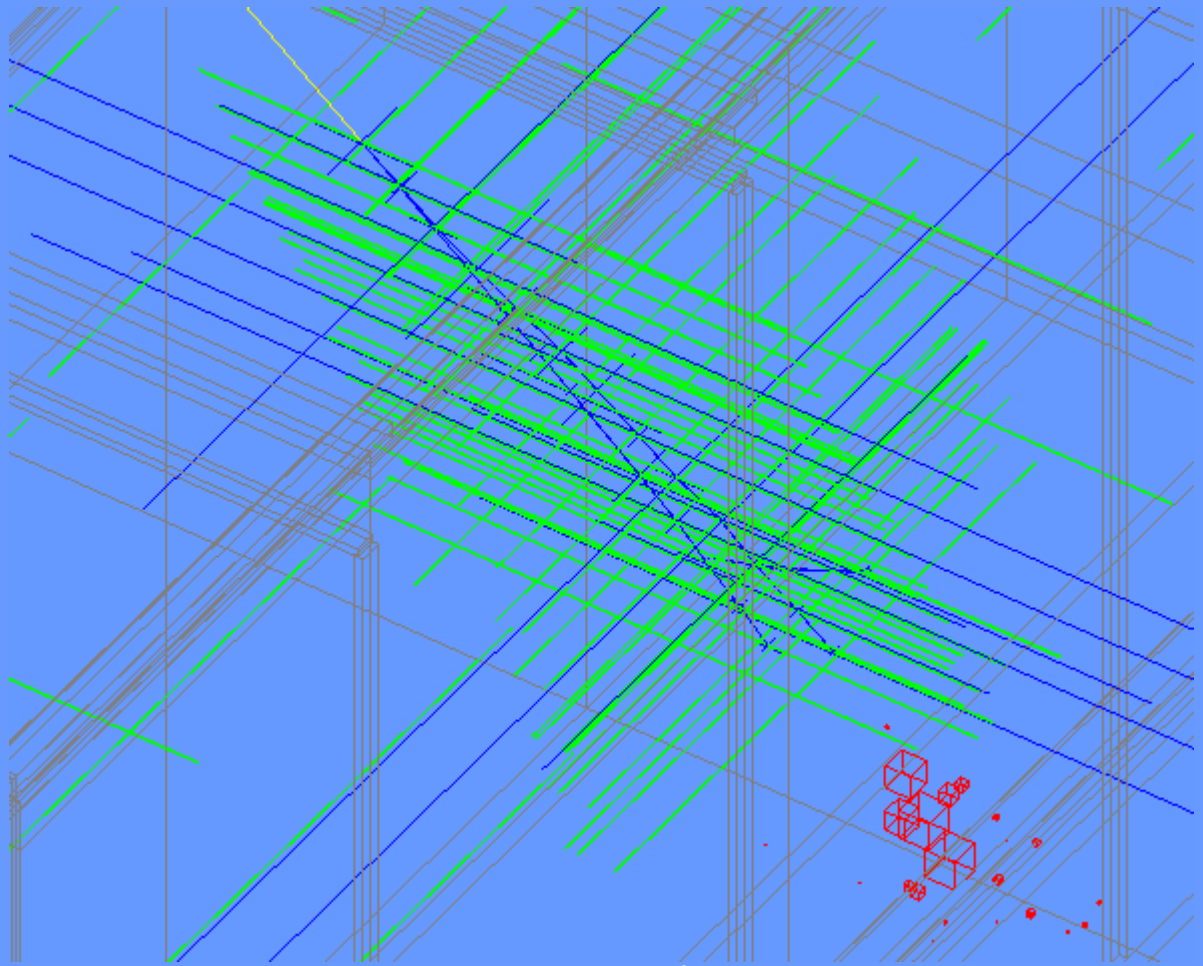
Instrument Analysis Workshop I

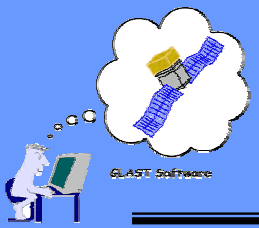
Overview of the TKR Reconstruction



Tracy Usher

June 7-8, 2004

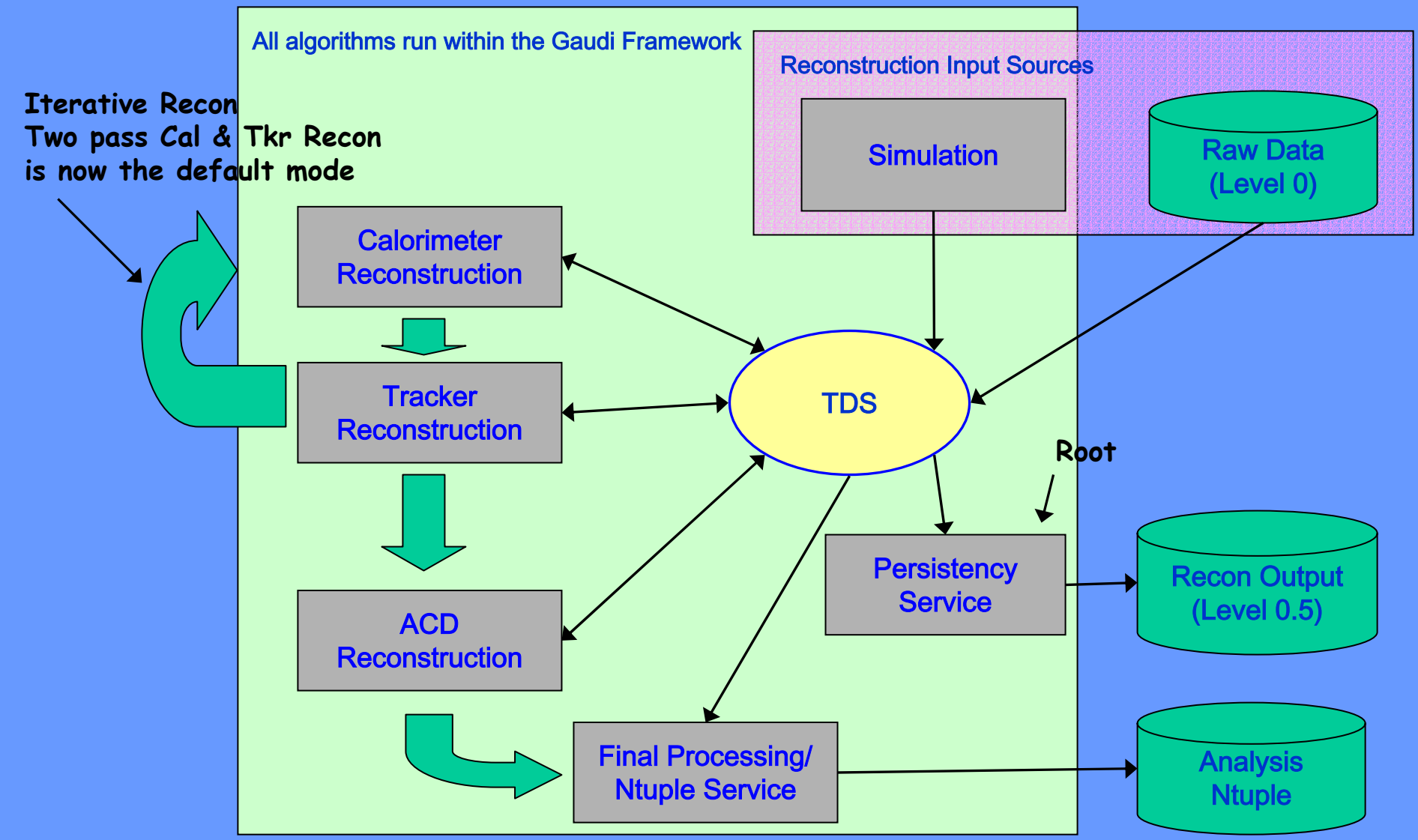
SLAC



Outline

- Overview of Reconstruction
- General TkrRecon Strategy
- Overview of Tracker Reconstruction
- Details of Clustering
- Details of Track Finding
- Details of Track Fitting
- Details of Vertexing

