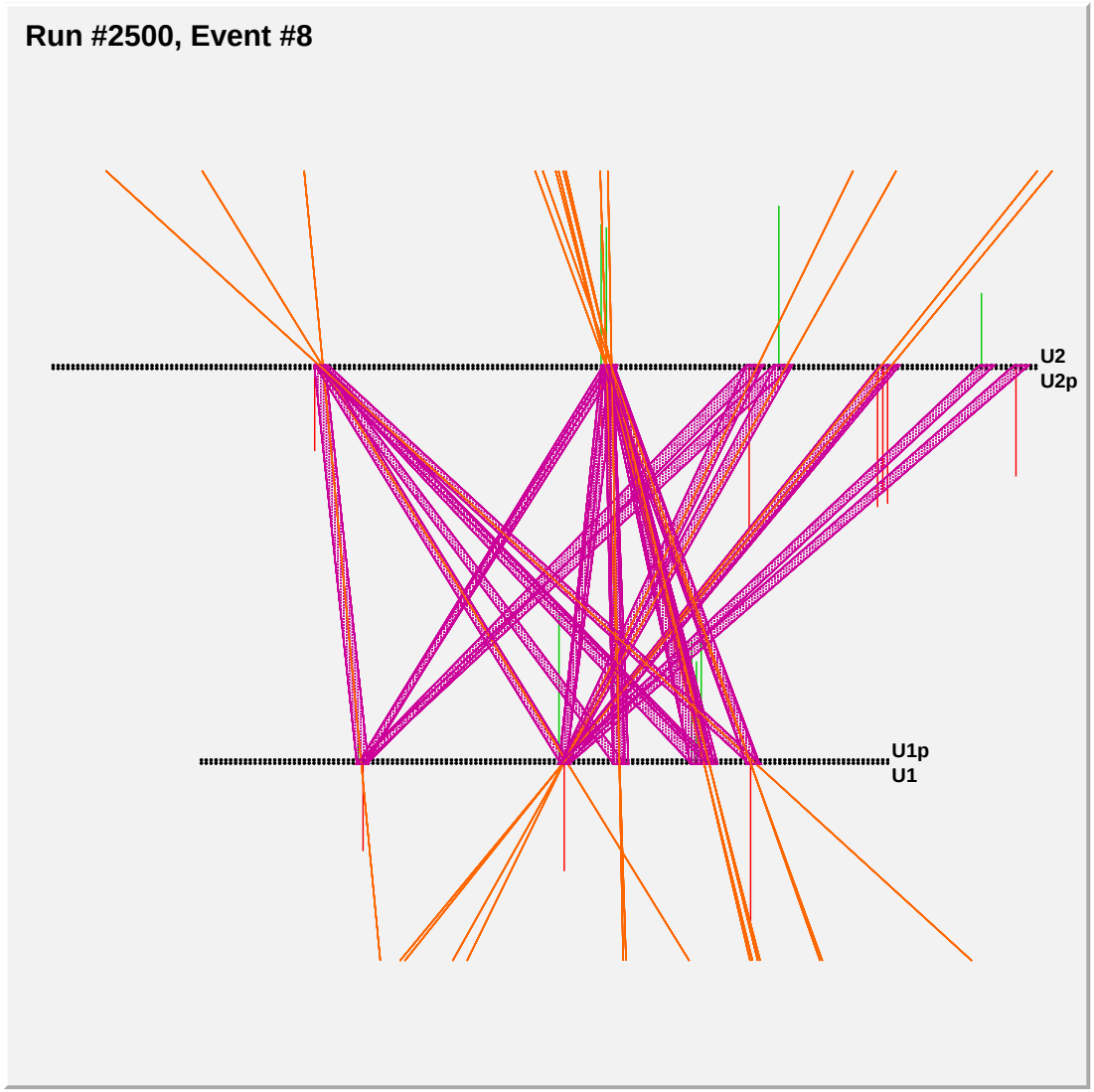
Multi-Threading of 2D Processing

- Multi-threading of `Projection::Track()` implemented (all 2D processing)
- Standard ROOT threads (pthreads)
- Maximum 3 threads (configurable)
- With noisy E04-007 data, results are disappointing (10-15% gain)
- On some older machines, multi-threaded code actually runs slower than single-threaded code! L2 cache?

