Track Finding in GlueX Development Report II

D. Lawrence
Jefferson Lab

September 2, 2005

Abstract

Progress of track finding in multi-track Monte Carlo data are presented. An alternative track-seed based approach is used to identify track candidates in 2D which are then used to find tracks in 3D. Efficiencies for 4 muon track events ($\mu^+\mu^+\mu^-\mu^-$) are given as functions of kinematic variables, p, ϕ, θ and the number of hits per track. Results are given for clean MC data as well as for "smeared" data with random noise hits. Results of a simulation targeting a suggestion by the 2004 Detector Review regarding FDC package placement are also given. Finally, the organization of the tracking code and its implementation in the DANA framework are given. This note documents a work in progress.

Contents

1 Introduction

This note documents the current progress of the track finding (sometimes called *pattern recognition*) software for GlueX. It almost supercedes the earlier document GlueX-doc-441[1], but there may still be some useful information there.

The GlueX detector is designed such that it can simultaneously measure several charged particle tracks as they move through the magnetic field of the solenoid. The primary detector systems used for this are the central drift chambers(CDC) and the forward drift chambers(FDC). Each chamber will supply data in the form of "hits" which must be sorted into lists of track candidates before they can be passed through the final fitting/filtering stage to produce tracks.

Most of the software described in [1] was re-written replacing several of the algorithms. Among these, the Hough Transform[7] method was replaced with a seed-based method described in section 2. Also new is the way track finding efficiency is measured. The earlier parameter matching method of measuring efficiency was replaced with a hit matching method described in section 4.

All of the work reported in this document used HDGeant generated "truth tags" as the initial source of hit information. The tracking code works only with 3D hit positions from CDC and FDC hits. It does not use any GlueX geometry specific information with the execption of the range in z covered by the GlueX target. In this way, the code can be used as-is to test how geometry changes affect the track finding efficiency. An example of this is given in section 6.

2 Seed-based Track Finding

The concept used in seed-based tracking is simple: Start at the outside edge of the chamber where the hit density is lowest and look for hits that are close together. One starts with a single hit and looks for its nearest neighbor and then its nearest neighbor and so on until it finds that either the nearest (unused) neighbor is greater than a certain distance [1] away or it has obtained enough points [2] for an initial fit. This initial set of points is called the *seed*. Seeds are only found in the X/Y plane (z information in unused at this point).

2.1 Seed Finding/Fitting

To help with diagnostics and development of the tracking code, the *patfind* program was updated. Figure 1 shows the updated patfind with one of the 4μ track events used for the studies presented in this report. Points are drawn with varying color and size to indicate at what phase of the track finding the point was used. For example, in figure 1, the green hits made up the seed for the selected track. Similarly, the slightly smaller megenta points are "on-circle", the blue points are "on-line" (i.e. on the line in the $\phi - z$ plane) and the small red dots are those hits included as part of the final track candidate.

The thick, cyan-colored line in figure 1 represents the circle obtained from a fit to just the seed hits (green). The darker, thinner line is the result of the 3-D fit for the track. The yellow lines represent similar information for the un-selected tracks.

[1] DANA configuration parameter TRK:MAX_SEED_DIST = 5.0cm
[2] DANA configuration parameter TRK:MAX_SEED_HITS = 10

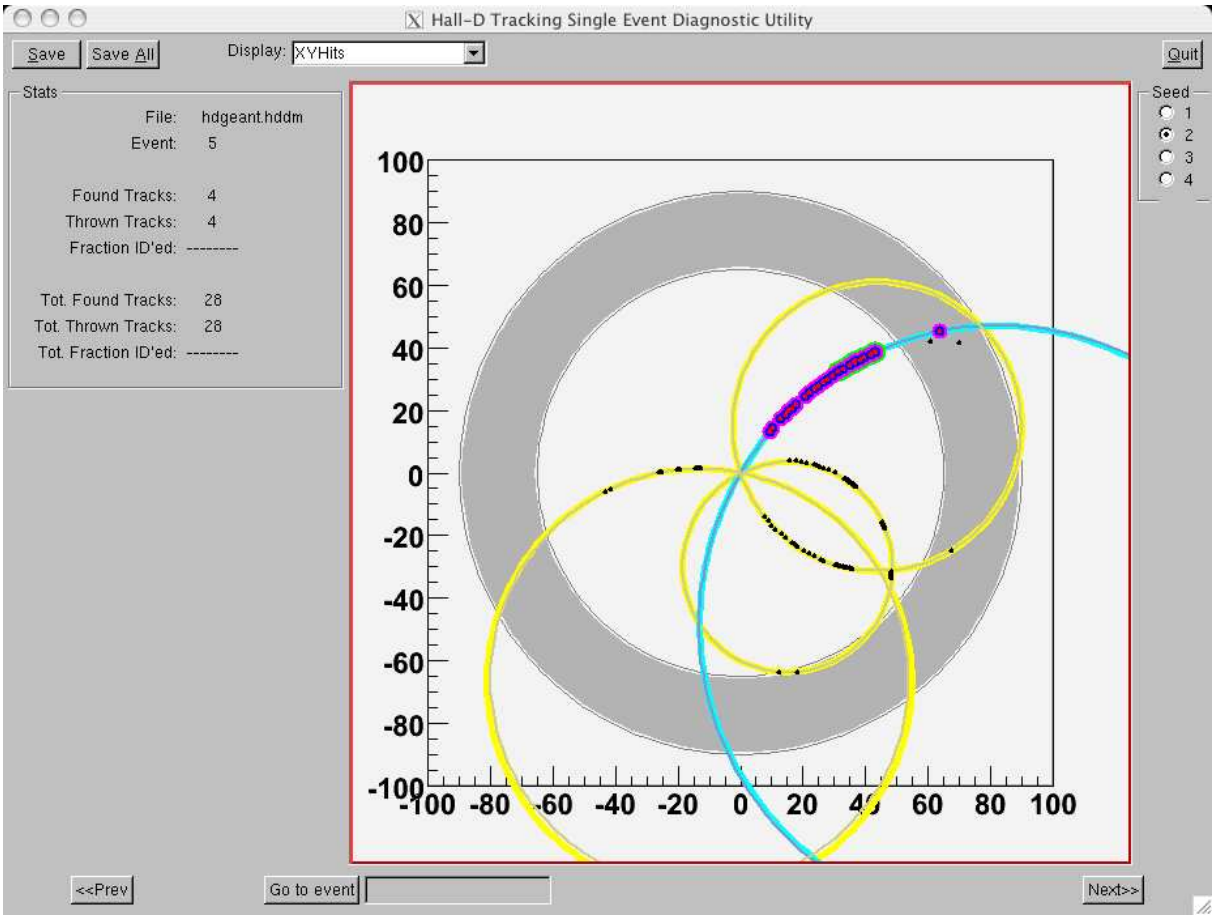

Figure 1: Updated *patfind* program that used track-seeds.

