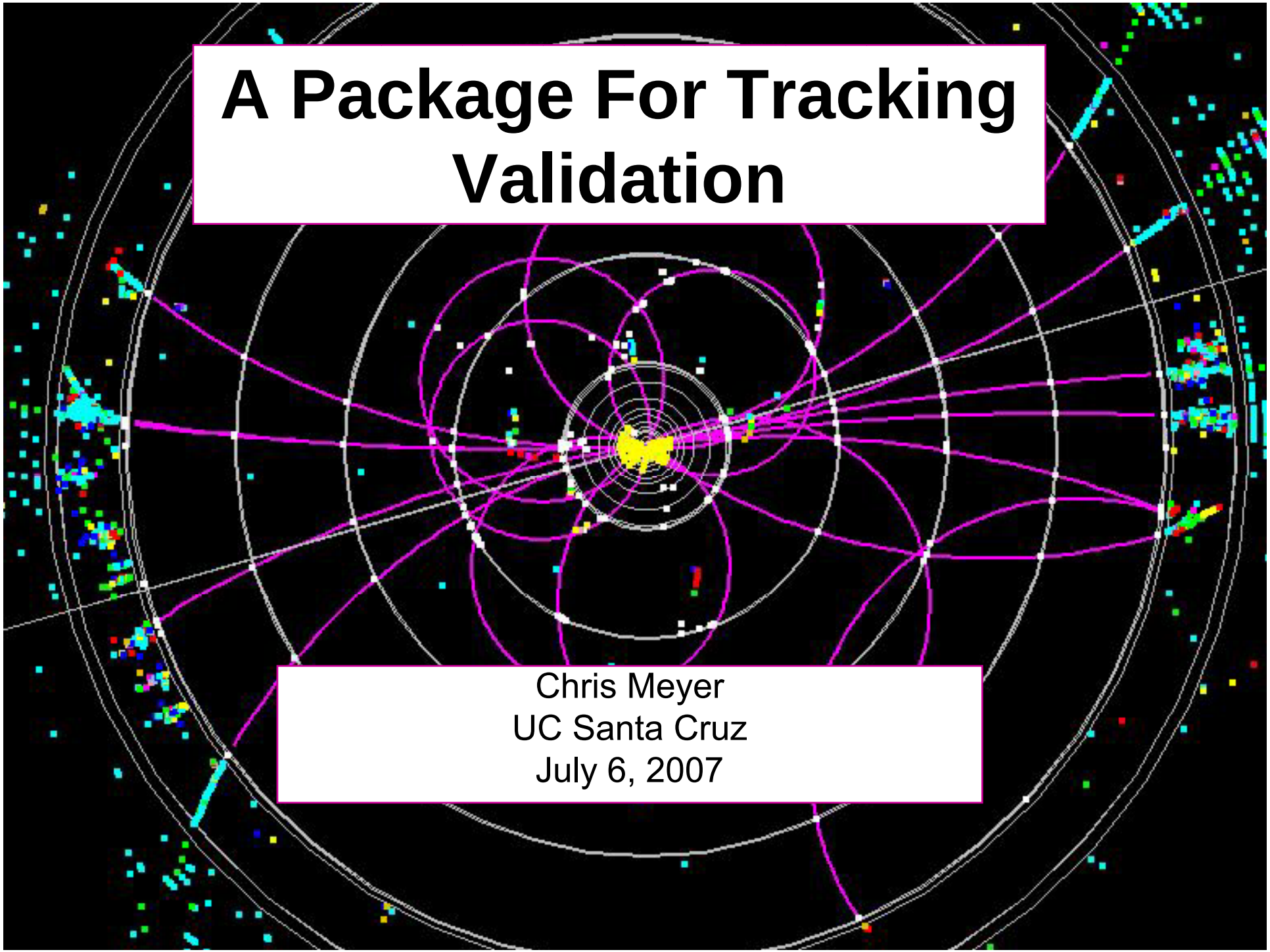
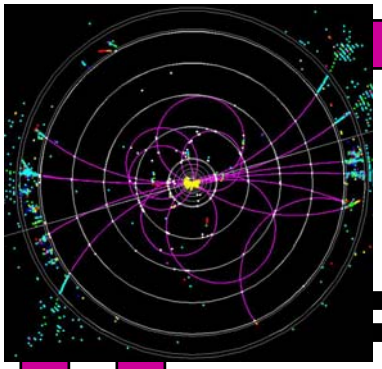


