PFA status

Mat Charles
U. Iowa

Talk overview

- Executive summary
    - The goalposts
    - Algorithm overview
    - Getting & running the PFA
    - Performance snapshot
- More detailed walk-through of algorithm
- Next steps & a note on manpower

Goals (1)

- In a nutshell, we want a PFA that:
  - is realistic
  - has good enough performance to do the physics
  - can help us make technology choices
- As rough figure of merit, use dijet mass residuals for $e^+e^- \rightarrow Z(\nu\nu)\ Z(qq)$ @ 500 GeV for sid01 (q=uds), $|\cos\vartheta|<0.8$
- Proposed in previous workshops as OK for physics:

$$dM/M \sim 3\% - 4\%$$

$$\text{i.e. } dM \leq 3.6 \text{ GeV or so}$$

Goals (2)

- What is achievable?

Perfect pattern recognition (PPR): 2.35 GeV

And track matched to cluster: 2.54 GeV

And passes E/p veto: 3.41 GeV

DTree clustering: 2.8 GeV

And track matched to cluster: ??? GeV

And passes E/p veto: ??? GeV
Guesstimate: about 3.5-3.9 GeV

Right now, goal is to push resolution down to ~ 4.0 GeV.

(Once we're there, can think about going further...)

Algorithm overview

- Find photons. Set to one side.
- Run DTreeClusterer on remaining hits
- Within each DTreeCluster, look for substructure
  - MIP segments, clumps, etc
  - Define score to link them based on geometric quantities
  - Fuzzy clustering for individual / small-cluster / halo hits
- Extrapolate tracks to calorimeter, match to “seed” clusters
- Build charged showers outwards from seeds
  - Use links based on score, E/p (complicated -- more detail later!)
  - Optionally, apply hard E/p veto after final clustering
- Build neutral hadron showers from remaining clusters