Fits are done in three stages, first, a linear regression style fit is done to the hits in the X/Y plane to find the parameters of the circle that best descibes the hits AND passes through the beamline. Second, the center of the circle is used to calculate the ϕ angle of the hit about the axis of the helix. This will have a linear relationship with z as shown in figure 2. The ϕ, z points are used to determine the angle of the line in the ϕ/z plane. Third, using the $\phi - z$ angle, the same ϕ, z points are used to determine the z coordinate of the vertex.

2.2 Finding the $\phi - z$ angle

The range covered by the target in z [3] is used to determine the range of angles each hit is consistent with. Figure 2 (right) shows a histogram whose peak gives the $\phi - z$ angle for the hits shown in the plot on the left. The histogram is filled in the following way: For each hit [4] , the angles are calculated for a vertex position at the upstream and downstream ends of the target, on the beamline. These two angles define a range in the

[3]DANA configuration parameters TR:TARGET_Z_MIN and TRK:TARGET_Z_MAX (30cm and 80cm)
[4]These are "on-cicle" hits. i.e. hits which were close to the circle found using only X/Y information

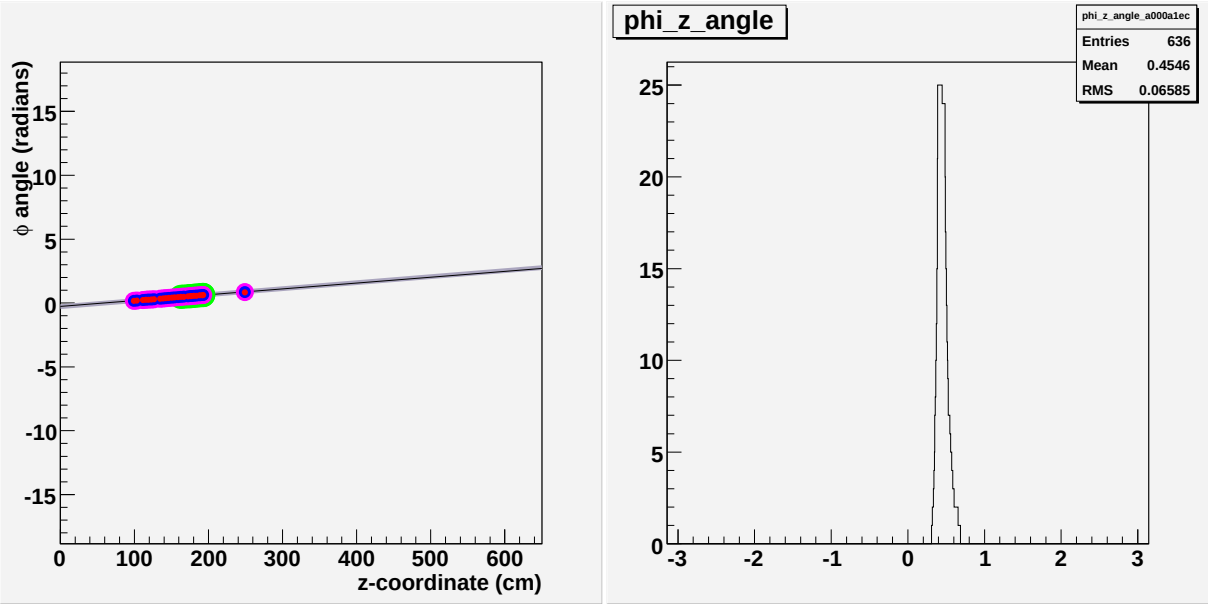

Figure 2: Left:ϕ vs. z for hits that were "on circle". Right:Histogram of angle in $\phi - z$ plane each point makes with the found z-vertex position.

$\phi - z$ angle histogram. Each bin in this range is incremented by one. In this way, hits close to the target are effectively weighted less because they fill a broader range of bins in the histogram. The hits that are further away have a longer lever-arm and fill a smaller range of bins, effictively giving them more weight.

Since this often leads to a plateau as opposed to a sharp peak, the center of the plateau is used for the actual value of the $\phi - z$ angle. The range covered by the plateau, however, is used to determine the z-vertex as described in section 2.3.

Often, hits from different tracks are on-circle in the X/Y plane. These however, do not tend to fall on the line in the $\phi - z$ plane. Figure 3 shows one such event. It is because of these extra hits, that we cannot simply do a linear regression on all ϕ, z points as the extra hits would significantly bias the results. As indicated in figure 3, the extra hits will tend to form a second peak in the $\phi - z$ angle histogram. The taller peak is always chosen.

2.3 Finding the Z vertex

The z-vertex is found in a very similar way as the $\phi - z$ angle. Using the range of $\phi - z$ angles from the previous step, a range in the z-vertex position is calculated for each hit. The bins corresponding to this range are then incremented in another histogram resulting in one like that of figure 4. Here, again, a natural weighting occurs that gives the correct bias. The points closest to the target will increment a narrower range in the histogram while those furthest away will increment a wider range. This, like in the $\phi - z$ case, tends to lead to a plateau like that shown in figure 4. Again, we take the center of the plateau as the value of the z-vertex position.