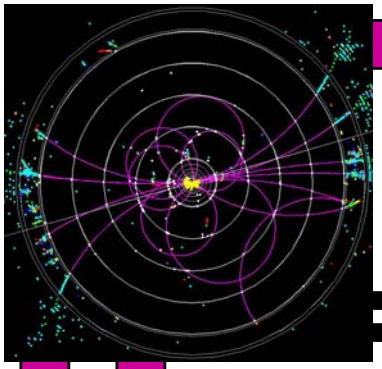
A Package For Tracking Validation

Chris Meyer
UC Santa Cruz
July 6, 2007



Motivation

- Analyze pattern recognition
- Determine efficiency of tracking
- Asses performance of track fitting algorithms
- Create a benchmark for track finding/fitting algorithms

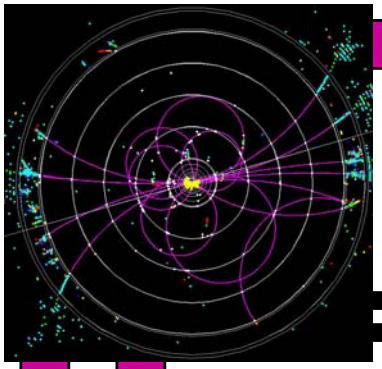


Approach

We first want to run a tracker to fill the track banks with data. Still within JAS we pass events to a java write-out routine that prints out information on all the tracks and MC Particles of the event (ASCII file).

Next the file is read into a c++ analysis package. The data is analyzed for tracking efficiency and fitting accuracy, as well as general tracking statistics and distributions.

The results are plotted by ROOT and saved as gif files with a text file summarizing fake track rates, as well as an info file on what cuts were made and general approach.



Event Selection

Since most track finders are implemented in JAS we do the initial runs there. We start by selecting events that pass a JetAccept test. Tracks fed into the test:

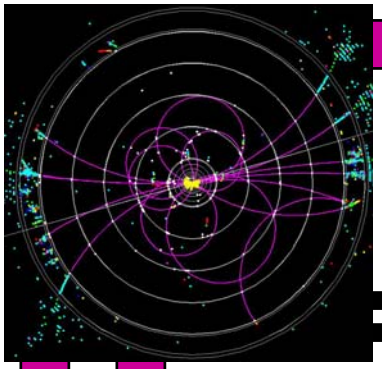
- Are FINAL or INTERMEDIATE state
- Have a radius to origin < 1cm
- Have a path length > 500 cm
- Are NOT backscatter

To pass the JetAccept test, the event shape (formed from tracks passing the above criteria) must have:

- Cos θ_{TA} < 0.5
- Thrust > 0.94

Change this for forward region

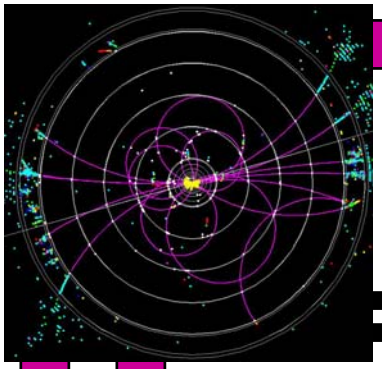
The successful events now proceed to the write-out stage.



General Principles of Write-Out File

For each passing event:

- List of information written for each reconstructed track
- Initial loose cuts applied to MC Particles
- Corresponding list written out for each particle
- Cuts on tracks and tighter cuts on MC Particles applied later by user in c++ code.