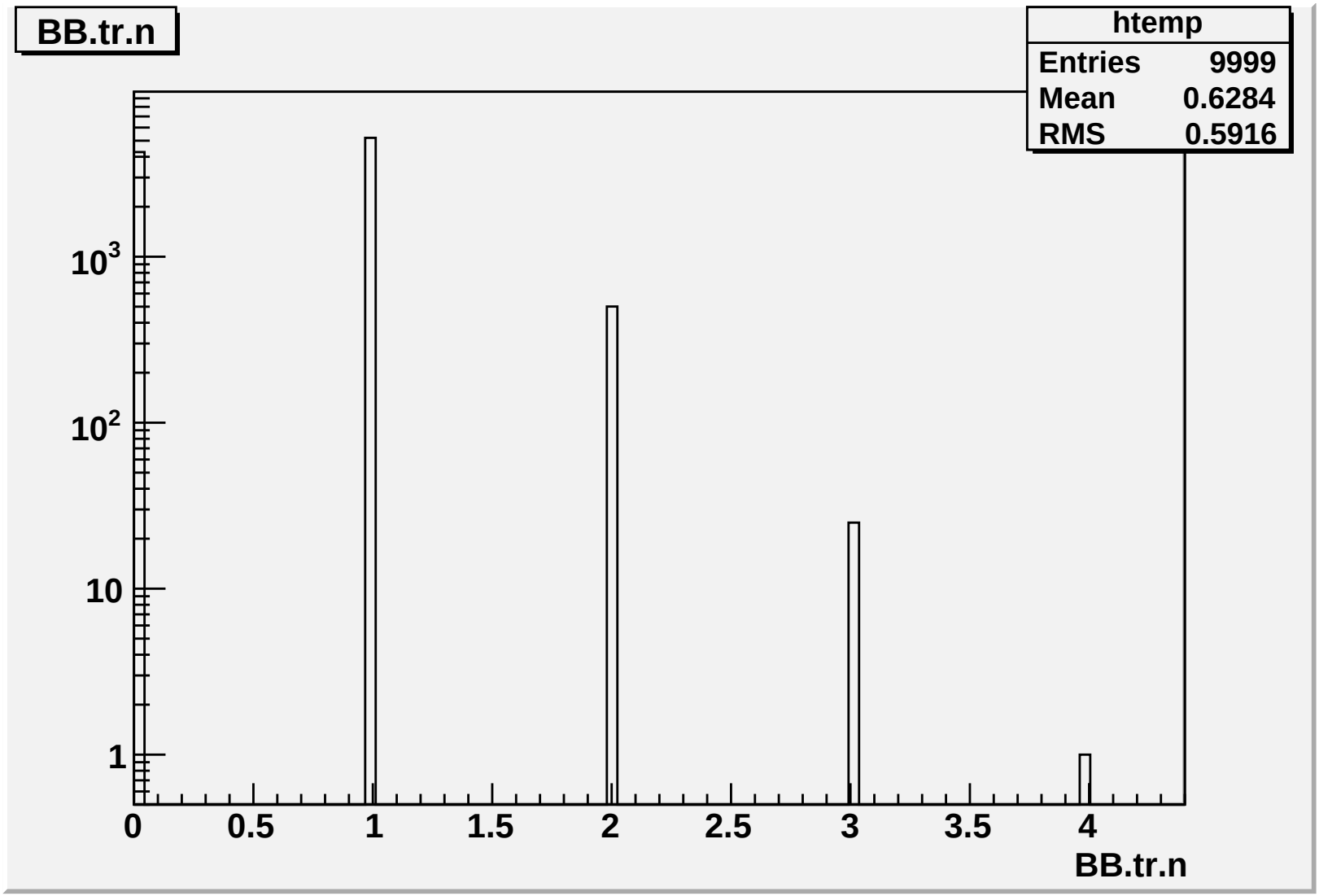
Event Display Planar View (E04-007)