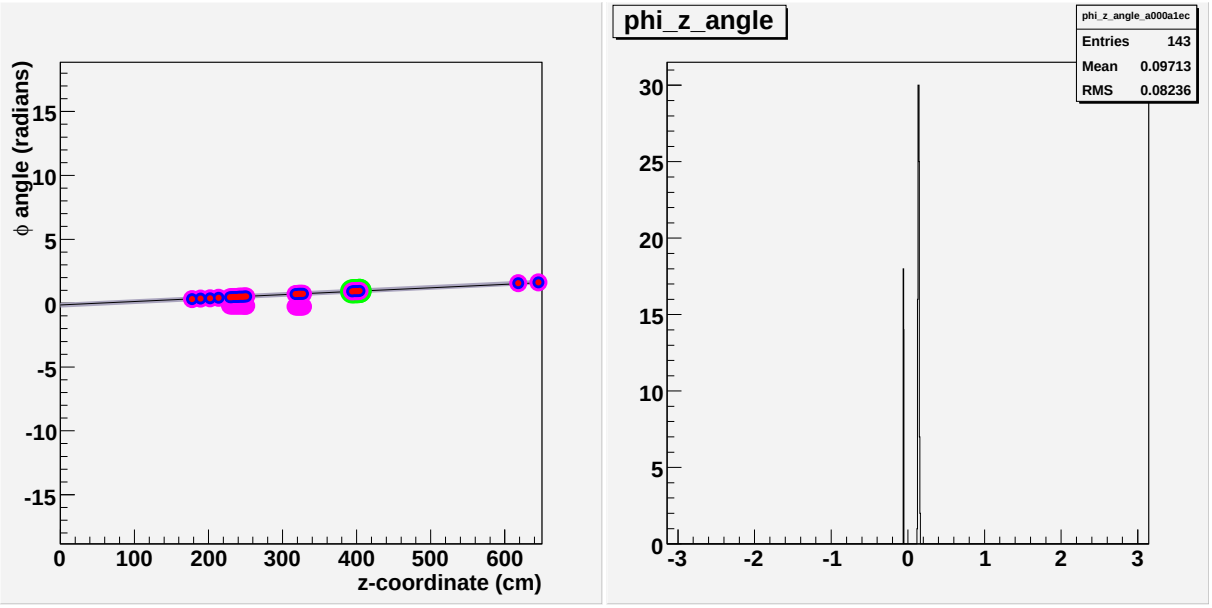

Figure 3:

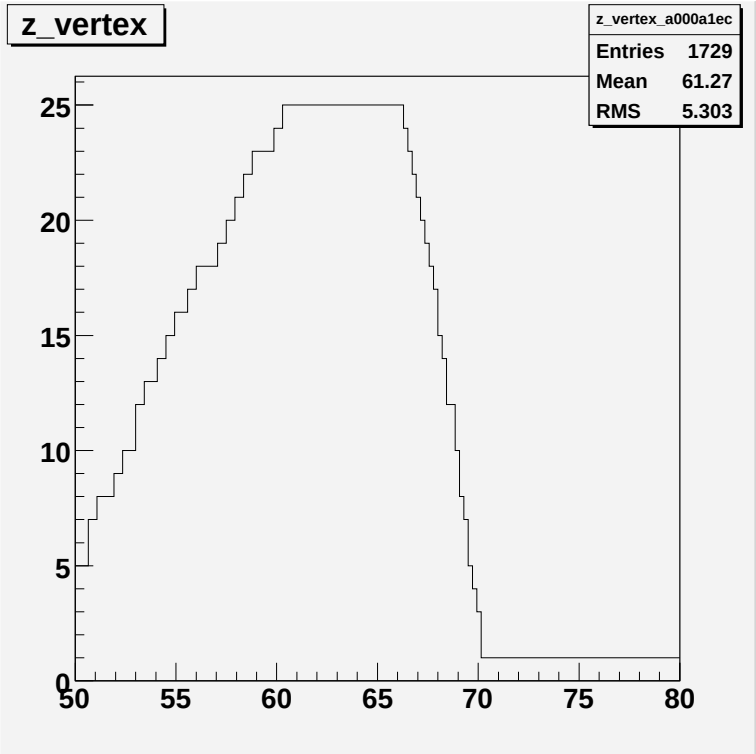

Figure 4: Histogram of z-vertex made by projecting each point back to beamline over range of $\phi - z$ angles determined by peak as in figure 2. The center of the plateau is used as the vertex. For this simulation, the thrown vertex was at $z = 65cm$

3 Data Smearing and Noise Hits

To better simulate real data, the values obtained from the "truth tags" for the CDC and FDC hits were smeared and random noise hits were added. The smearing and noise hits were done individually for the CDC and FDC systems. The following sections describe the details.

On average, there were about 126 noise hits in the CDC and FDC combined per event. The average track has around 30 hits per event. For the 4 track event used, that meant an average of about 120 real hits per event.

Figure 5 shows an event in its pure, smeared, and smeared with noise forms.

3.1 CDC smearing and noise

The X and Y hit positions in the CDC are smeared by randomly placing them in a circle of diameter 1.6cm centered on the actual hit position. The 1.6cm is the diameter of a straw tube.

The Z hit position in the CDC is smeared by sampling a gaussian with $\sigma = 7.5cm$ centered at the actual hit position. The value of 7.5cm is obtained by estimating the error as $\sigma_z = 0.8/sin(6^o)$. The 6^o i the pitch of the stereo layers with respect to the axial layers. The 0.8cm is the radius of a straw tube. The value of σ_z was rounded down to 7.5cm.

The noise hits in the CDC were generated assuming an average noise hit occupancy of 3%. With 3240 wires in the CDC, this meant an average of 97 noise hits per event. The exact number of noise hits generated for an event was sampled from a gaussian distribution with $\sigma = \sqrt{97}$. The noise hits were evenly distributed throughout the fiducial volume of the CDC.

3.2 FDC smearing and noise

The X and Y hit positions in the FDC are smeared by randomly placing them in a square of side 0.5cm centered on the actual hit position. The 0.5cm is the wire spacing in the FDC. This would correspond to using hit-based information only from the chambers. It was done this way since it would allow track finding to be done at a hit-based level in the FDC, eliminating the need for at least one set of calibration constants.

No smearing was done on the FDC z coordinate since it that should be well known.

The noise hits in the FDC were generated assuming an average noise hit occupancy of 1%. With 2856 anode wires in the FDC, this meant an average of about 29 noise hits per event. The exact number of noise hits generated for an event was sampled from a gaussian distribution with $\sigma = \sqrt{29}$. The noise hits were evenly distributed throughout the fiducial volume of the FDC.

3.3 Noise Hit Rejection

The introduction of random noise hits into the data worked to confuse the track finding algorithm, leading to lots of ghost tracks which, in turn, tended to mask real tracks. To counter this, a filter was used to remove hits believed to be noise before track finding