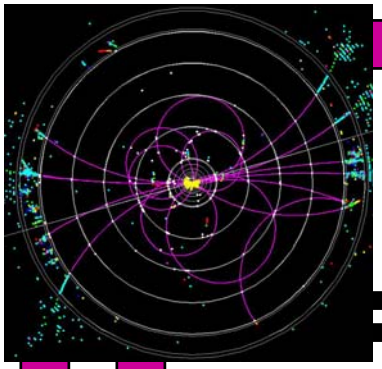
Loose Findable Particle Cuts

First we cut particles that we know we don't want. These initially include those that:

- Are NOT FINAL or INTERMEDIATE state
- Are backscatter
- Aren't charged

In addition if the associated MC Particle has an unknown PDGID the track and associated MC Particle are dropped

Information about all remaining tracks and MC Particles is written out to ASCII file (*.dat)

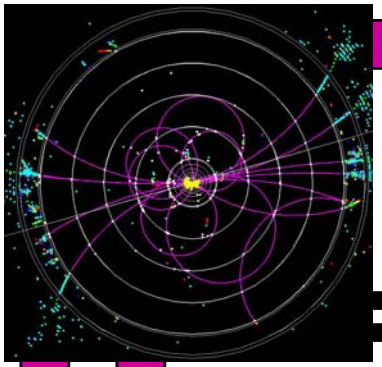


Write Out Frame Work (1)

To include all data that might be useful in analysis at some latter point every MC Particle and Track (found by finder and passing loose cuts) are output into an ASCII data file, one MCP or Track per line.

Each line contains the relevant track or particle information on its owner separated by a single space (although any amount of white space is OK)

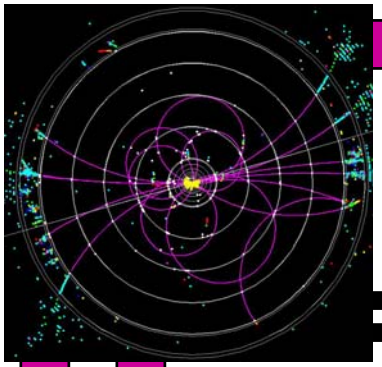
Event #	Track/Particle Parameter 1	Track/Particle Parameter 2	Etc…



Write Out Frame Work (2)

The order of the track parameters is (in array index):

0 Event Number	11 zOrg *(particle)*
1 $\|\alpha\|$ *(particle)*	12 zDCA (track)
2 $\|\alpha\|$ (track)	13 Path Length *(particle)*
3 cos θ *(particle)*	14 Path Length (track)
4 cos θ (track)	15 p_t *(particle)*
5 φ *(particle)*	16 p_t (track)
6 φ (track)	17 nDOF (track)
7 ω *(particle)*	18 Purity (track)
8 ω (track)	19 ω_{error} (track)
9 rOrg *(particle)*	20 Index
10 DCA (track)	21 Associated Index



Write Out Frame Work (3)

When writing out index (track or particle number in write-out file):

- Tracks have positive index
- Particles have negative index

For tracks:

- All particle values are filled by value of associated MCP
- All particle values filled with 0 if no associated MCP

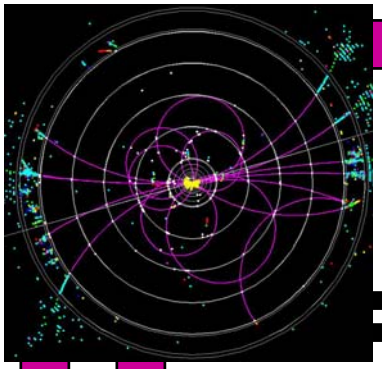
For particles:

- All track values are filled with 0
- Purity, nDOF, ω_{error} are filled with 0

Associated Index refers to the index:

- Associated MCP for tracks
- Associated track for MCP
- 0 if no association

Note: For multiple data sets it's okay to repeat event # in the write-out file.