Reconstruction: Overview



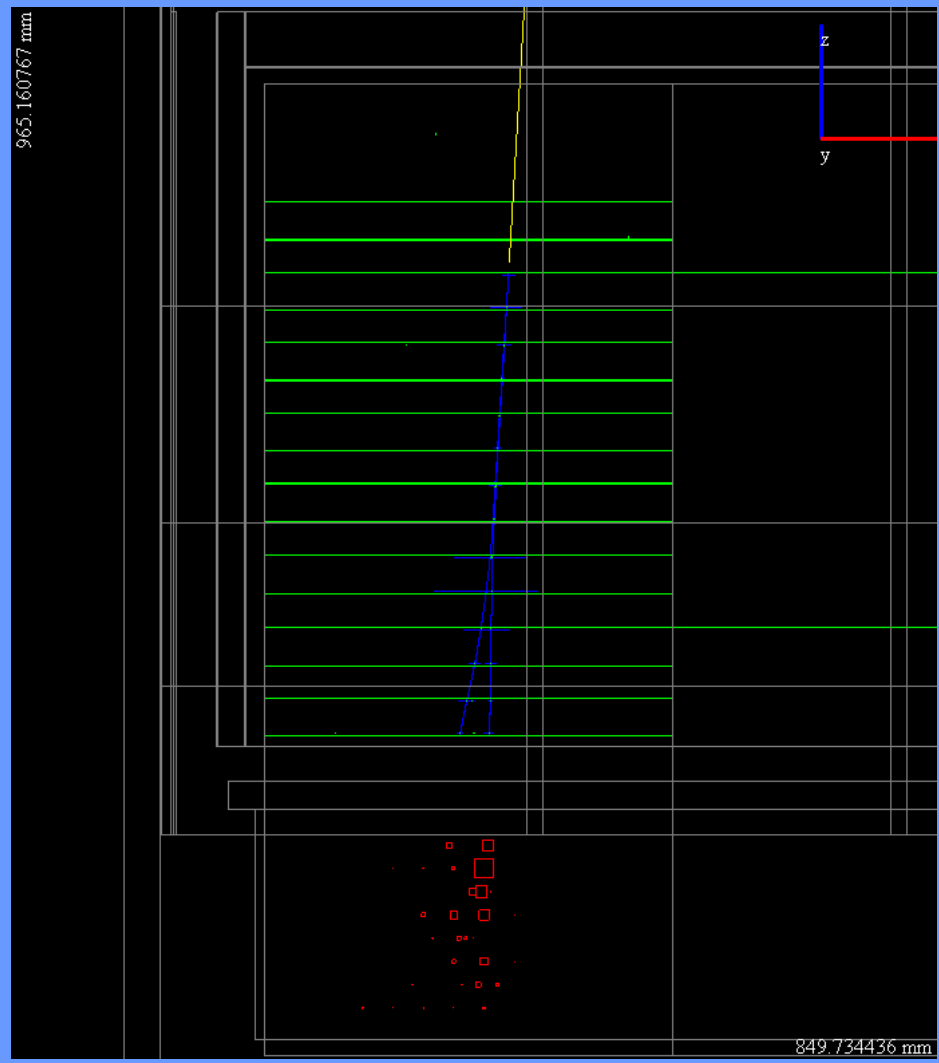
TkrRecon
General Strategy

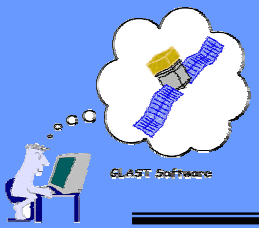
- What is desired:
  - Basic: determine the direction of incident gamma rays converting within the tracker
  - In addition: help improve the determination of the incident gamma energy

- Adopt four step procedure
  1) Form "Cluster Hits"
     - Convert hit strips to positions
     - Merge adjacent hit strips to form single hit
  2) Pattern Recognize individual tracks
     - Associate cluster hits into candidate tracks
     - Assign individual track energies
  3) "Fit" Track to obtain track parameters
     - 3D position and direction
     - Estimated errors on above
     - Track fit energy estimate
  4) "Vertex" fit tracks
     - Find common intersection point of fit tracks
     - Calculate position and resultant direction

- New twist (post DC-1)
  - "Iterative" Recon (see next slide)

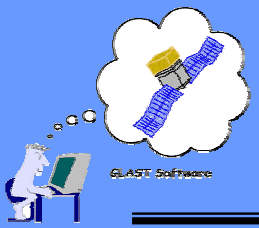
- Standard HEP approach to problem





Tracker Recon
Some Package Design Considerations

- Goal: Easy Interchangeability
  - Do we have the best solutions already?
  - Want to be able to quickly and easily switch implementations of various algorithms
- Implementation
  - Overall Algorithm to control Tracker Reconstruction at top level
  - Sub Algorithms to control each of four reconstruction steps
  - Gaudi Tools for specific implementation of particular task
  - Use TDS to store intermediate results – communicate between algorithms
- Have not fully achieved goal – yet
  - Current structure not as transparent as would like
  - TDS objects need to be simplified
  - TkrRecon undergoing critical review amongst tracking folks now



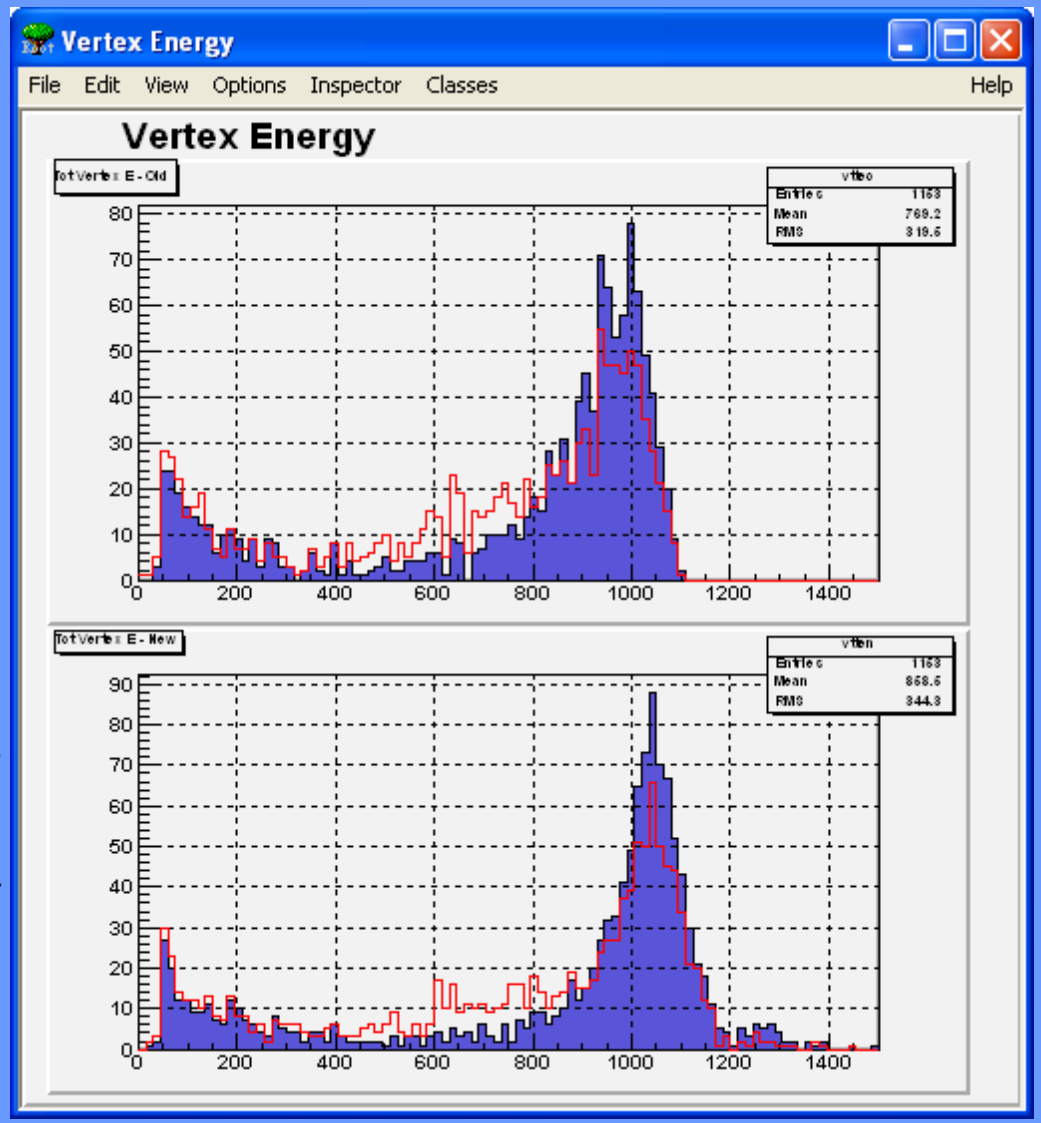
Iterative Tracker Recon
What and Why

- What is it?
  - The Iterative Recon is a mechanism for allowing parts of the Tracker Reconstruction software to be called more than once per event
  - In particular, existing pattern recognition tracks can be refit and the vertex algorithm re-run
- Why is it needed?
  - The Calorimeter would like the output of TkrRecon when running the energy correction algorithms.
  - At the same time, TkrRecon wants the best energy estimate from the Calorimeter to get the best track fits and, subsequently, the best vertices
  - The Iterative Recon solves this problem by providing the Calorimeter Recon with sufficient tracking information to get an improved energy estimate, which can be fed back to the track fit and vertexing algorithms
  - The process can be repeated as many times as the user likes (in principal)