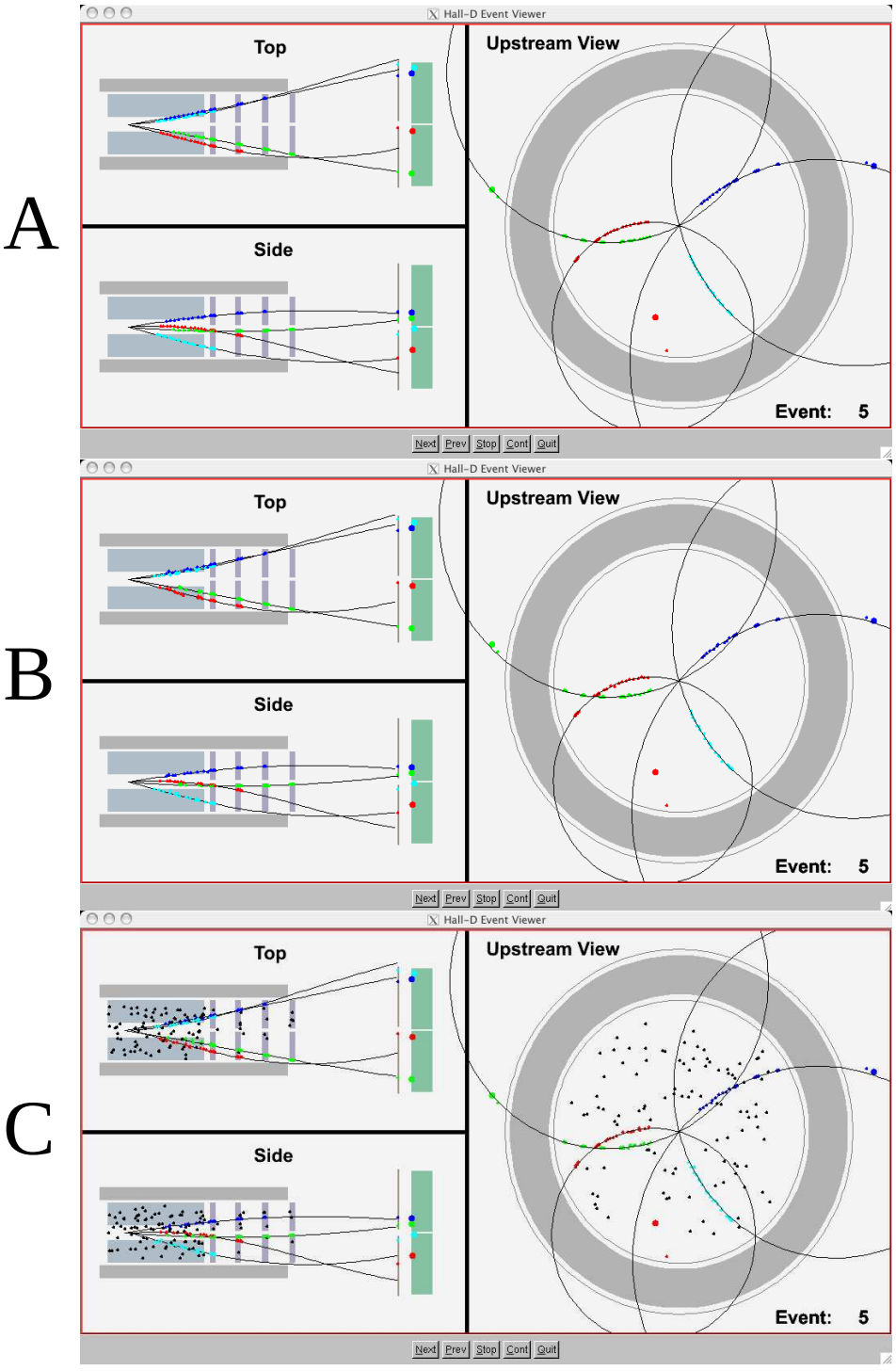

Figure 5: hdview of the same simulated $n\mu^+\mu^+\mu^-\mu^-$ event with A.) No smearing B.) FDC and CDC smearing (see text) and C.) FDC and CDC smearing as well as random noise hits.

began. The filter worked by simply cutting out hits whose nearest neighbor in the X/Y plane was less than some threshold [5].

The threshold is used to discriminate between hits that are from the real tracks, and hits from noise. Real tracks tend to have hits closely clustered in the X/Y plane where noise hits do not. The top plot in figure 6 shows the distribution of the nearest neighbor distance for hits from the same track(blue) and hits from different tracks or noise hits(red). The black line indicates the threshold value used of 2.0cm.

The value for this threshold was optimized by measuring the track finding efficiency at several values of the threshold. The bottom plot in figure 6 shows the results of these simulations for both pure data and data smeared with noise hits. The grey line indicates the threshold value of 2.0cm in this plot as well.

4 Track Finding Efficiency

The track finding efficiency was measured as a function of momentum, θ angle, ϕ angle, and number of hits per track for each of the three data conditions shown in figure 5. The efficiencies are shown in figures 7-10. Overall, the efficiency is about 92% for the worst case data while it is at about 98.5% for the ideal case.

4.1 How Tracking Efficiency Is Measured

There are two main approaches to determining the tracking efficiency: *Hit Matching* and *Parameter Matching*[6]. In the previous GlueX note[1], an implementation of the parameter matching was used. For parameter matching, one compares the final fit parameters with those of the thrown particles. This is straight forward, but may not yield accurate results, particularly in the present case when we have not performed the final Kalman fit.

For the present study, a hit matching approach was used. In this approach, truth information about each hit is used to determine if a track was found. Specifically, if 70% or more of the hits in a found track came from the same thrown track, then the track is considered to be found. This was the criteria used to produce the plots in figures 7-10.

The efficiency plots of figures 7-10 were all produced by the *DEventProcessor_TrackHists* object in the TRACKING package. This is accessible to any DANA program wishing to create these histograms. They will be placed in a directory called "TRACKING" in the currently open ROOT file. Also, one can simply run the *mctrk_ana* program on the HDGeant produced HDDM file. This is available so that anyone can easily check how geometry or early phase analysis changes affect the efficiency. See section 6 for one example.

5 Event Processing Rates

The speed at which the reconstruction code operates will be important to both the level-3 trigger and online monitoring. It is therefore worthwhile to keep track of how fast it operates to ensure it is not prohibitively slow. The event processing rate was measured

[5]DANA configuration parameter TRK:XY_NOISE_CUT (2.0 cm)