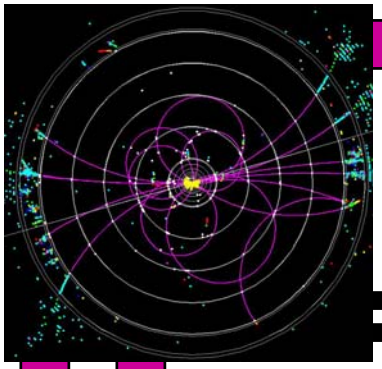
Analysis

Once the files have been read into the analysis code there are several options for analysis:

- Efficiency vs. Track Parameter (Pattern Recognition)
- Distribution of Track Parameter
- Fake Tracks by Track Parameter
- Total Fake vs. Found Rate
- Triplot Generation (Overall Tracking Efficiency)
- Fake Track Write Out

This is still under development and relies on commenting / un-commenting code lines for now. In the future control flags will be added for ease of use.

These options are described in the following slides.



Default Cuts In C++ Code

Findable Particles:

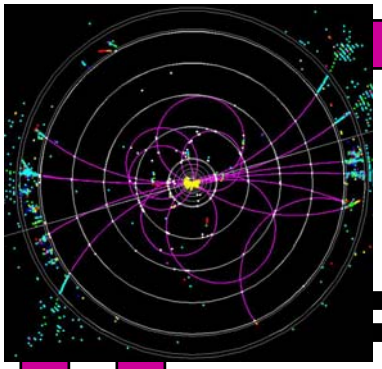
- rOrg < 1 cm
- cos θ < 0.5
- Path Length > 500 cm
- p_t > 0.75

Acceptable Tracks:

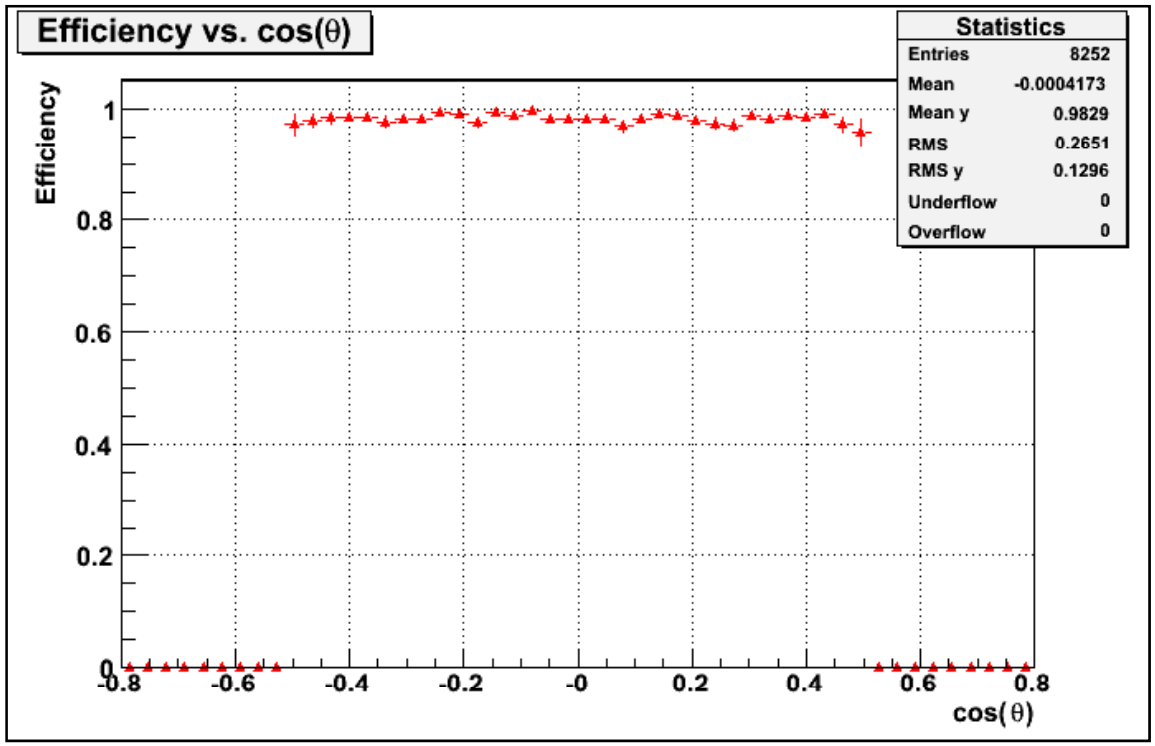
- DCA < 1 cm
- $\omega_{error} \neq 0$ (Fit Successful)
- cos θ < 0.5
- p_t > 5