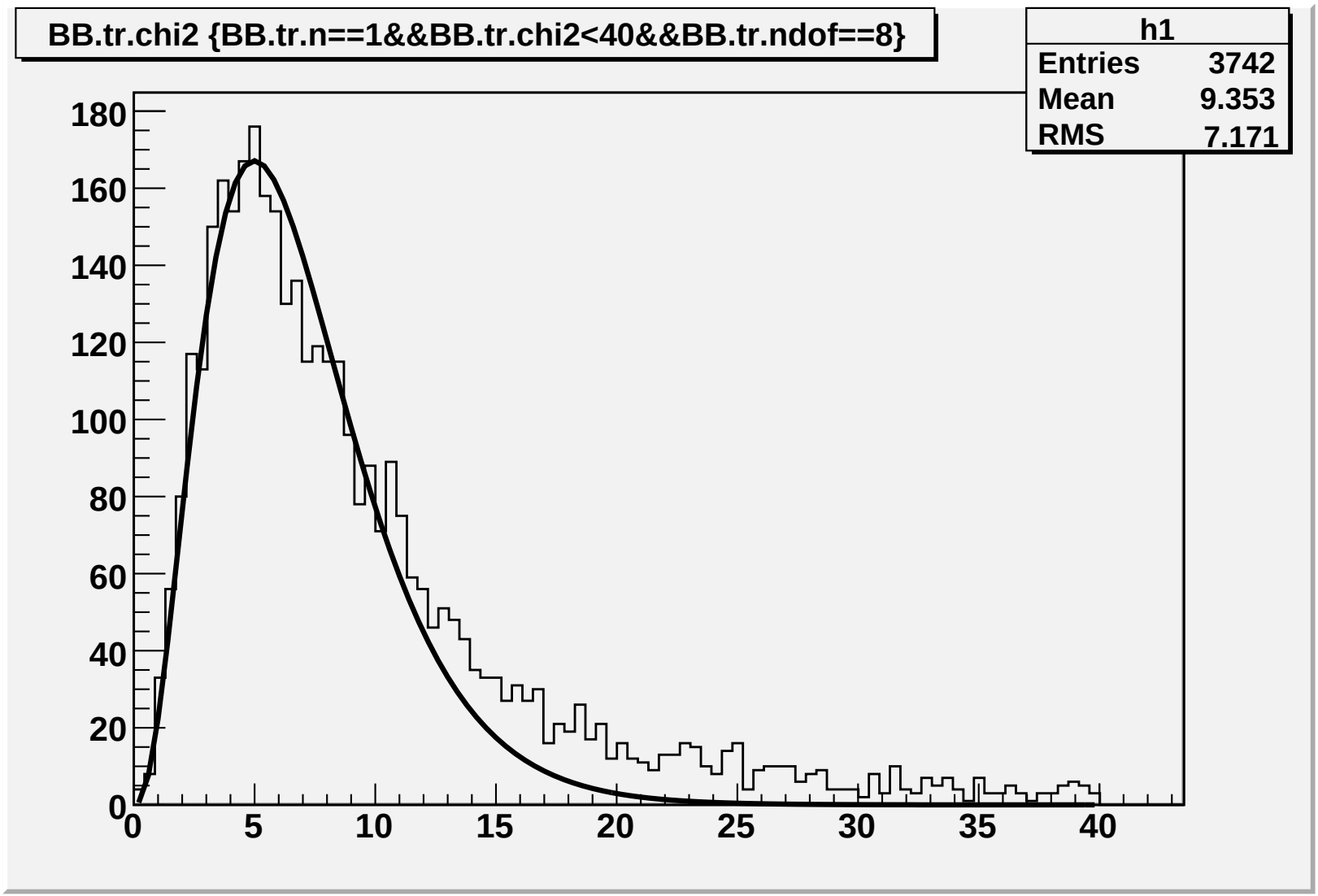
Event Display Projection View (E04-007)



Number of tracks per event (E04-007 run 3574)