Getting & running the PFA

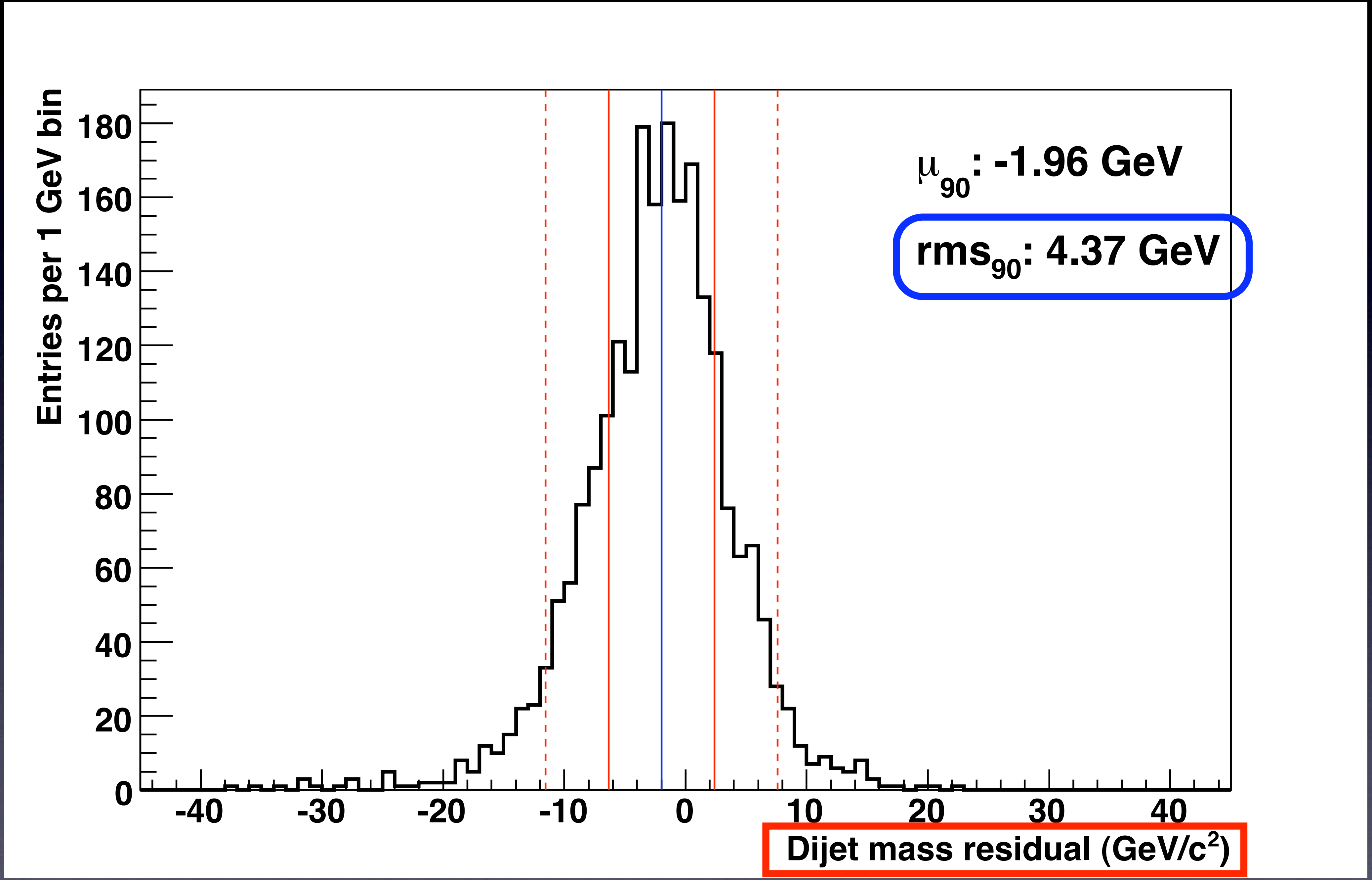
- Get up-to-date CVS checkout & build
- Minimal code
  (see example at org.lcsim.pfa.structural.RunAndWriteOutPFA)

```
add(new NonTrivialPFA());
add(new SetUpDTreeForReclustering());
add(new ReclusterDTreeDriver("DTreeClusters",
"FSReconTracks", "ReconFSParticles"));
```

- Output lists:
  - DTreeReclusteredParticles
  - DTreeReclusteredParticles_withEoverPveto ← the one you want
  - DTreeReclusteredParticles_forConfusionMatrix
- Additional code to flush & write out needed
  - Again, see org.lcsim.pfa.structural.RunAndWriteOutPFA

Current performance

$e^+e^- \rightarrow Z(\nu\nu)\ Z(qq)$ @ 500 GeV for sid01 (q=uds), $|\cos\vartheta|<0.8$



For comparison: 4.61 GeV in March (with old calibration)
4.87 GeV at January SLAC workshop
5.46 GeV at October FNAL workshop (NonTrivialPFA)

Algorithm in detail

Track extrapolation

- Not using real tracking -- but trying to keep things realistic.
- Start with Ron's FSReconTracks list of tracks which are reconstructible
  - Find position of 3 outermost truth hits in tracker
  - Extrapolate to calorimeter as a helix
  - Look for matching cluster (next slides) in ECAL -- preferably a MIP
- This works pretty well most of the time, but can go wrong for:
  - Low-momentum tracks
  - Loopers
  - Material interactions / decays in flight near ECAL
- ... but then, those cases are hard for real tracking too.
- Make a note of any unmatched tracks for later.