The last two are for distributions and fake rate calculations only; not applied when calculating efficiency

These cuts can be changed in the Std Header File

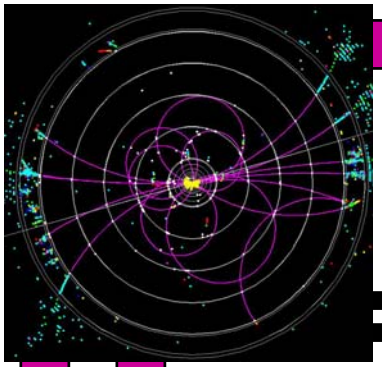


Efficiency

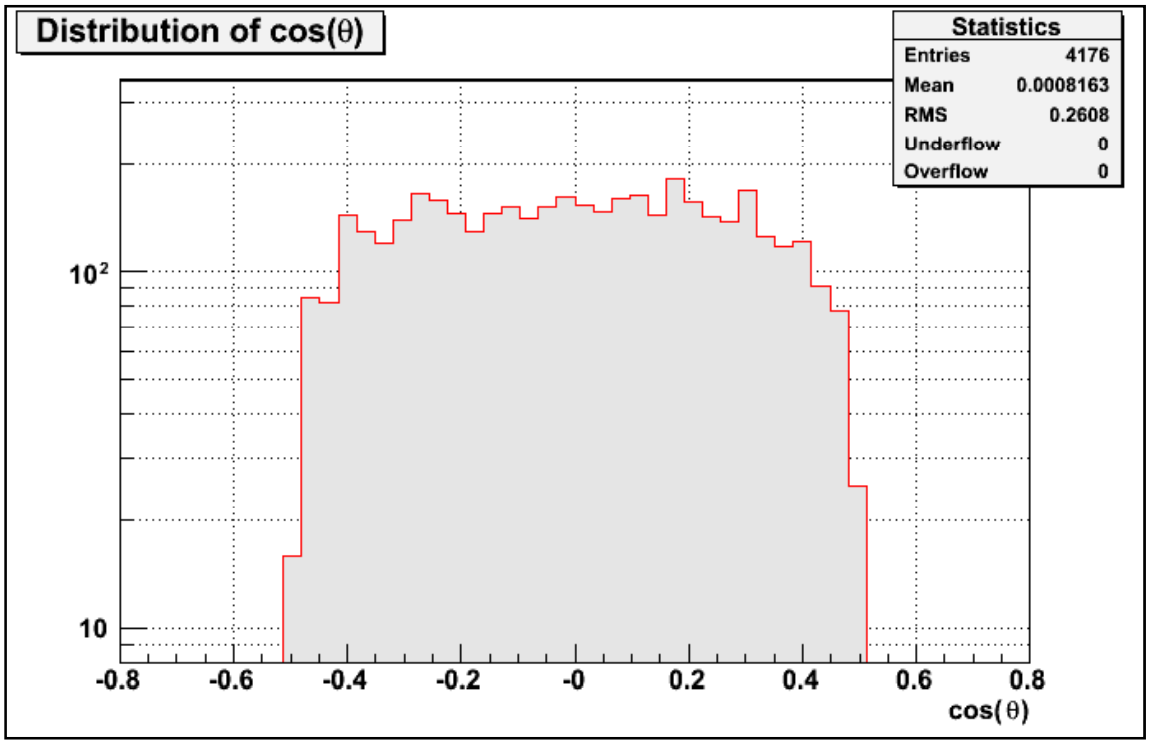


For Findable particles, shows a Profile Histogram of Track Parameter by Efficiency (0-1).

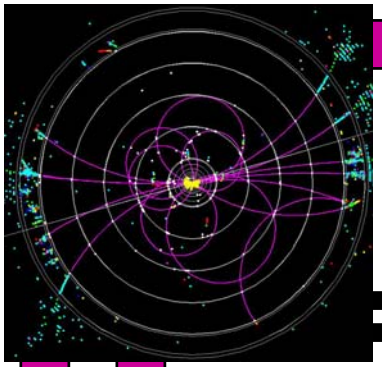
Particle are “found” efficient if the associated track has a truth matching purity of > 0.79.