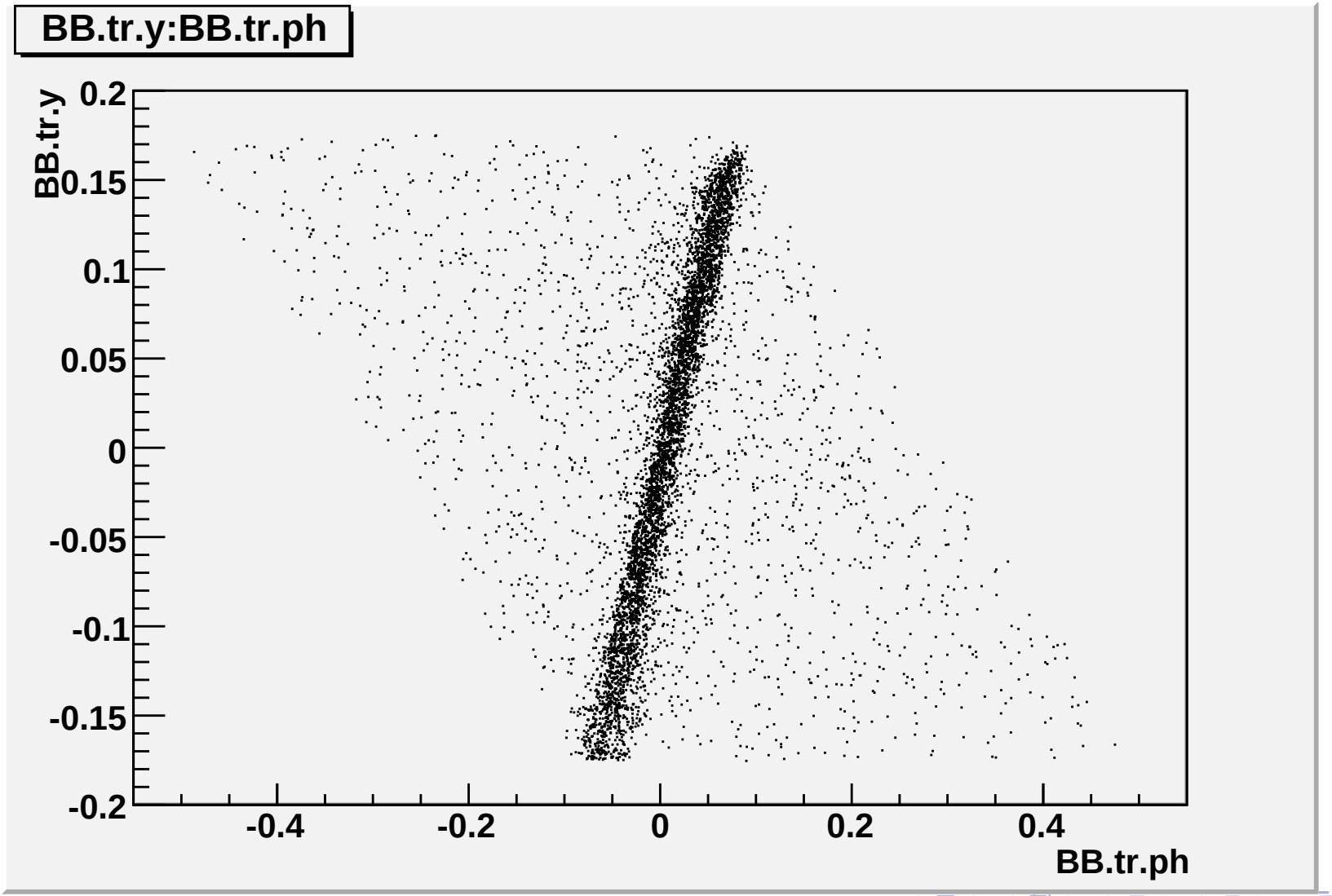


Track χ^2

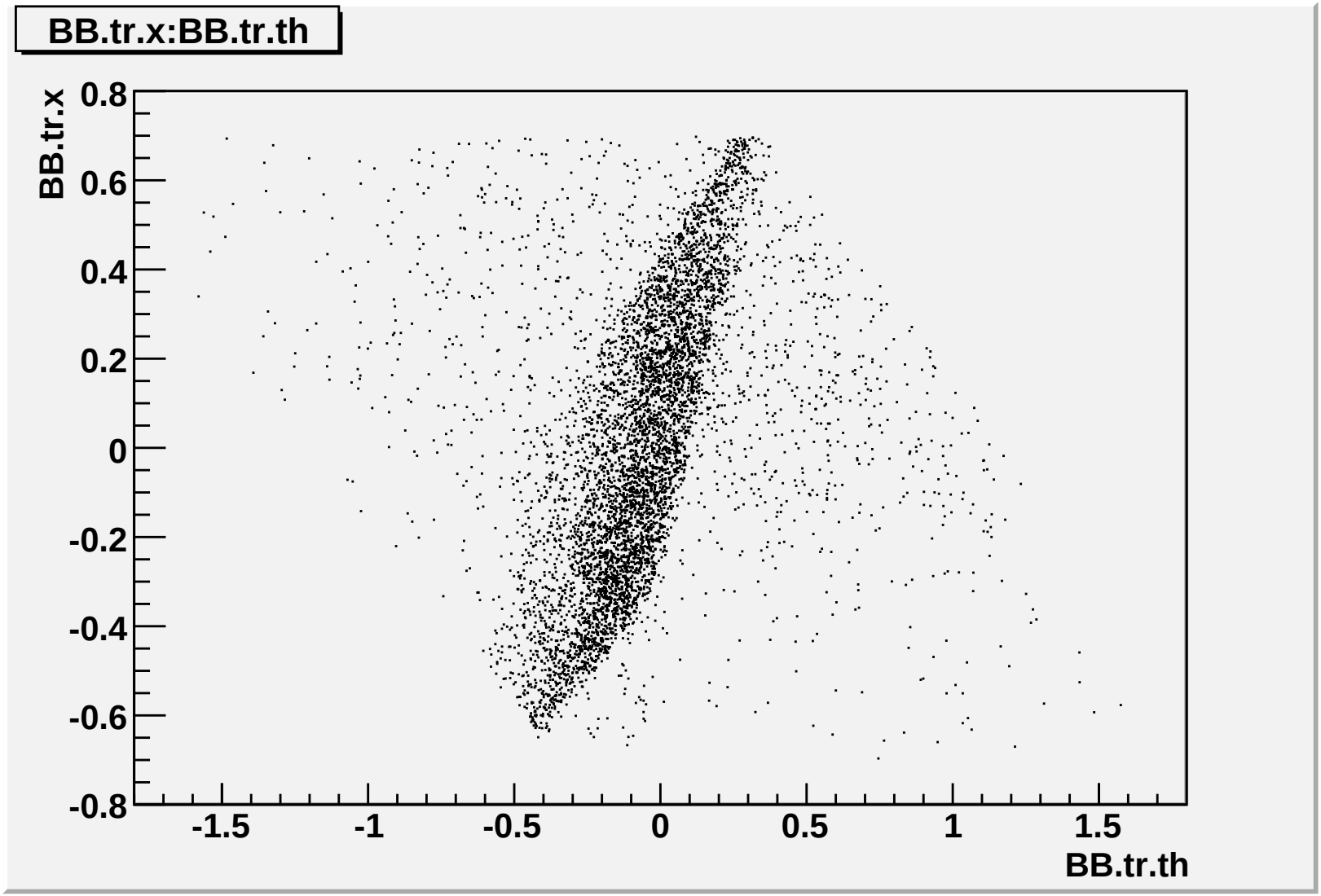

BB.tr.chi2 {BB.tr.n==1&&BB.tr.chi2<40&&BB.tr.ndof==8}
h1
Entries 3742
Mean 9.353
RMS 7.171
180
160
140
120
100
80
60
40
20
0
0
5
10
15
20
25
30
35
40