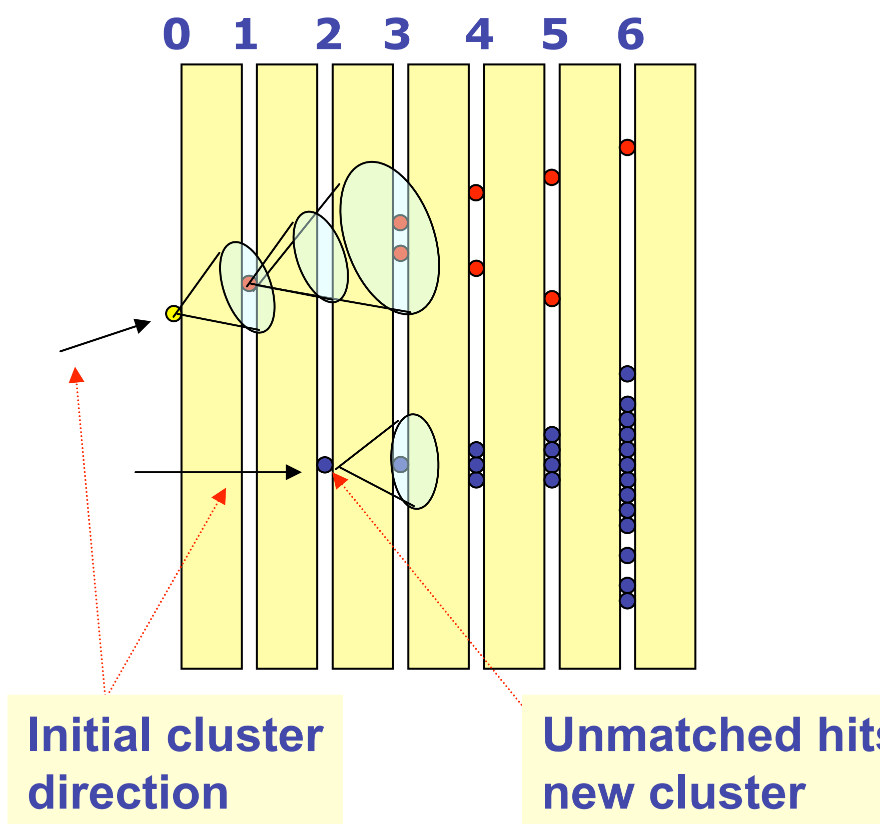
Finding photons

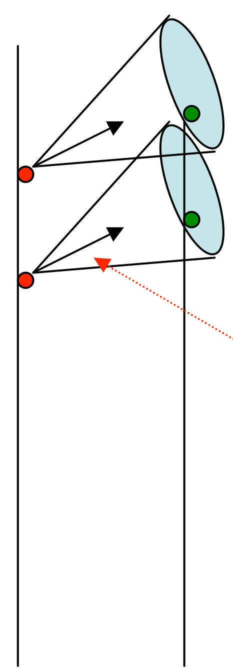
- Use Ron's updated photon finder -- efficiency & purity around 90%
- Check if there is a track directly connected to the photon
  - If no track match, treat as pure photon and NEVER assign hits to a charged shower.
  - If track match and cluster E = track p, treat as electron
  - If track match and cluster E ≠ track p, split up photon into pieces (see next slide)
    - Aim to handle case of nearby/overlapping photon & track
    - Piece directly connected to tracks is treated as charged
    - Fuzzy clustering for tiny pieces (<4 hits)
    - Rest is treated as photon

Photon splitting algorithm

ii) ECAL/HCAL Clustering

- ★ Start at inner layers and work outward
- ★ Tracks can be used to "seed" clusters
- ★ Associate hits with existing Clusters
- ★ If no association made form new Cluster
- ★ Simple cone based algorithm





Simple cone algorithm based on current direction + additional N pixels

Cones based on either:
initial PC direction or
current PC direction

Parameters:
- cone angle
- additional pixels

I use a ~ 90° cone and allow it to skip one layer.

Neighbours of seed in same layer also picked up.