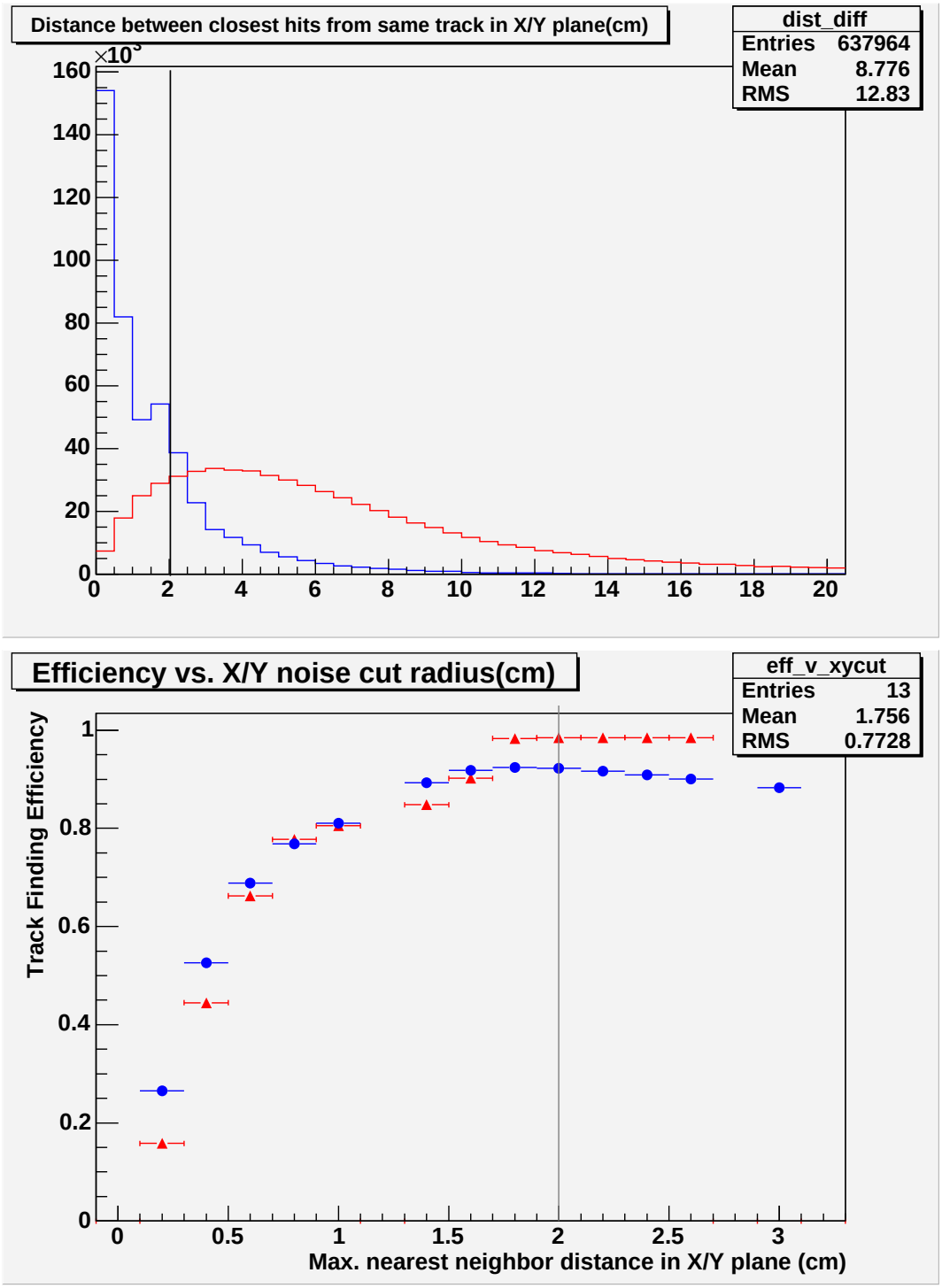

Figure 6: Plots used to determine the nearest nieghbor cut radius used in the noise hit filter (see text). Top:Distance between closest hit (in cm) when closest hit is from the same track(blue) or different tracks(red). Bottom: Overall tracking efficiency as a function of cut parameter for pristine tracks(red) and noisy, smeared tracks(blue).

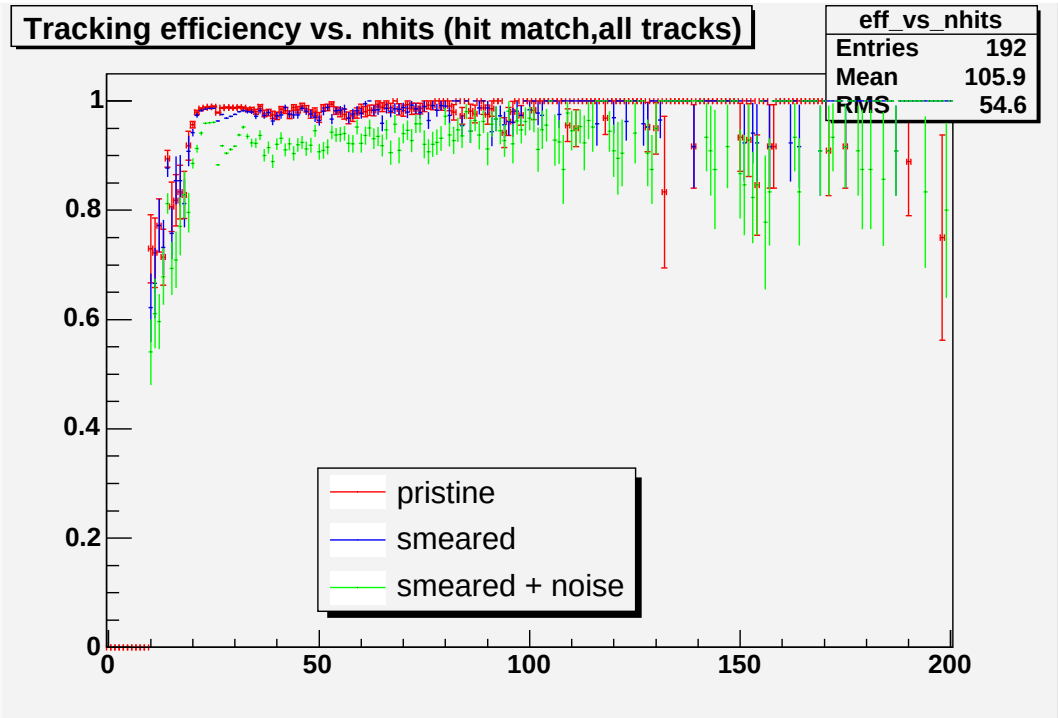

Figure 7: Track finding efficiency vs. number of hits per track for the three levels of data quality.