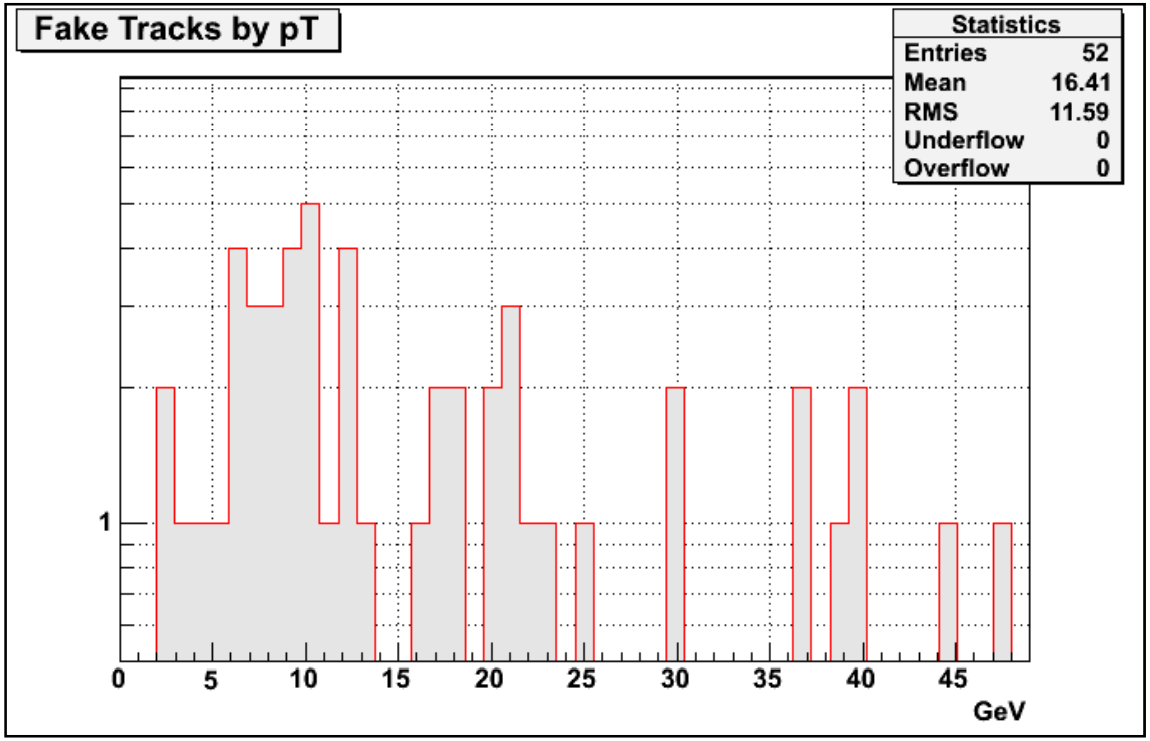
Distribution



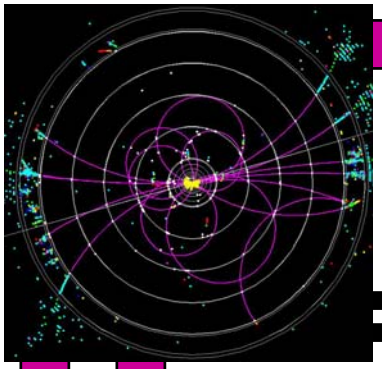
For Acceptable Tracks, shows a Histogram of the Track Parameter.



Fake Tracks



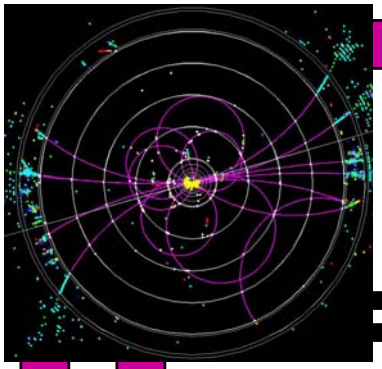
For successfully fit acceptable tracks (DCA<1 cm, ω_{error}>0) that don't have an associated MC Particle, or have a purity of < 0.79.