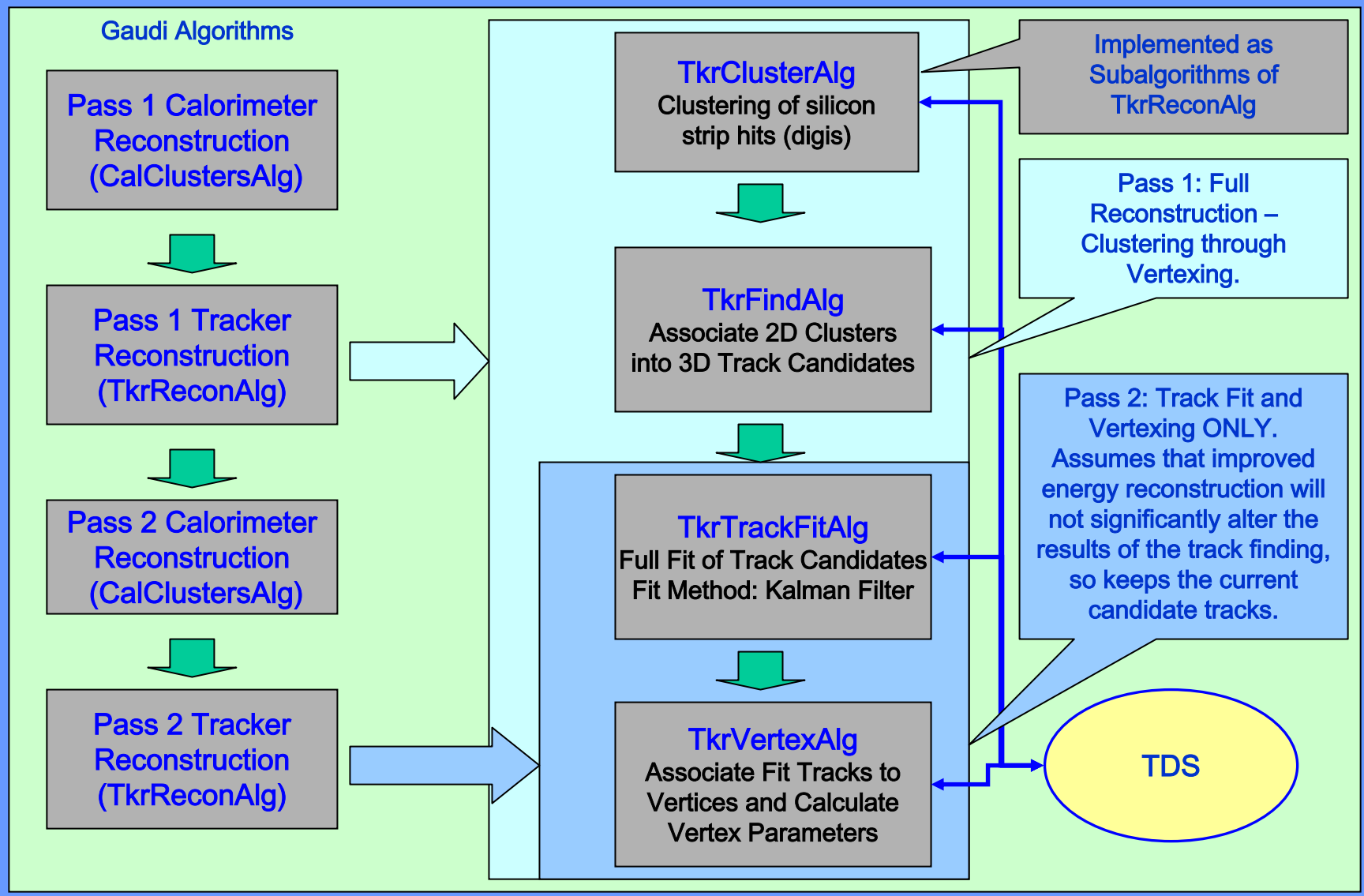
Iterative Tracker Recon

Example of Expected Improvement to Recon

- Example using "WAgammas"
  - 1 GeV
  - 5° cone about normal
  - Into 6 m^2 area containing Glast
- Plot Energy of the reconstructed vertex
  - Red Histograms - energy of "best" vertex
  - Blue Histograms – include energy of second track if not part of vertex
- In General
  - Reconstructed vertex energy improves
  - Some details to be understood
    - Shift in energy above 1 GeV probably still fallout (in this version of Gleam at least) of Tkr/Cal displacement
    - High energy tail?



Iterative Tracker Recon
Overview

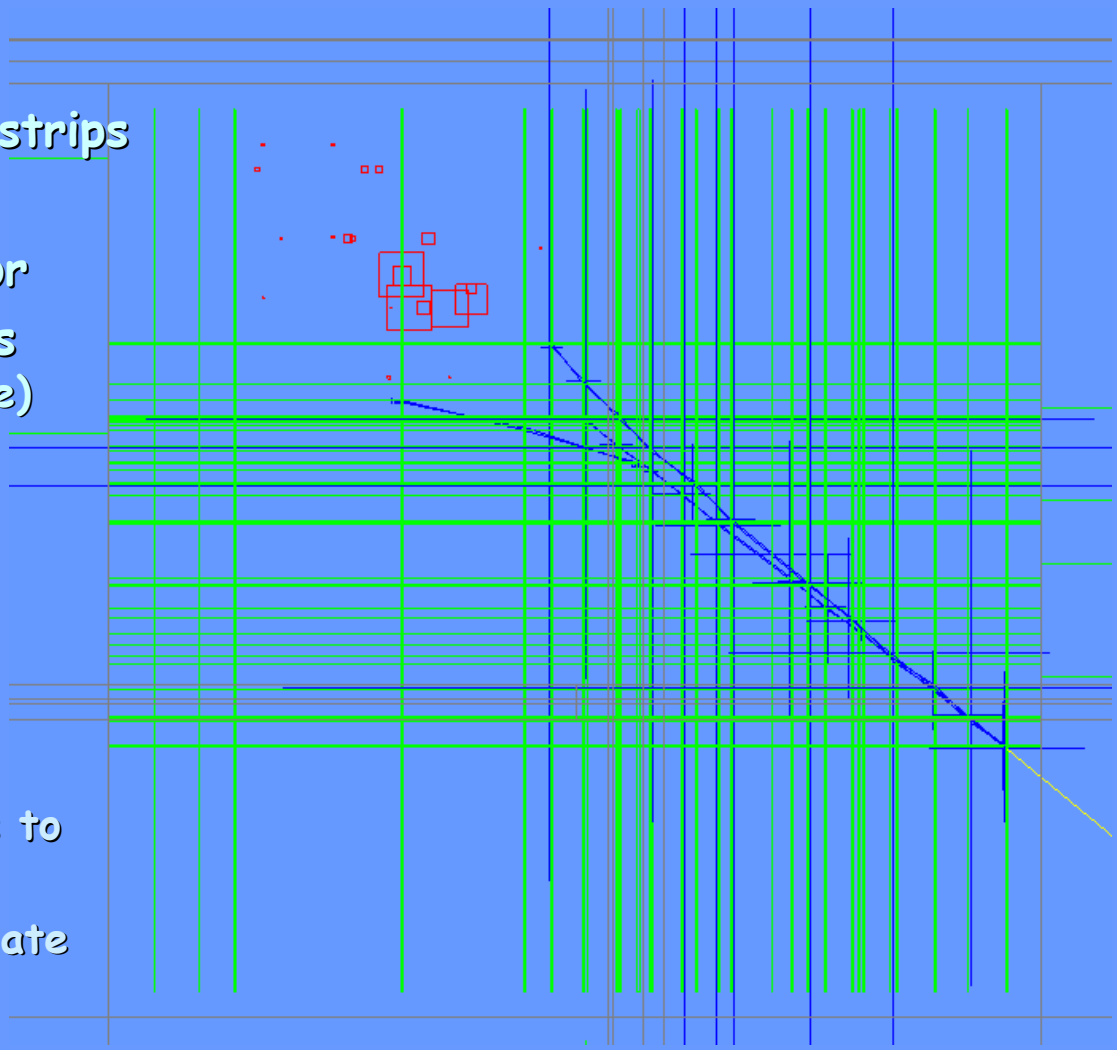


Strip Clustering TkrClusterAlg

- Top View looking down
- Green lines represent hit strips
- Blue lines are fit tracks
- Yellow line is vertex vector
- Red boxes are hit crystals (with some mc drawing license)