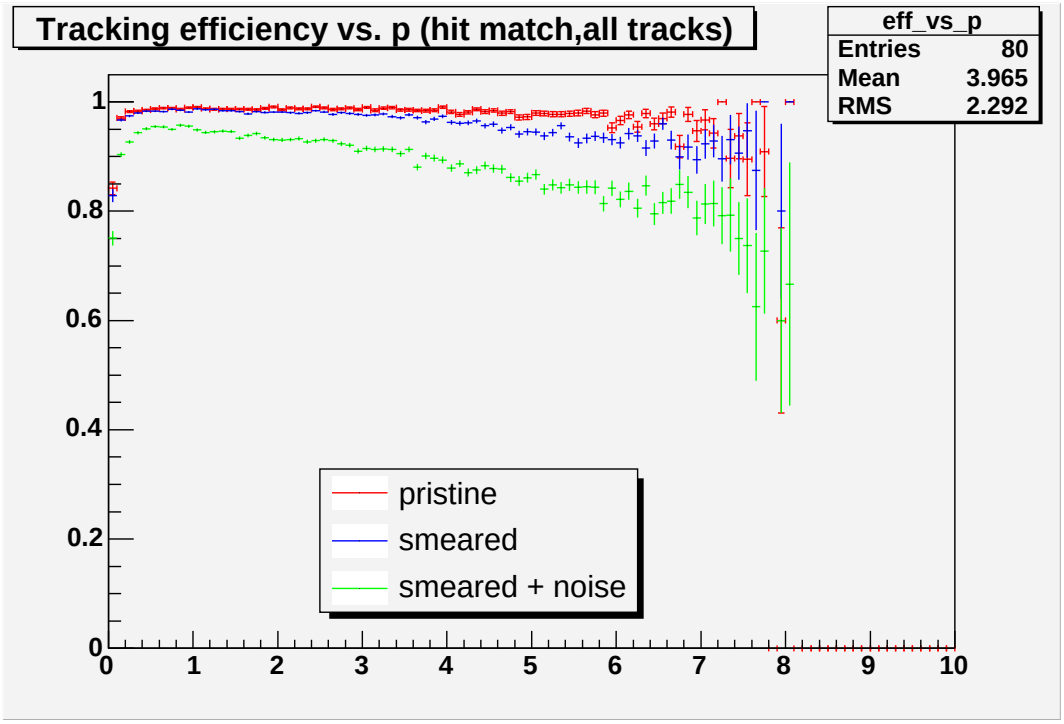

Figure 8: Track finding efficiency vs. μ momentum(GeV/c) for the three levels of data quality.

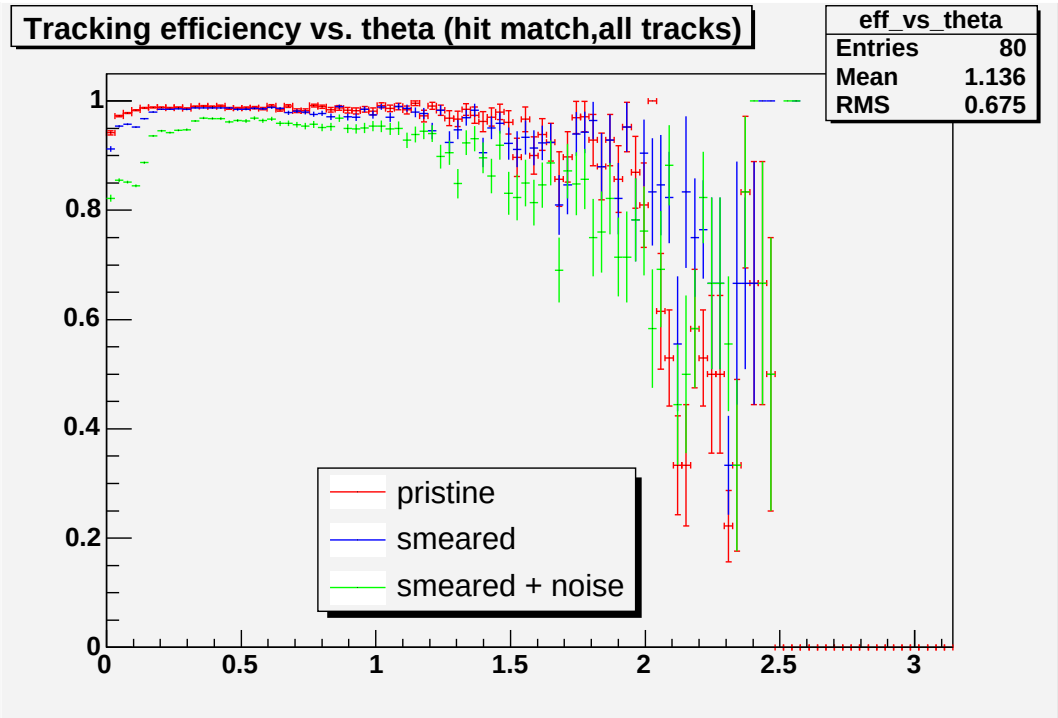

Figure 9: Track finding efficiency vs. μ θ angle(rad) at vertex for the three levels of data quality.

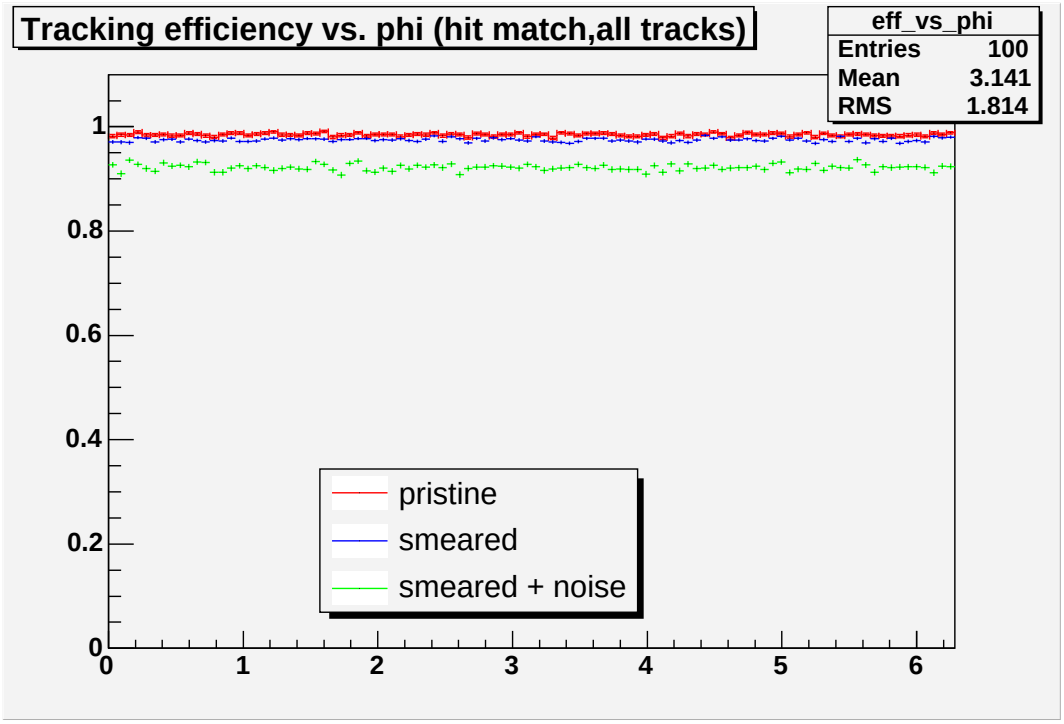

Figure 10: Track finding efficiency vs. μ ϕ angle(rad) at vertex for the three levels of data quality.

using a quad 2.4GHz Opteron machine with 3.5GB of RAM. For the tests, only track finding was done. No other reconstruction code was active.