Idea taken from Mark Thomson, but implemented separately in org.lcsim

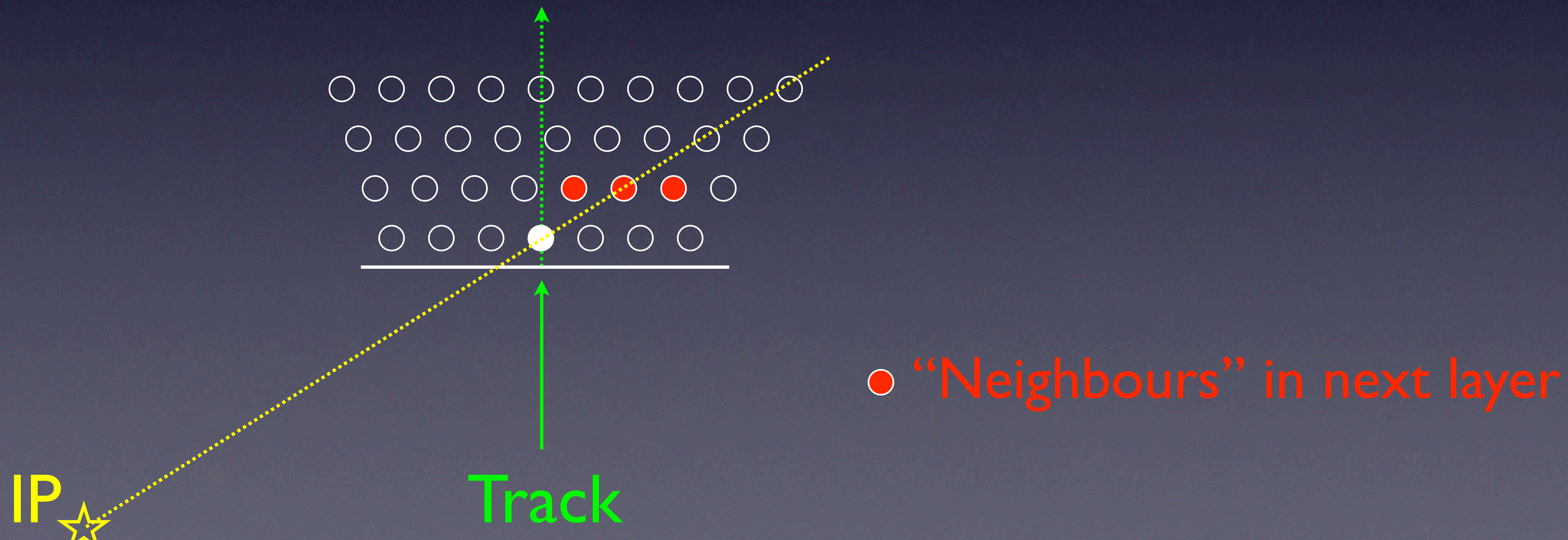
Structure in DTrees

- Apply DTree clusterer to remaining (non-photon) hits
  - Goal here is to find "envelopes" for showers so that fuzzy/halo hits get assigned back to their parent properly
- Inside each DTree cluster, look for structure:
  - Clumps with internal structure:
    - MIPs & MIP-like segments (projective & non-projective) [see next slide]
    - Clumps (high local hit density)
    - Other hits (share with nearby pieces in same DTree)
  - Large clumps without internal structure treated as blocks (>= 20 hits in ECAL, >= 15 hits in HCAL)
  - Smaller clumps shared with nearby clusters (fuzzy)

Finding MIPs (1)

`org.lcsim.recon.cluster.mipfinder.TrackClusterDriver`

- Primary MIP finder is very simple:
  - Use only isolated or semi-isolated (∂) hits
  - Look for neighbouring hits in sequential layers
- ... but because "neighbouring" is defined in a projective way, that can fail for non-projective or curved tracks.