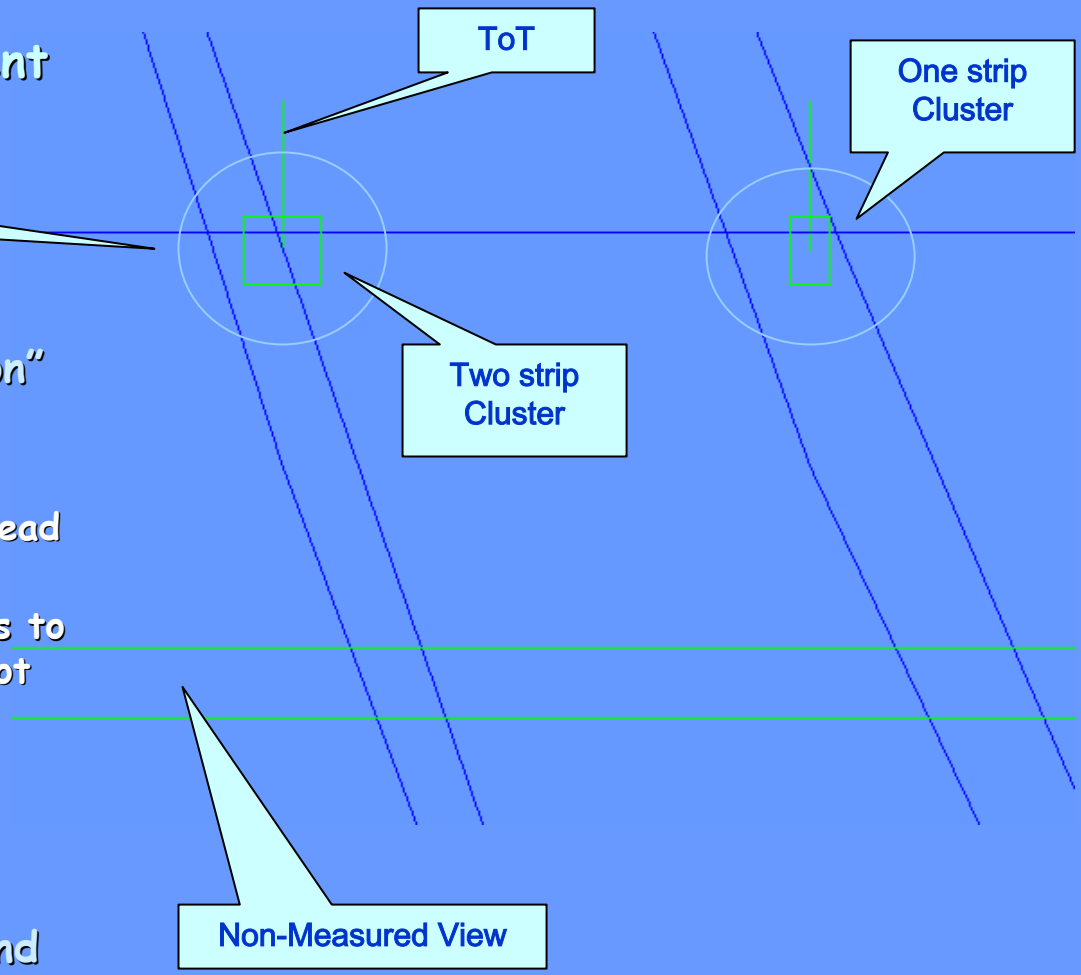
- Digis give to clustering:
  - Hit Strip Numbers
  - ToT

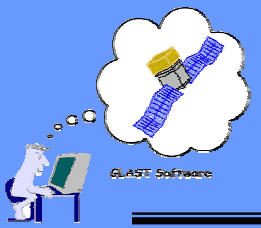
- Basic job of clustering:
  - Group adjacent hit strips to form cluster
  - Given hit strip ids, calculate cluster center position



Strip Clustering TkrClusterAlg

- Close up YZ view of same event from previous page
- Additional Clustering work:
  - Apply the Tracker "calibration" to account for hot/dead/sick strips
    - Merge clusters with known dead strips between them
    - Decide whether to add strips to clusters when known to be hot (this part tricky)
    - Etc.
- Result of TkrClusterAlg:
  - List of Clusters in TDS with associated xyz coordinates and value of ToT associated with these strips

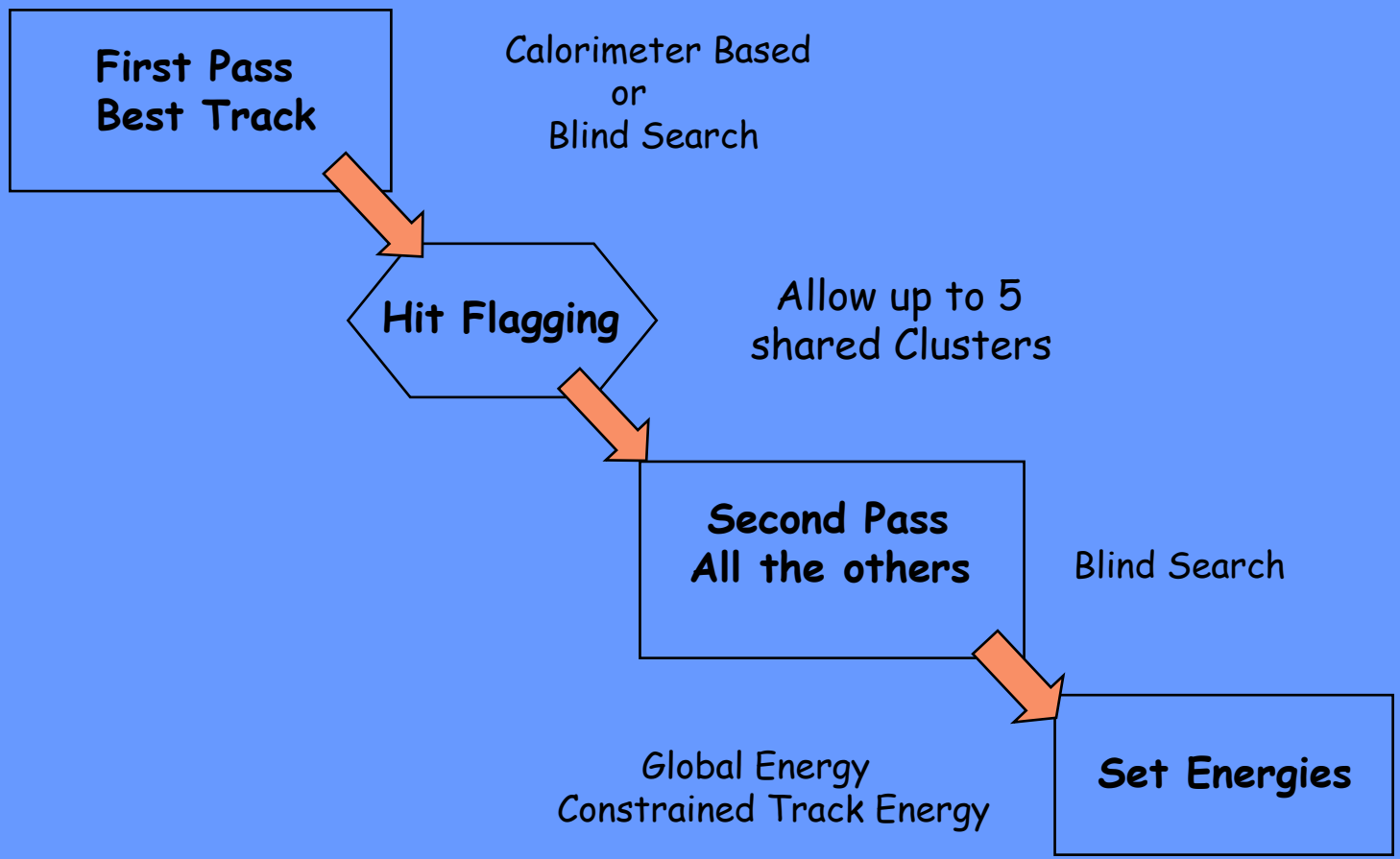




Pattern Recognition – Track Candidates
TkrFindAlg

- The next step of TkrReconAlg is to associate clusters into candidate tracks
  - Note that clusters are inherently 2D
  - Track finding must yield 3D track candidates
- Three approaches exist within the TkrRecon package
  - "Combo" – Combinatoric search through space points to find candidates
  - "Link & Tree" – Associate hits into a tree like structure
  - "Neural Net"
    - Link nearby space points forming "neurons"
    - link neurons by rules weighting linkages.
  - Also: "Monte Carlo" pattern recognition exists for testing fitting and vertexing
- "Combo" method is "track by track"
  - Pro's: Simple to understand (although details add complications)
  - Con's: Finding "wrong" tracks early in the process throws off the rest of the track finding by mis-associating hits.
    Also, can be quite time consuming depending upon the depth of the search
- "Link and Tree" and "Neural Net" are "global" pattern recognition techniques
  - Pro's: Optimized to find tracks in entire event, less susceptible to mis-associating hits
  - Con's: Both methods can be quite time consuming. In addition, "Link and Tree" (in its current implementation) operates in 2D and then requires mating to get 3D track
- The "Combo" method is (still) the most advanced and best understood method
  - It is the default for GLAST reconstruction

Combinatoric Pattern Recognition
Overview



Combinatoric Pattern Recognition
ComboFindTrackTool

Starting Layer: One furthest from the calorimeter

Two Strategies:

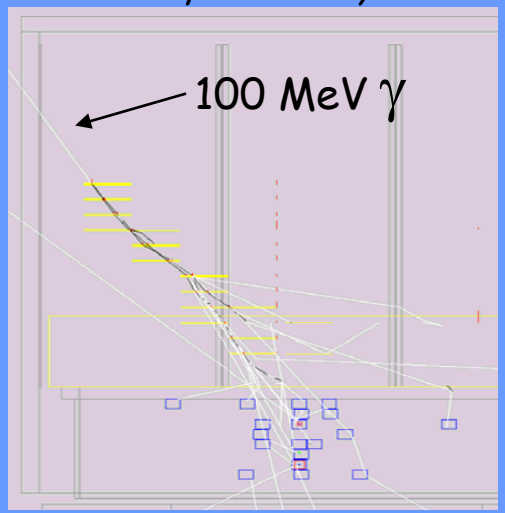
1) Calorimeter Energy present ⇒ use energy centroid (space point!)
2) Too little Cal. Energy ⇒ use only Track Hits