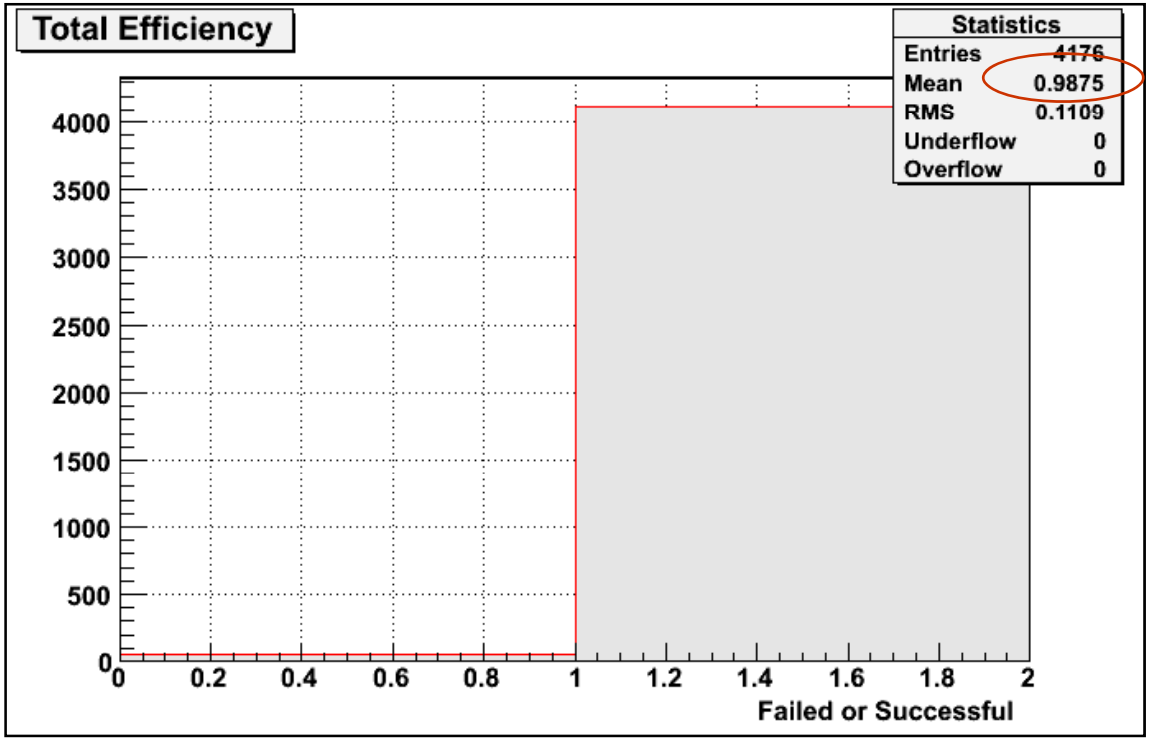
Fake Tracks Write Out

Omega Range (cm^-1)	Fake Tracks	Total Tracks	% Fake
3e-005 to 7.5e-005	0	0	0
7.5e-005 to 0.00012	0	23	0
0.00012 to 0.0002	0	111	0
0.0002 to 0.000375	3	387	0.775194
0.000375 to 0.00075	14	848	1.65094
0.00075 to 0.0015	16	1202	1.33111
0.0015 to 0.003	15	1596	0.93985
0.003 to 0.0075	4	9	44.4444
0.0075 to 0.015	0	0	0
0.015 to 0.03	0	0	0

Gives a write out by ω range of fake tracks (successfully fit acceptable tracks without associated MCP or purity < 0.79).