Finding MIPs (2)

`org.lcsim.recon.cluster.mipfinder.NonProjectiveMipFinder`

- Work-around: Second pass with a different MIP-finder
- Taking (semi-)isolated hits, find 3-hit stubs in sequential layers
- Expand outwards from both ends of stub
  - Linear extrapolation based on outermost hits
  - Cuts on direction, distance to next hit
  - Allowed to skip a layer from time to time
- Lots of fiddly code to merge/pair stubs & resolve ambiguities
- Only run it within a DTree cluster -- combinatorics are too awful to run it on an entire event

Notes on scoring

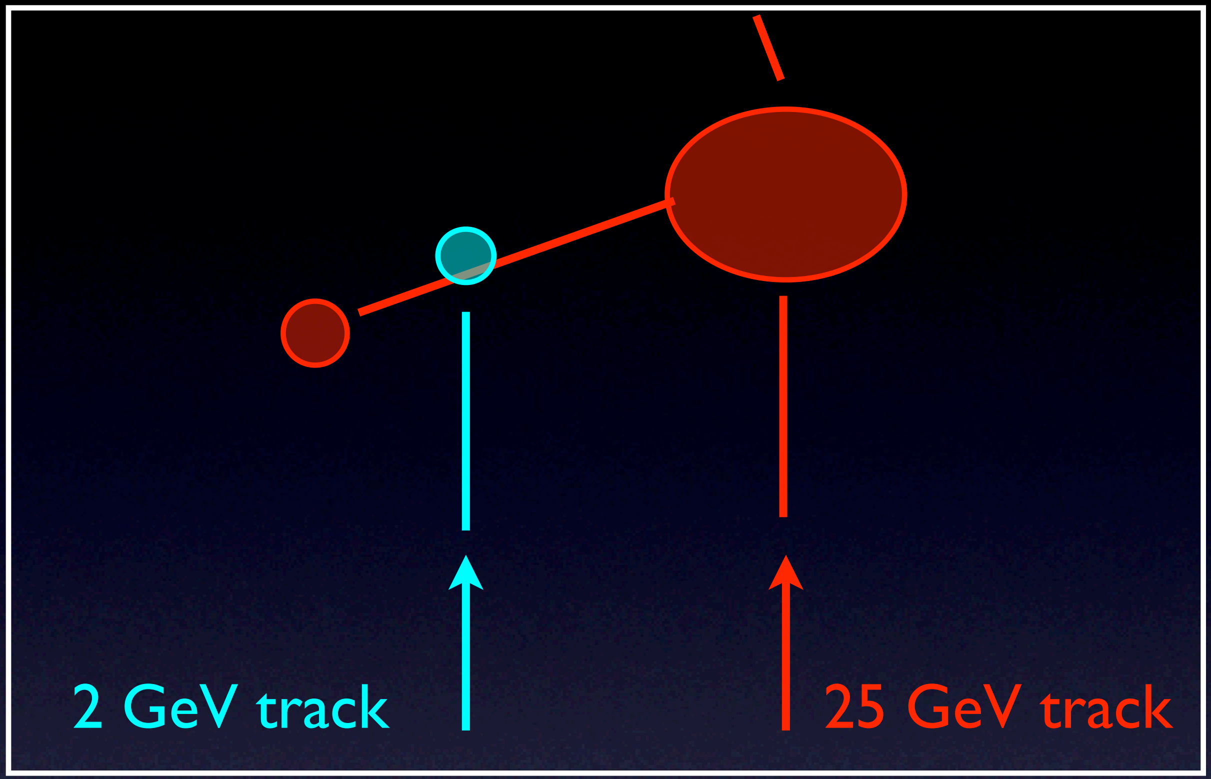
- For any given pair of clusters, can define a score between 0 and 1 to describe how likely they are to be connected
- MIP-MIP and MIP-clump links use likelihood computed with geometrical quantities like proximity, DOCA, etc.
  - Penalty if not both inside same DTree cluster
- MIP links to other things (larger clusters treated as blocks, seed photons, seed small clusters) based on proximity and pointing
- Clump-clump, clump-smallSeed, block-block, block-smallSeed links based on proximity & opening angle
- Some link types not included (e.g. photon seed to block)
- Scoring algorithms & constants based mostly on my judgement -- may not be fully optimal.