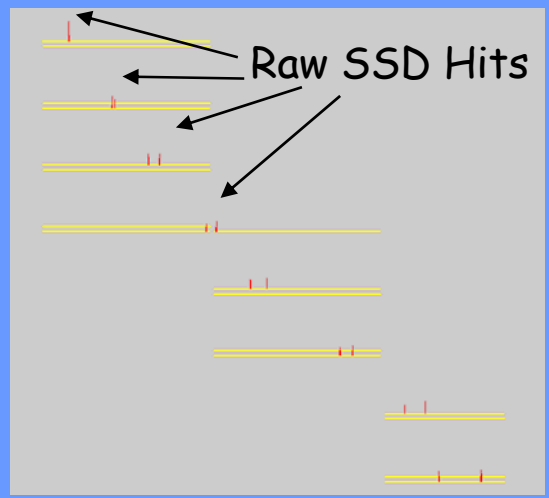
Work with 3D hits:

- Combine clusters in adjacent x-y layers to form 3D space points

"Combo" Pattern Recognition - Processing an Example Event:

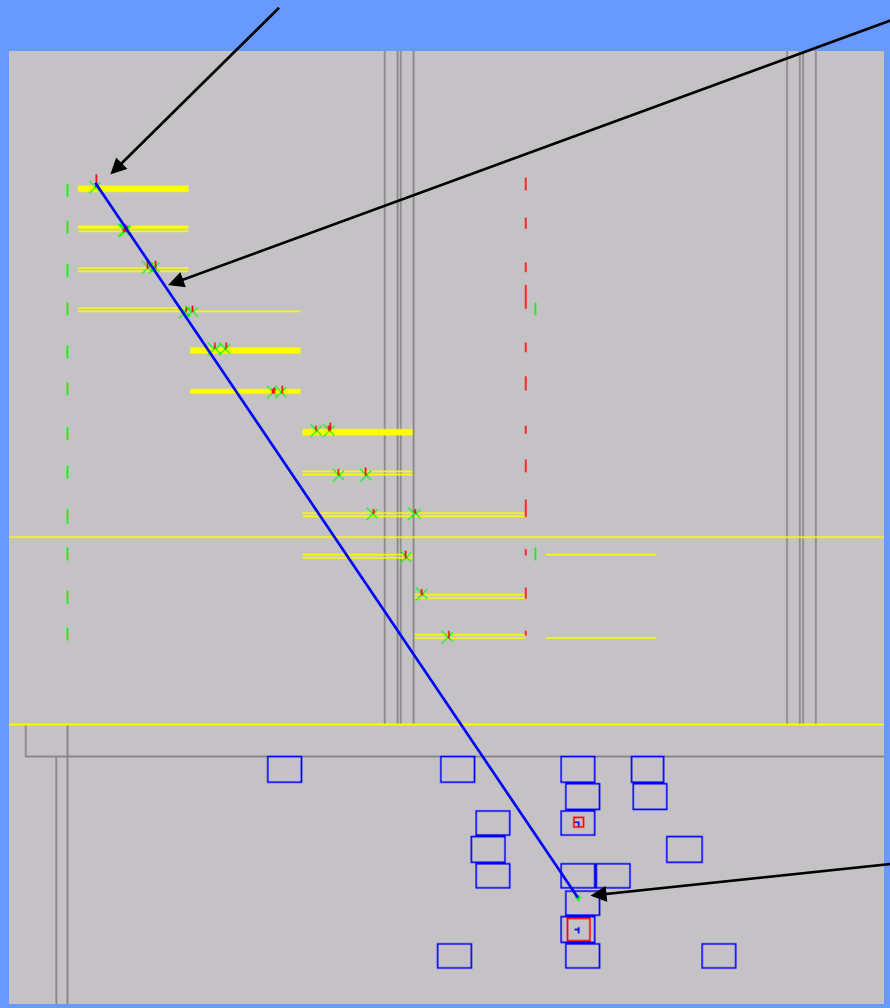
The Event as produce by GLEAM



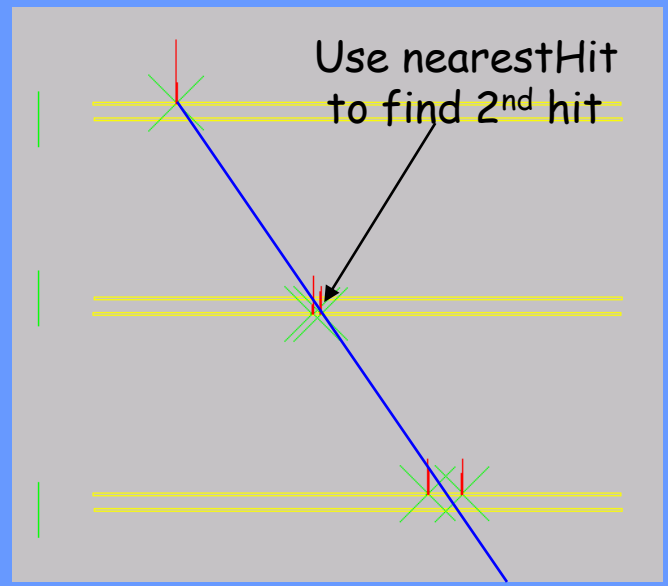


Combinatoric Pattern Recognition

ComboFindTrackTool



First Guess: connect hit with Cal. Centroid



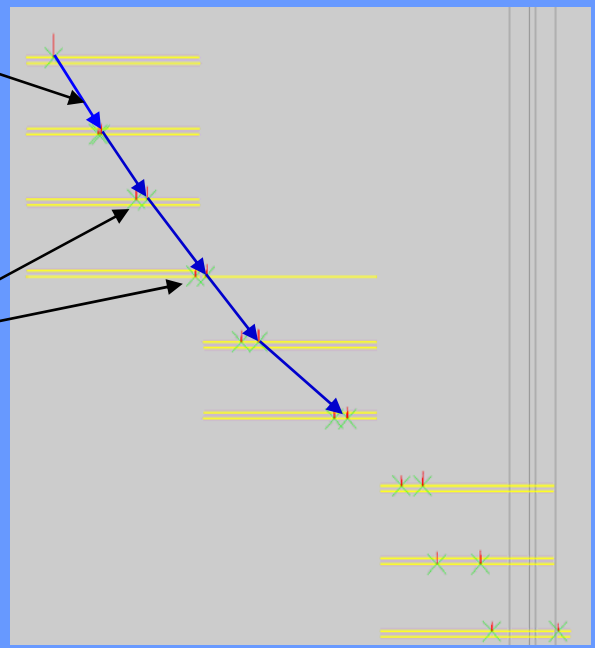
Sufficient Cal. Energy (42 MeV) Use Cal. centroid

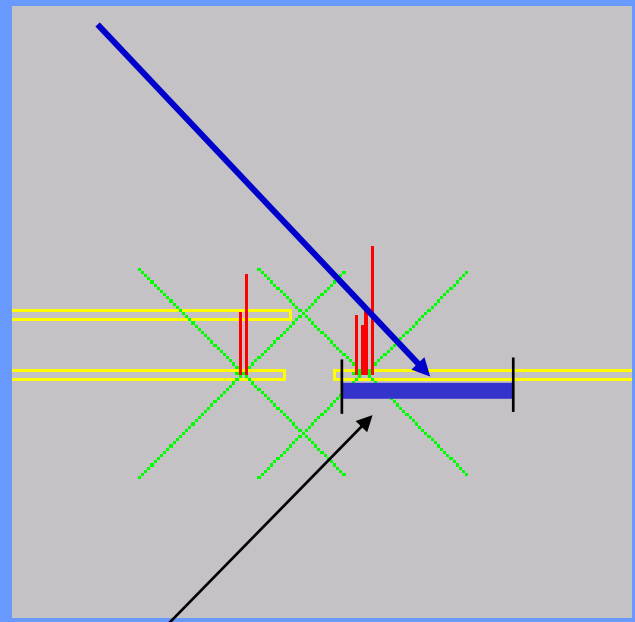
Combinatoric Pattern Recognition

ComboFindTrackTool

Initial Track Guess:
Connect first 2 Hits!

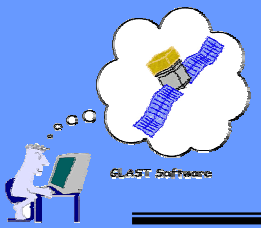
Project and Add Hits Along the Track within Search Region





The search region is set by propagating the track errors through the GLAST geometry.

The default region is 9σ (set very wide at this stage)



Combinatoric Pattern Recognition

ComboFindTrackTool

The Blind Search proceeds similar to the Calorimeter based Search

- 1st Hit found found - tried in combinatoric order
- 2nd Hit selected in combinatoric order
- First two hits used to project into next layer -
- 3rd Hit is searched for -
- If 3rd hit found, track is built by "finding - following" as with Calorimeter search

In this way a list of tracks is formed.

Crucial to success, is ordering the list!