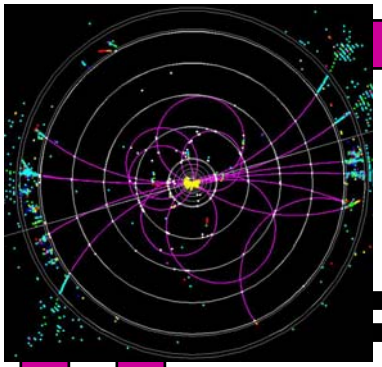


Total Fake vs. Found Rate

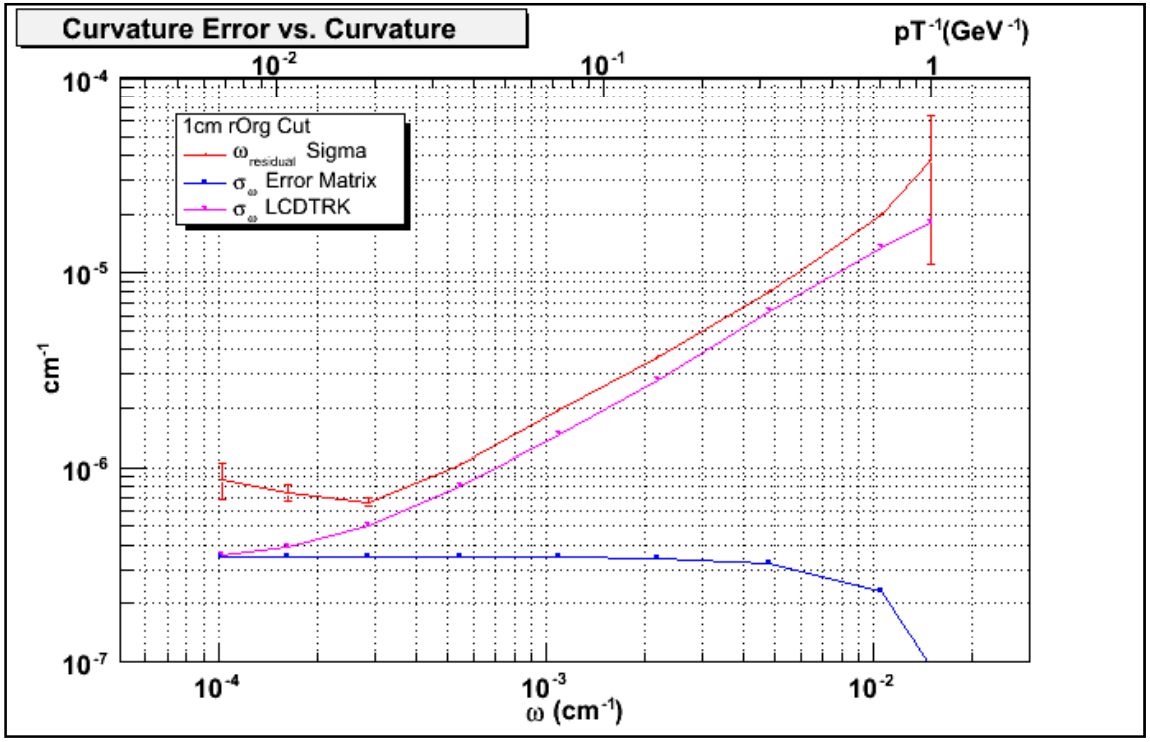


← Purity

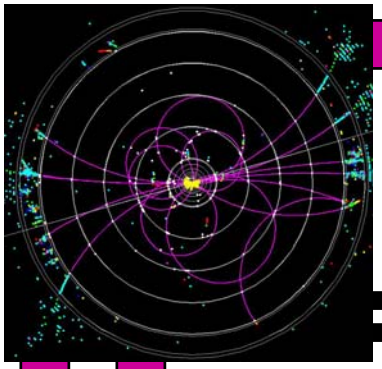
A Histogram of Acceptable Tracks with 1 if the fit was successful and purity > 0.79, 0 otherwise.



Triplot



The triplot compares rms, sigma from gaussian fit, and LCDTRK value of successfully fitted tracks.



Ready To Go

We have developed a framework (JAS to ASCII file; analysis with stand-alone c++ program) that performs tracking validation.

We have a package within this framework that is ready to run on any finding or fitting algorithm.

The package and framework are easily edited or extended by user or by us upon request.