Building charged showers (1)

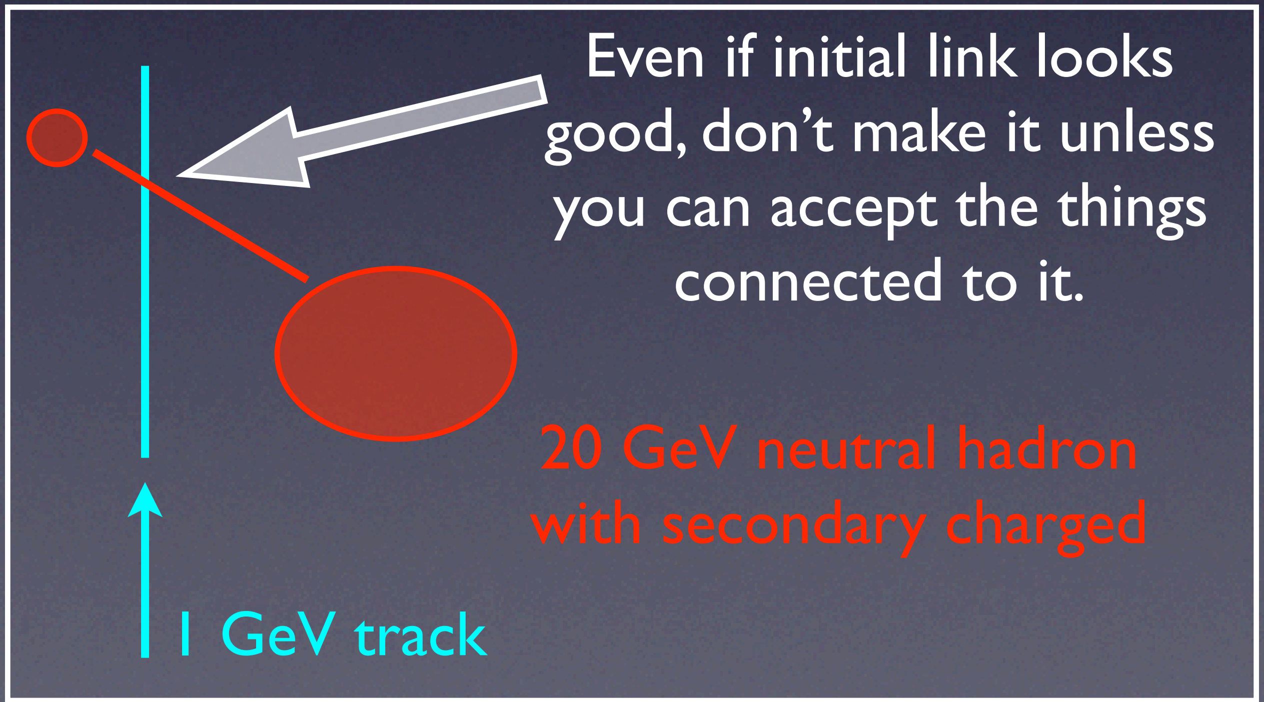
- Basic idea: build shower up from seed following simple rules:
  - Start with lowest-momentum tracks and work up
  - Don't take any clusters already assigned to a lower-momentum track ➡
  - Pick up highest-rated links first as long as they have score > threshold
  - If picking up a cluster x by following a link with score s_x, also pick up any unassigned clusters connected to x with score >= s_x ➡
  - Stop when no more links available with
    - score > threshold
    - $E_{clus} < \sqrt{(p^2+m^2)}$ + tolerance after following link
  - Add more clusters if possible (see slide after next)
- If shower incomplete (E << p), loosen threshold/tolerance and iterate.