Combinatoric Pattern Recognition
ComboFindTrackTool

Track Selection Parameter Optimization

Ordering Parameter

Q = Track-Quality - C_1* Start-Layer - C_2*First-Kink - C_3*Hit-Size - C_4*Leading-Hits

Track_Quality: "No. Hits" - χ^2 : track length (track tube length) - how poorly hits fit inside it

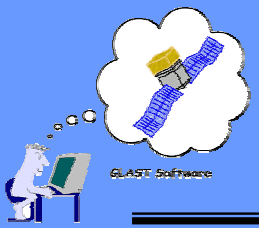
Start-Layer: Penalize tracks for starting late

First-Kink: Angle between first 2 track segments / Estimated MS angle

Hit-Size: Penalize tracks made up of oversized clusters (see Hit Sharing)

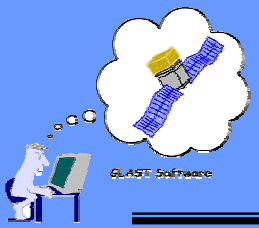
Leading-Hits: These are unpaired X or Y hits at the start of the track. This protects against noise being preferred.

Status: Current parameters set by observing studying single events. Underway – program to optimize parameters against performance metrics underway (Brian Baughman)



Track Finding Output

- An ordered list of candidate tracks to be fit
  - TkrPatCand contained in a TkrPatCandCol Gaudi Object Vector
    - Estimated track parameters for candiate track (position, direction)
    - Energy assigned to the track
    - Track candidate "quality" estimate
    - Starting tower/layer information
  - Each TkrPatCand contains a Gaudi Object Vector of TkrPatCandHits for each hit (cluster) associated with the candidate track
    - Cluster associated to this hit, needed for fit stage
  - All stored in the TDS



Track Fit
TkrFitTrackAlg

- The next step of TkrReconAlg is to Fit the candidate tracks
  - Track parameters
    - x, m_x (slope in x direction), y, m_y
  - Track parameter error matrix (parameter errors and correlations)
  - Track energy (augment calorimeter estimate with fit info)
  - Measure of the quality of the track fit
  - etc.

- Track Fit Method: use a Kalman Filter

- Extra unnecessary details
  - Actually have four fitters in TkrRecon package:
    - Three based on Kalman Filter
      - KalFitTrack: also includes hit finding methods for Combo Pat Rec
      - KalFitter: Pure track fit using same underlying fit as above
      - KalmanTrackFitTool: a "generic" fitter used to cross check the above
    - 2D Least Squares fitter
      - "Classic" straight line fit

Track Fitting:
The Kalman Filter

The Kalman filter process is a successive approximation scheme to estimate parameters

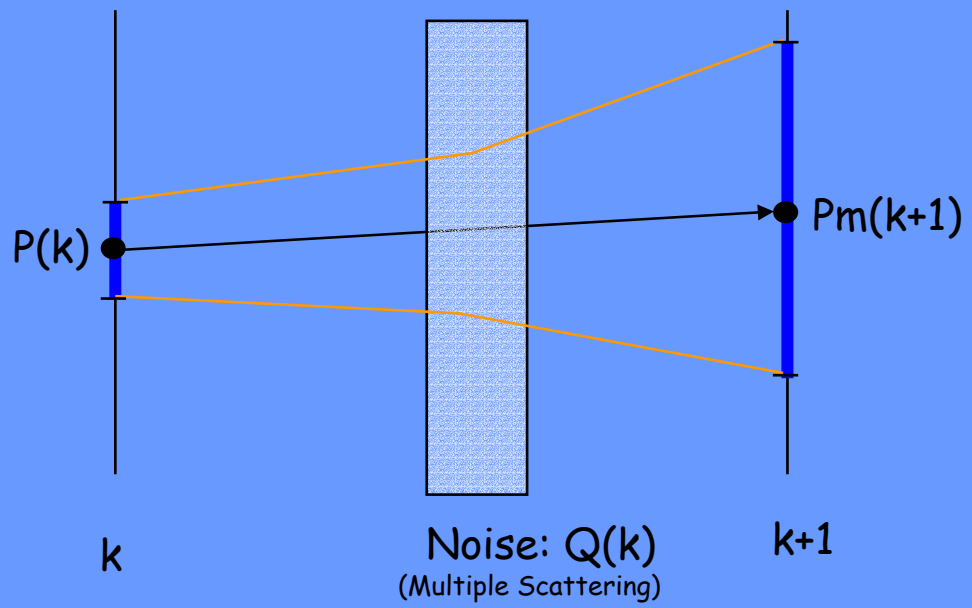
Simple Example: 2 parameters - intercept and slope: $x = x_0 + S_x * z$; $P = (x_0 , S_x)$

Errors on parameters x_0 & S_x (covariance matrix): $C = \begin{pmatrix} Cx\text{-}x & Cx\text{-}s \\ Cs\text{-}x & Cs\text{-}s \end{pmatrix}$

$Cx\text{-}x = \langle (x-x_m)(x-x_m) \rangle$

In general

$C = \langle (P - P_m)(P-P_m)^T \rangle$



Propagation:

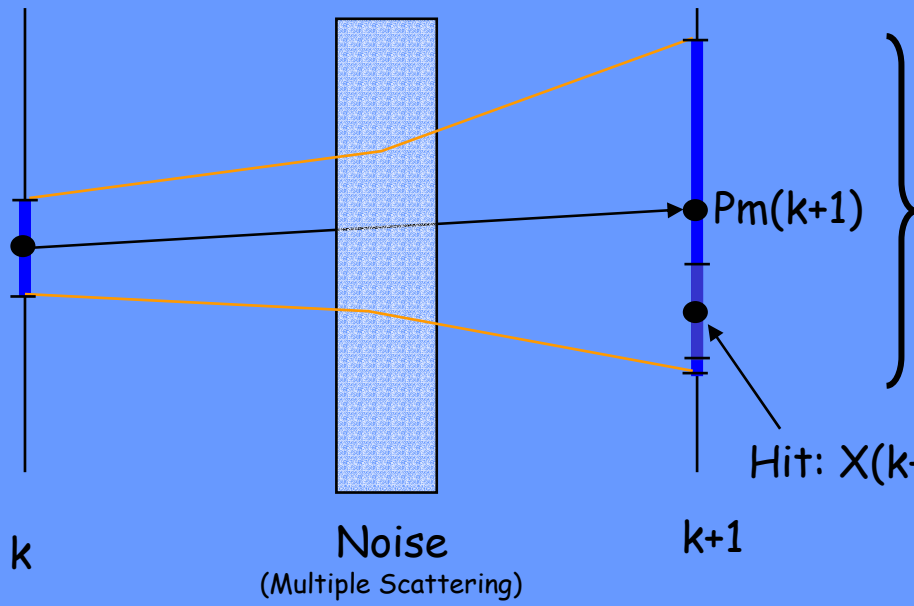
$x(k+1) = x(k)+Sx(k)*(z(k+1)-z(k))$

$Pm(k+1) = F(\delta z) * P(k)$ where

$$F(\delta z) = \begin{pmatrix} 1 & z(k+1)-z(k) \\ 0 & 1 \end{pmatrix}$$

$$Cm(k+1) = F(\delta z) * C(k) * F(\delta z)^T + Q(k)$$

Track Fitting:
The Kalman Filter



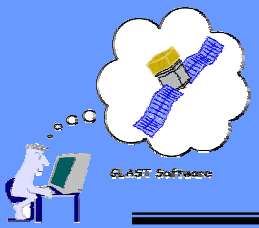
Form the weighted average of the k+1 measurement and the propagated track model: Weights given by inverse of Error Matrix: C^{-1}

Hit: X(k+1) with errors V(k+1)

$$P(k+1) = \frac{Cm^{-1}(k+1)*Pm(k+1) + V^{-1}(k+1)*X(k+1)}{Cm^{-1}(k+1) + V^{-1}(k+1)}$$

and $C(k+1) = (Cm^{-1}(k+1) + V^{-1}(k+1))^{-1}$

Now its repeated for the k+2 planes and so - on. This is called FILTERING - each successive step incorporates the knowledge of previous steps as allowed for by the NOISE and the aggregate sum of the previous hits.



Track Fitting:
The Kalman Filter

We start the FILTER process at the conversion point

BUT… We want the best estimate of the track parameters at the conversion point.

Must propagate the influence of all the subsequent Hits backwards to the beginning of the track - Essentially running the FILTER in reverse.

This is call the SMOOTHER & the linear algebra is similar.