The DANA analysis framework has the ability to process events using multiple threads. Running a single thread on a SMP machine will utilize only a single CPU while the others essentially sit idle. With 4 threads running, all 4 CPUs are used. The table below shows the event processing rate obtained for 4μ events for 1,2,3,4,5, and 20 threads. The row labled *pure* are rates obtained when processing the clean MC data. The row labled *noisy* are rates obtained when processing MC data whose values were smeared and noise hits were added.

	1 thread	2 threads	3 threads	4 threads	5 threads	20 threads
pure	357Hz	680Hz	994Hz	1348Hz	1155Hz	1087Hz
noisy	177Hz	346Hz	512Hz	663Hz	632Hz	584Hz

One encouraging result indicated by the above table is that the rate roughly scales with the number threads. At least until one has more threads than CPUs. Another interesting result is that the rates actually *decrease* when one uses more threads than there are CPUS. There are a few possible causes for this. Regardless of the root cause, we will need to optimize the number of threads based on the exact hardware utilized in the online systems.

According to the design report[9], a level-3 farm would need to process events at around 200kHz. The means at least 570 dual CPU boxes would be needed just for track finding. More optimization will clearly be needed.

6 FDC Package Positions

Among the suggestions made by the GlueX detector review committee was one concerning spacing of the FDC packages[5]. The suggestion was that if the first two packages were placed closer to the target, and the last two evenly spaced in the remaining space, it might improve the track finding efficiency for low momentum tracks. With the track finding and efficiency measurement code in hand, this suggestion was explored.

Using the hdds package[8] the position of the second FDC package was moved upstream by 39.5cm and the third by 21.5cm. Figure 11 shows an event drawn by hdview with the modified spacing.

Two simulations of 4μ events were done with 50k events each (200k tracks total). One with the nominal symmetric spacing and the other with the asymmetric spacing suggested by the review committee. The track finding efficiency vs. momentum and theta are shown in figure 12. As shown in the plots, the geometry change does not seem to affect the track finding efficiencies, even for the low momentum tracks where one might expect an improvement. The reason for this is most likely because the CDC adequately detects the tight-looping, low momentum tracks before they even get to the first FDC package.

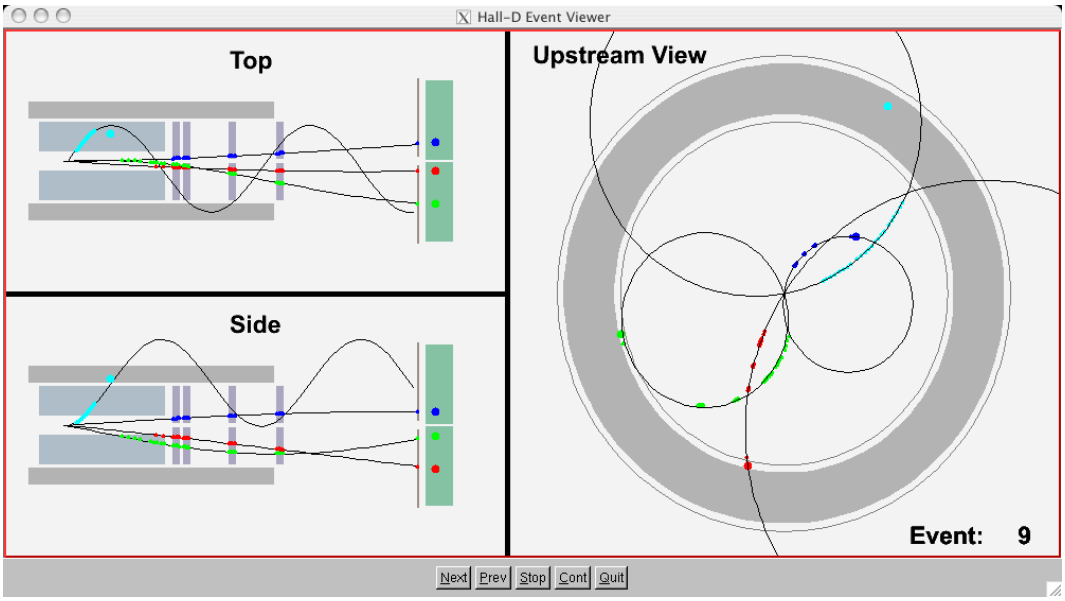

Figure 11: An event from the simulation with un-even spacing between FDC packages.

7 Tracking Code Organization

The tracking code was reorganized in accordance with a more formal plan for tracking in the final reconstruction code. Figure 13 shows a flow chart indicating the dependancies of the factories that make up the tracking package.

As indicated in the flowchart, Monte Carlo data can enter the track finding factory *DTrackCandidate* through two different paths. The one on the left derives all CDC and FDC hits from the truth tags and is what was used for the current study. When the first stage reconstruction factories are implemented for the CDC and FDC, the DTrackCandidate factory can get its data from there. In fact, a DANA configuration parameter[2] named "TRK:TRACKHIT_SOURCE" is used to determine which factory DTrackCandidate uses for its input. The parameter specifies the factory tag to use. At the time of this writing, the default is to use the DTrackHit factory with the "MC" tag. This will be changed to use the untagged factory when the DCDCHit and DFDCHit factories become available.

8 Still To Do

The track finding code currently does not support detached vertices. More studies are also needed to look at the tracking efficiency of different particle types (e.g. πs). Secondary tracks, a realistic magnetic field, and systematic geometry shifts all still need to be explored in more detail. A final track fit (likely with a Kalman filter) needs to be implemented as well.