Illustrations

- Don't take any clusters already assigned to a lower-momentum track



- If picking up a cluster x by following a link with score s_x, also pick up any unassigned clusters connected to x with score >= s_x



Building charged showers (2)

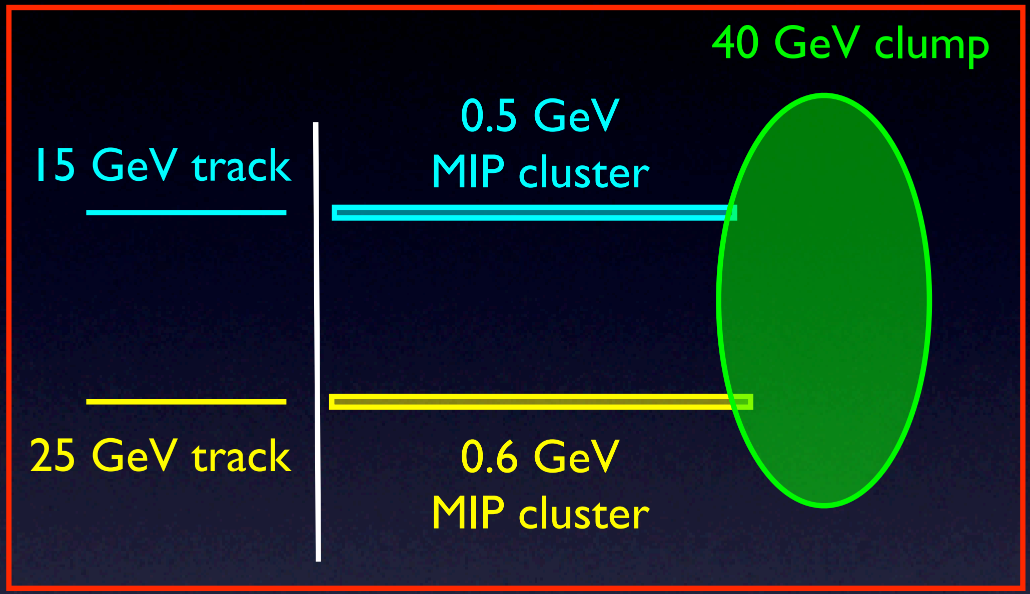
- At the end of each iteration (i.e. after building showers for all tracks), search for this case:
  - Cluster C isn't attached to any shower
  - C is not a photon
  - The best potential link for cluster C was to a track T
  - E/p for track T wouldn't be too high if C was added to its shower
  - ... then add C to the shower of track T
- This helps a LOT
  - In cases where showers are close by / overlapping and the code gets paralyzed with indecision, this helps it break the deadlock
  - ... but sometimes introduces mistakes.

Building charged showers (3)

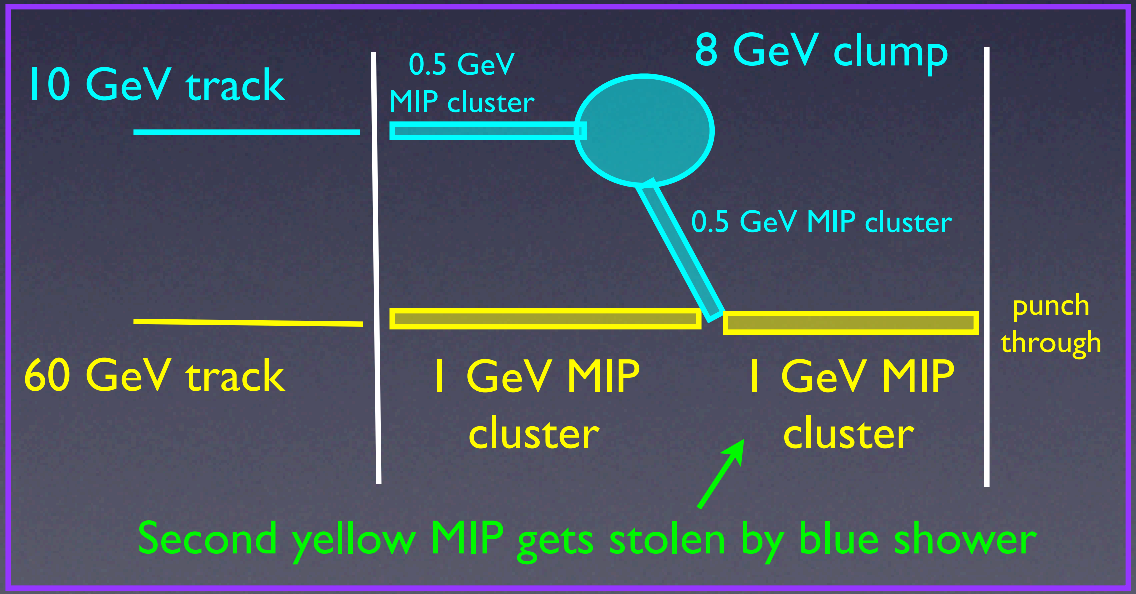
- When done building shower, look at E/p for overall veto
- E >> p is rare (by construction)
- E << p more common -- veto to prevent double-counting (allow $\pm 2.5\sigma$)
- Exception: punch-through tracks (hits in last layers of HCAL) are allowed to under-shoot