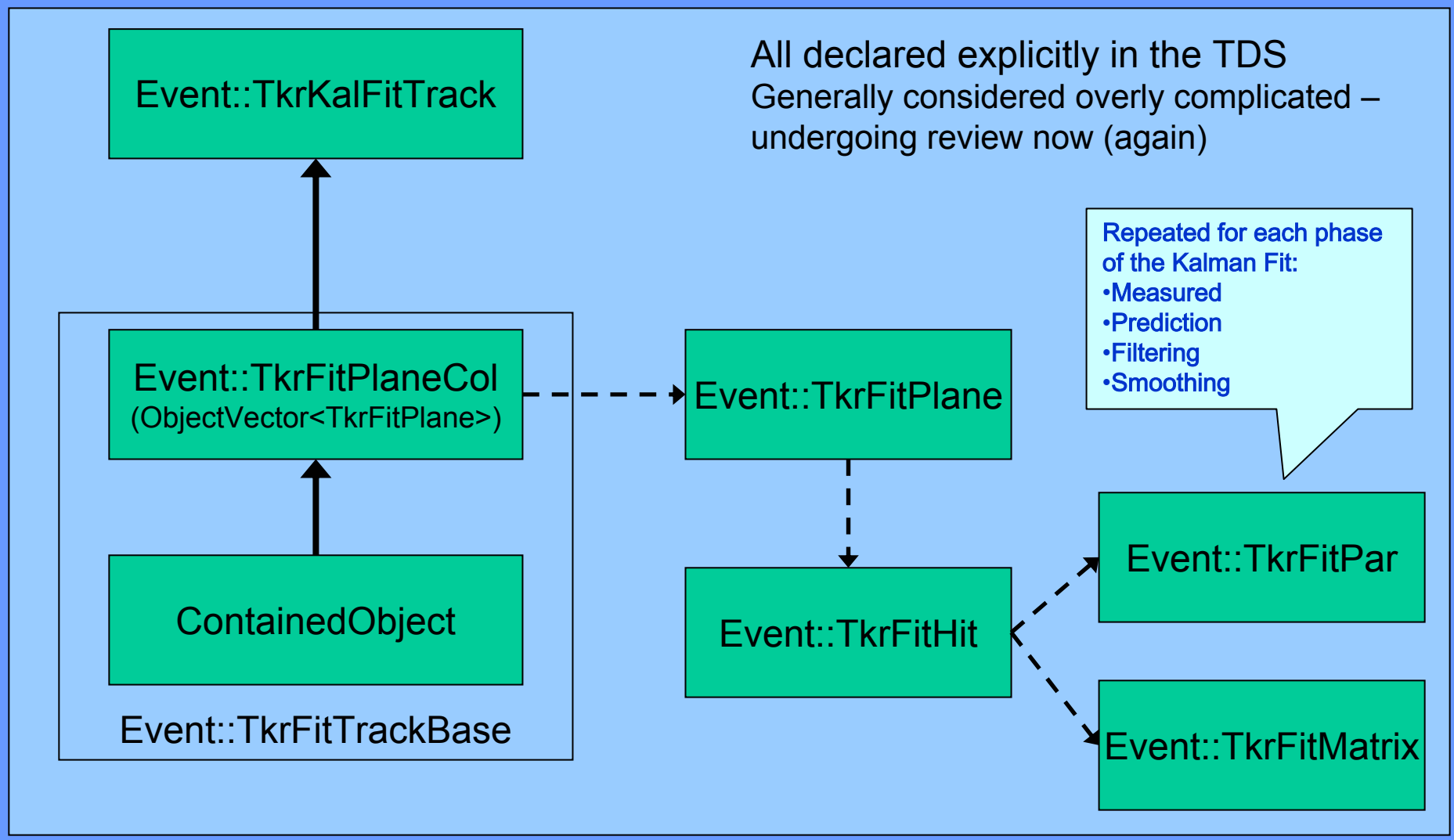
Residuals & χ^2:

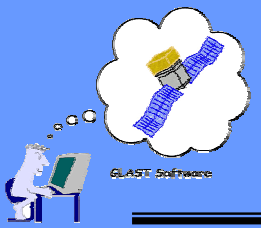
Residuals: $r(k) = X(k) - Pm(k)$

Covariance of r(k): $Cr(k) = V(k) - C(k)$

Then: $\chi^2 = r(k)^T Cr(k)^{-1} r(k)$ for the k^{th} step

Track Fit Output



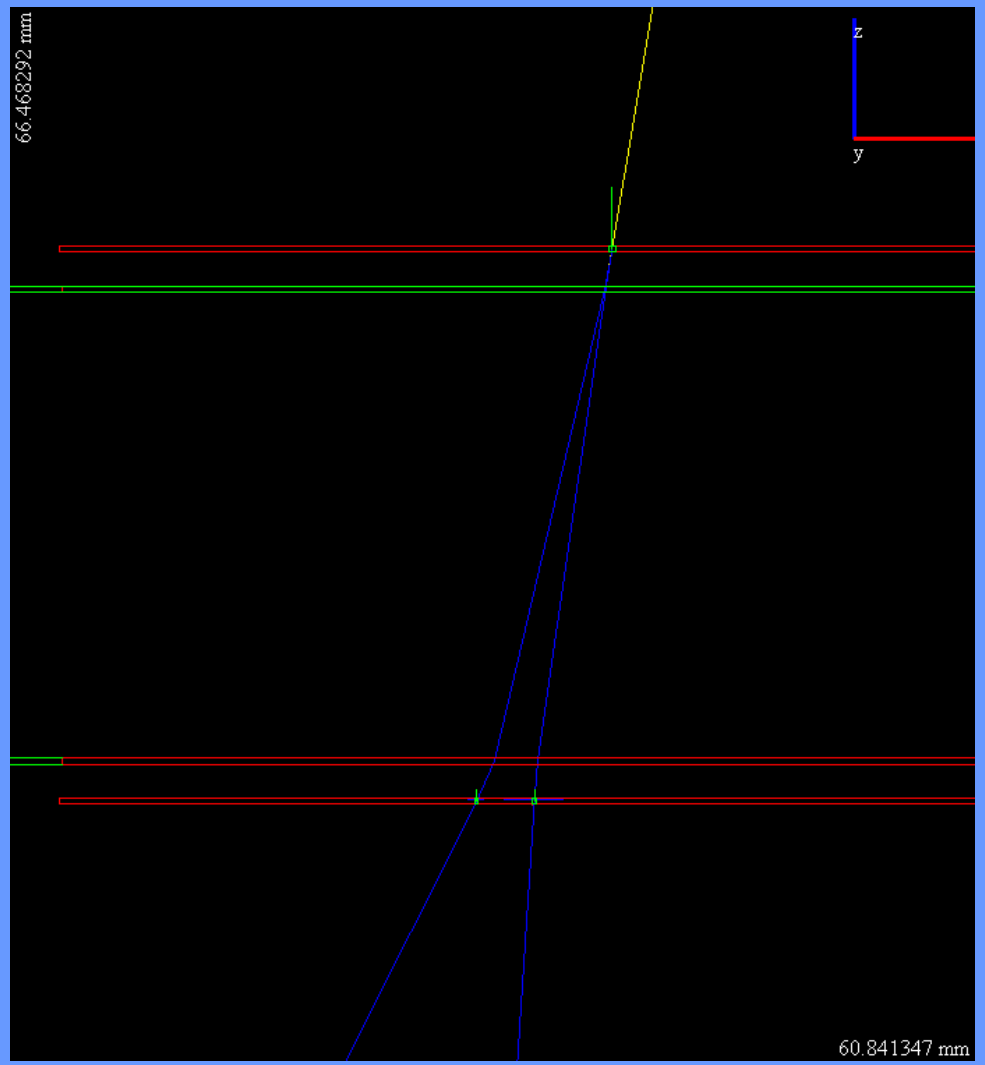


"Vertexing"
TkrVertexAlg

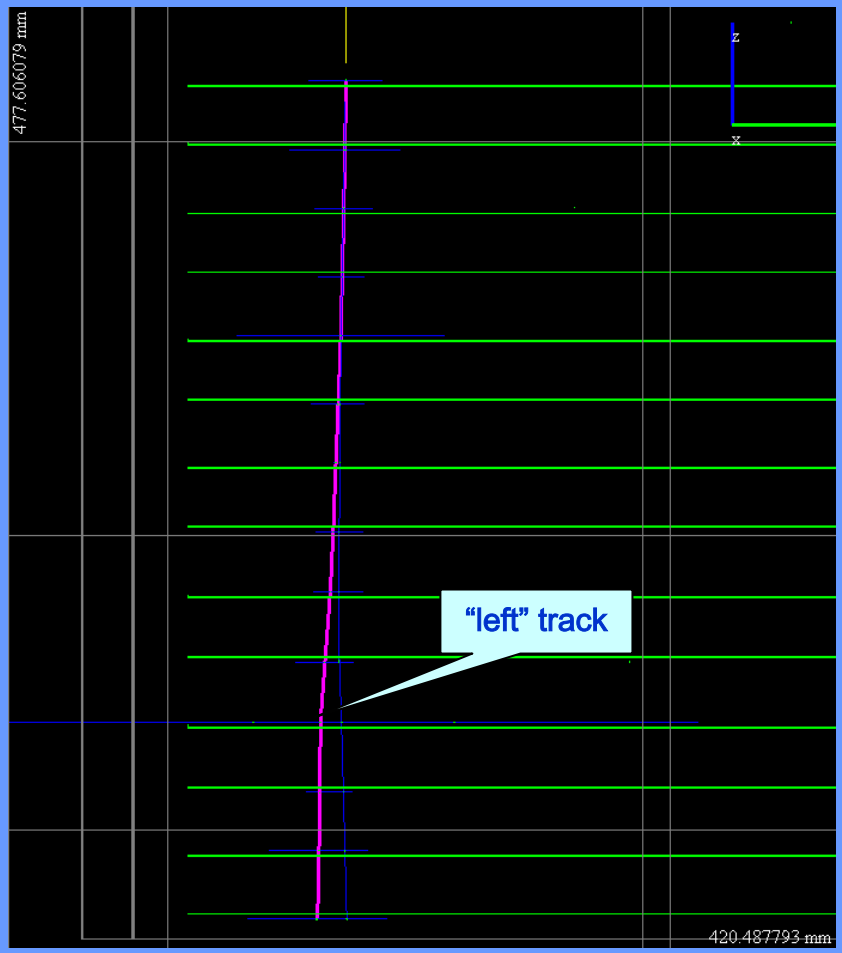
- Assume Gamma conversion results in 1-2 primary tracks
  - 1 track in pair conversion could be lost
    - Too low energy (track must cross three planes)
    - Tracks from pair conversion don't separate for several layers
      - Track separation due to multiple scattering
    - etc.
  - Secondary tracks associated with primary tracks, not part of gamma conversion process
- Task of the vertexing algorithm
  - Attempt to associate "best" track (from track finding) with one of the other found tracks
    - Finds "the" gamma vertex
    - Unassociated ("Isolated") tracks returned as single prong vertices
  - Determine the reconstructed position of the conversion
  - Determine the reconstructed direction of the conversion
- Currently two methods available:
  - "Combo" Vertex Tool (the default for Gleam)
    - Uses track Distance of Closest Approach (DOCA) to associate tracks
  - Kalman Filter Vertex Tool (Not used, still under development)
    - Uses a Kalman Filter to associate tracks