Track y vs phi

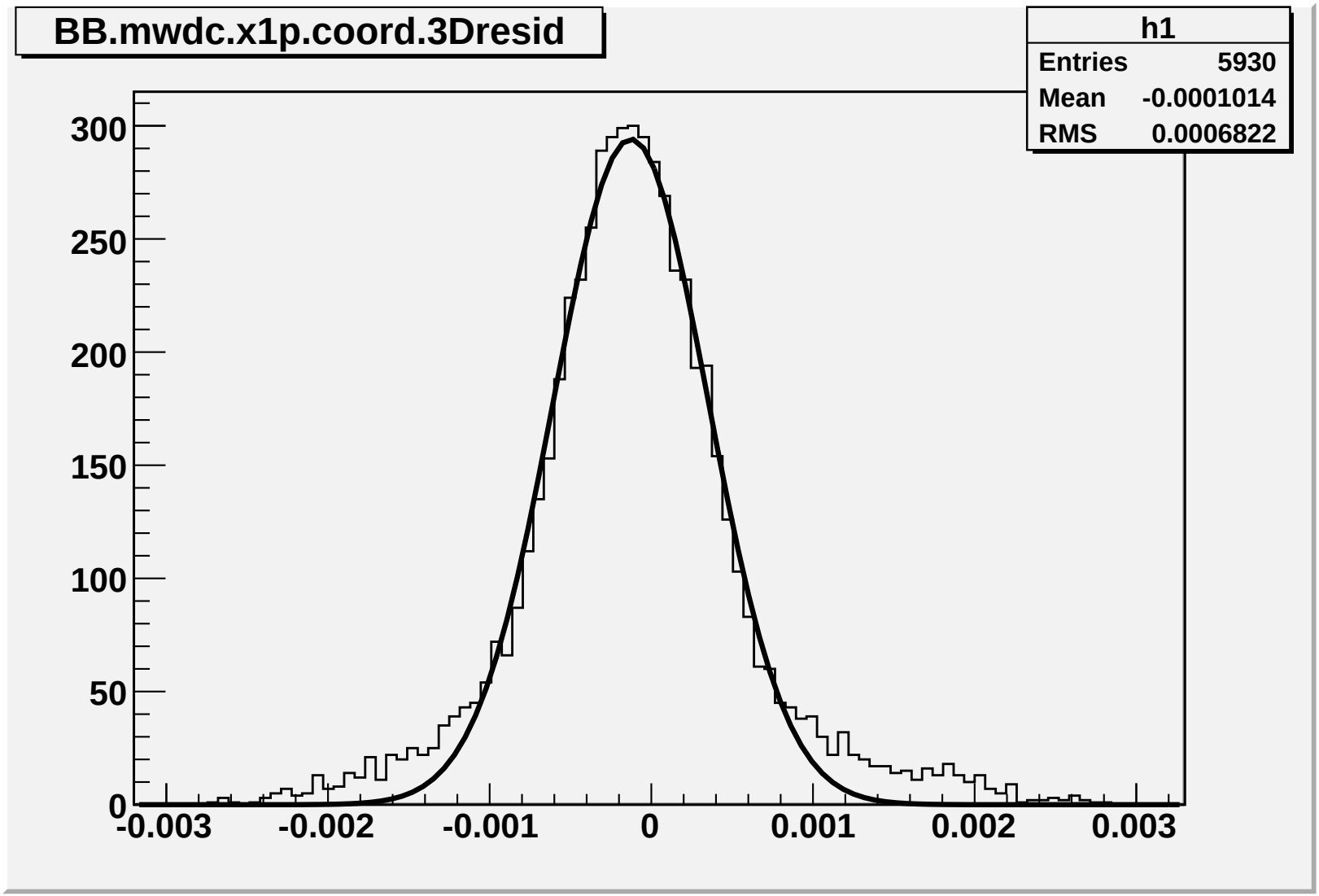
NB: approx. point target



Track x vs theta



Hit position residuals



Unfinished in libTreeSearch

- Include improved clustering algorithm
- Investigate sporadic events with missing tracks
- Detailed Monte Carlo tests
- Detailed comparison with results from G_E^n code
- Optimize algorithms (esp. 2D de-cloning)
- Add ability to pre-cut using shower, scintillator, etc.
- Add $\cos\theta$, timing, fringe field corrections etc. ("FineTrack")

Unfinished in libTreeSearch

- Include improved clustering algorithm
- Investigate sporadic events with missing tracks
- Detailed Monte Carlo tests
- Detailed comparison with results from G_E^n code
- Optimize algorithms (esp. 2D de-cloning)
- Add ability to pre-cut using shower, scintillator, etc.
- Add $\cos\theta$, timing, fringe field corrections etc. ("FineTrack")

Unfinished in libTreeSearch

- Include improved clustering algorithm
- Investigate sporadic events with missing tracks
- Detailed Monte Carlo tests
- Detailed comparison with results from G_E^n code
- Optimize algorithms (esp. 2D de-cloning)
- Add ability to pre-cut using shower, scintillator, etc.
- Add $\cos\theta$, timing, fringe field corrections etc. ("FineTrack")