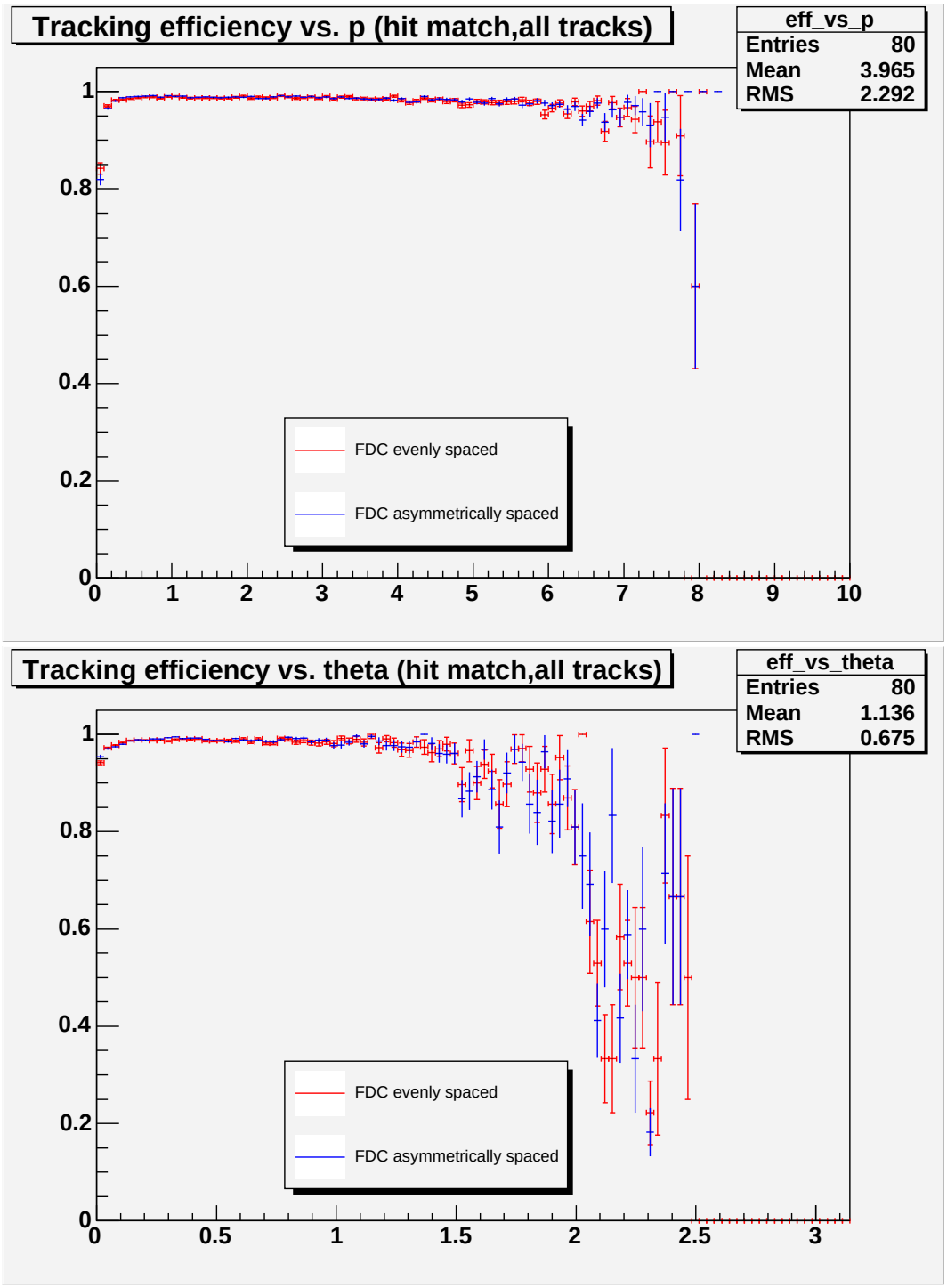

Figure 12: Efficiency vs momentum (top) and efficiency vs. theta (bottom) for two different FDC package spacings.

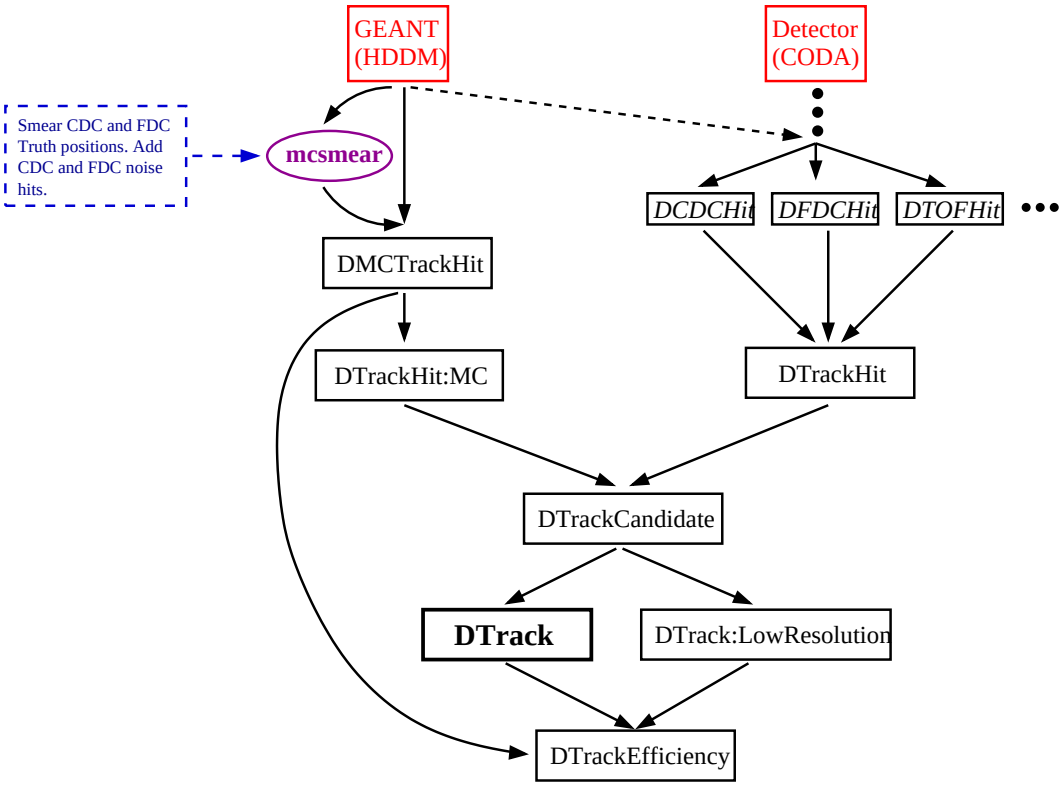

Figure 13: Dependancy of factories in TRACKING package of GlueX reconstruction code.

References

[1] D. Lawrence *Track Finding in 2-Dimensions* GlueX-doc-441 (2005)

[2] D. Lawrence *DANA Manual* GlueX-doc-527 (2005)

[3] GlueX Collaboration *GlueX Detector Review Mini-design Report* GlueX-doc-346 (2004)

[4] GlueX Collaboration *Brief Summaries of Detector Systems* GlueX-doc-323 (2004)

[5] Mike Albrow et al. *Report of the GlueX Detector Review Committee* GlueX-doc-381 (2004)

[6] R. Mankel *Pattern Recognition and Event Reconstruction in Particle Physics Experiments* Rep. Prog. Phys. **67**(2004) 553-622

[7] R. Frühwirth, M. Regler, R.K. Bock, H. Grote, D. Notz *Data Analysis Techniques for High-Energy Physics 2nd Ed.* Cambridge Monographs on Particle Physics, Nuclear Physics and Cosmology 11 (2000) chapter 2

[8] R. Jones *Geometry Specifications for Hall-D* GlueX-doc-61 (2003)

[9] GlueX Collaboration *The Science of Quark Confinement and Gluonic Excitations: GlueX/Hall D Design Report Version 4* Chapter 8 Nov (2002)