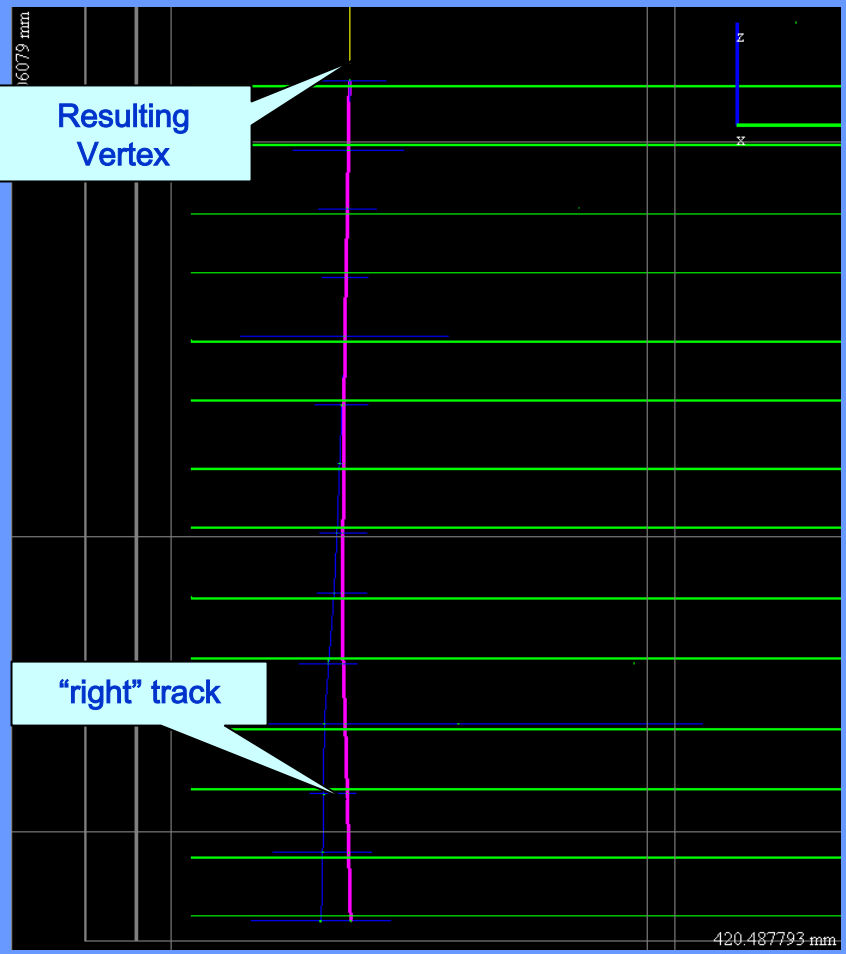
"Combo" Vertex Reconstruction

- Vertex determined at two track Distance of Closest Approach (DOCA)
  - First track is "best" track from track finding/fitting
  - Loop through "other" tracks looking for best match:
    - Smallest DOCA
    - Weighting factor
      - Separation between the starting points of the two tracks
      - Track energy
      - Track quality
- Calculate Vertex quantities
  - Vertex Position
    - Midpoint of DOCA vector
  - Vertex Direction
    - Weighted vector sum of individual track directions
- Not a true HEP Vertex Fit



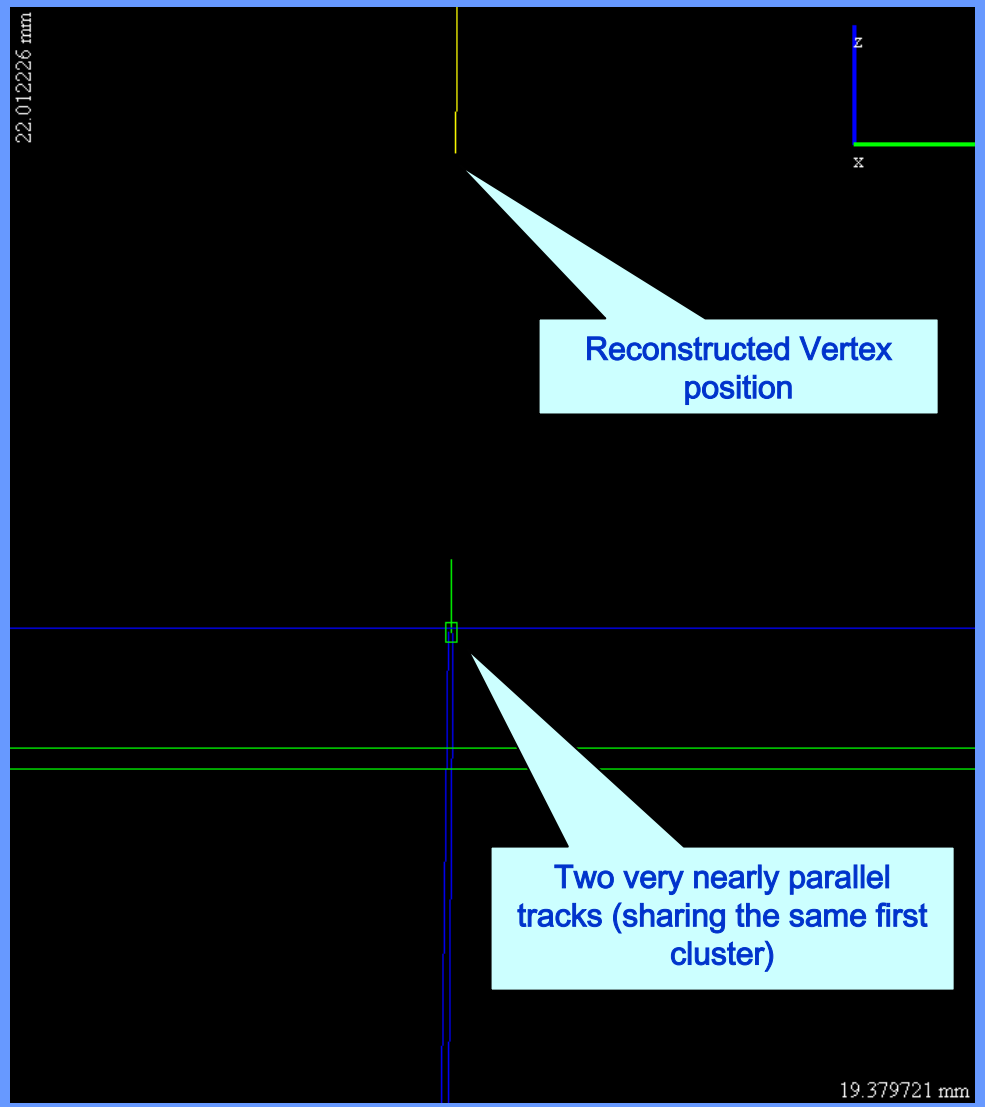
ComboVtxTool





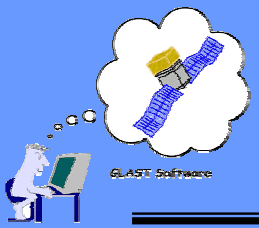
DOCA Vertexing
Illustration of Parallel Track Problem

- Parallel (or close) tracks can be problematic
  - Direction probably very good
  - Vertex position can easily be significantly off
- (from previous example!)



Vertexing Output

- An ordered list of vertices
  - TkrVertex objects contained in a Gaudi Object Vector
    - Vertex Track Parameters
      - (x, m_x, y, m_y)
    - Vertex Track Parameter covariance matrix
    - Vertex Energy
    - Vertex Quality
    - First tower/layer information
    - Gaudi Reference vector to the tracks in the vertex
  - First Vertex in the list is the "best" vertex
    - Mostly the rest are associated with "isolated" tracks



Overview of TkrRecon
The End

- Very General Overview of TkrRecon
  - Hopefully provides a roadmap for people to start digging in
  - Current structure of Algorithms, Tools and interfaces is somewhat complicated
  - Hope to simplify this over the summer as we prepare for DC-2
- No Discussion of Output Ntuple
  - Personal bias: Can't do detailed studies of Tracker performance from ntuple alone
    - Must dig into the TDS/PDS output
- No Discussion of Monte Carlo relational tables
  - A separate talk by itself
- Don't be afraid to ask questions!
  - More eyes looking at things is good
  - Valuable feedback on how to improve things