Building charged showers (4)

- Sometimes 2+ charged showers are hopelessly entangled...
  ... or one shower steals a critical piece of another
- In that case, treat them as a single "jet" and look at $\Sigma E/\Sigma p$
- Not optimal use of information, but better than failing altogether
- Careful about using punch-through showers in jets -- easy to lose information.



Tracker Calorimeter



Neutral hadron showers

- Any substantial clusters not used in a charged shower are assumed to be from neutral hadron
- Build NH showers same way as charged showers, except with no E/p cut and score threshold fixed

Putting the output together

- Make ReconstructedParticle objects:
  - Photons: 4-vector based on cluster position & calorimeter energy
  - Charged showers passing E/p veto and tracks that don't reach the calorimeter (low-p_T): 4-vector [etc] based on track momentum (assuming pion mass)
  - Charged showers failing E/p veto: 4-vector based on cluster position and calorimeter energy
  - Neutral hadron showers: 4-vector based on cluster position and calorimeter energy

Confusion matrix

- Output from Ron's diagnostic routines shows where all of the energy is going:

ZZ	Truth: photon	Truth: tracked particle	Truth: neutral hadron	Sum
Reco: photon	108,368	5,979	4,247	118,594 Purity: 91.4%
Reco: tracked particle	8,679	227,475	15,539	251,693 Purity: 90.4%
Reco: neutral hadron	6,905	22,673	42,666	72,244 Purity: 59.1%
Unused	1,037	9,177	2,214	12,428
Sum	124,989 Effic: 86.7%	265,304 Effic: 85.7%	64,666 Effic: 66.0%	

Diagonal elements: correct ID

Off-diagonal elements: mis-identified energy

Charged-neutral confusion especially bad...

Next steps

- Improve algorithm itself:
  - Study outlier events which are badly wrong
  - Systematic bias for different classes of event?
  - Scoring: More formal optimization of PFA constants? Make more use of likelihood selector tools? Add variables?
- Improve use of output:
  - Look for π^0, η?
- Look at other designs:
  - So far, only tuned for sid01
  - Quick look at sid01_scint => will need to retune for scint HCAL
  - Work with MIT group to scan

People

- I am moving to a new job in 4Q08 -- won’t be on ILC after.
- Replacement postdoc lined up to join U. Iowa group & overlap with me for few months. (Nominally part-time on ILC, but will be full-time at the start.)
- MIT group working on scan of parameter space
- A lot of the pieces are produced by other people, e.g.
  - DigiSim (Guilherme)
  - Photon-finding & ID (Ron+Qingmin)
  - Calibration (Ron)
  - Track list (currently using Ron’s)
  - DTree clusterer (Guilherme et al)
  - etc

(modular approach has made life so much easier...)