Unfinished in libTreeSearch

- Include improved clustering algorithm
- Investigate sporadic events with missing tracks
- Detailed Monte Carlo tests
- Detailed comparison with results from G_E^n code
- Optimize algorithms (esp. 2D de-cloning)
- Add ability to pre-cut using shower, scintillator, etc.
- Add $\cos\theta$, timing, fringe field corrections etc. ("FineTrack")

Unfinished in libTreeSearch

- Include improved clustering algorithm
- Investigate sporadic events with missing tracks
- Detailed Monte Carlo tests
- Detailed comparison with results from G_E^n code
- Optimize algorithms (esp. 2D de-cloning)
- Add ability to pre-cut using shower, scintillator, etc.
- Add $\cos\theta$, timing, fringe field corrections etc. ("FineTrack")

Unfinished in libTreeSearch

- Include improved clustering algorithm
- Investigate sporadic events with missing tracks
- Detailed Monte Carlo tests
- Detailed comparison with results from G_E^n code
- Optimize algorithms (esp. 2D de-cloning)
- Add ability to pre-cut using shower, scintillator, etc.
- Add $\cos\theta$, timing, fringe field corrections etc. ("FineTrack")

Unfinished in libTreeSearch

- Include improved clustering algorithm
- Investigate sporadic events with missing tracks
- Detailed Monte Carlo tests
- Detailed comparison with results from G_E^n code
- Optimize algorithms (esp. 2D de-cloning)
- Add ability to pre-cut using shower, scintillator, etc.
- Add $\cos\theta$, timing, fringe field corrections etc. ("FineTrack")

To Do List

- Status of coincidence time module; improve if necessary. Responsibility?
- Status of RICH software
- Cherenkov analysis? Database, calibration?
- Status of BigBite optics? Improve? Responsibility?
- Assign responsibility for online analysis; set up scripts and databases
- Plan required wire chamber surveys
- Learn calibration techniques from E04-007 (wire chamber etc.); plan calibration runs, if any

To Do List

- Status of coincidence time module; improve if necessary. Responsibility?
- Status of RICH software
- Cherenkov analysis? Database, calibration?
- Status of BigBite optics? Improve? Responsibility?
- Assign responsibility for online analysis; set up scripts and databases
- Plan required wire chamber surveys
- Learn calibration techniques from E04-007 (wire chamber etc.); plan calibration runs, if any

To Do List

- Status of coincidence time module; improve if necessary. Responsibility?
- Status of RICH software
- Cherenkov analysis? Database, calibration?
- Status of BigBite optics? Improve? Responsibility?
- Assign responsibility for online analysis; set up scripts and databases
- Plan required wire chamber surveys
- Learn calibration techniques from E04-007 (wire chamber etc.); plan calibration runs, if any

To Do List

- Status of coincidence time module; improve if necessary. Responsibility?
- Status of RICH software
- Cherenkov analysis? Database, calibration?
- Status of BigBite optics? Improve? Responsibility?
- Assign responsibility for online analysis; set up scripts and databases
- Plan required wire chamber surveys
- Learn calibration techniques from E04-007 (wire chamber etc.); plan calibration runs, if any

To Do List

- Status of coincidence time module; improve if necessary. Responsibility?
- Status of RICH software
- Cherenkov analysis? Database, calibration?
- Status of BigBite optics? Improve? Responsibility?
- Assign responsibility for online analysis; set up scripts and databases
- Plan required wire chamber surveys
- Learn calibration techniques from E04-007 (wire chamber etc.); plan calibration runs, if any

To Do List

- Status of coincidence time module; improve if necessary. Responsibility?
- Status of RICH software
- Cherenkov analysis? Database, calibration?
- Status of BigBite optics? Improve? Responsibility?
- Assign responsibility for online analysis; set up scripts and databases
- Plan required wire chamber surveys
- Learn calibration techniques from E04-007 (wire chamber etc.); plan calibration runs, if any

To Do List

- Status of coincidence time module; improve if necessary. Responsibility?
- Status of RICH software
- Cherenkov analysis? Database, calibration?
- Status of BigBite optics? Improve? Responsibility?
- Assign responsibility for online analysis; set up scripts and databases
- Plan required wire chamber surveys
- Learn calibration techniques from E04-007 (wire chamber etc.); plan calibration runs, if any