Energy Flow Reconstruction

Norman Graf
(SLAC)

November 8, 2002

Problem Statement

- In order to implement the energy flow algorithm as a part of the event reconstruction, we need a framework within which we can communicate between packages.
- Need common definitions (interfaces) for constituents of final reconstructed particles.
- Need canonical samples on which to develop and test reconstruction algorithms.

ReconstructedParticle I

- A class which encapsulates the behavior of an object which can be used for physics analysis.
  - mirrors MCParticle
- Kinematics determined by track momentum or calorimeter cluster energy at time of creation.
- ID determined later by particle ID algorithms, e.g. track dE/dx, cluster shape, or combination of detector element variables.
  - could entertain multiple hypotheses.

ReconstructedParticle II

- Can also be created from combinations of other ReconstructedParticles.
- e.g. Photon can be single EM cluster without associated track, or combination of e^+ and e^-, each composed of an EM cluster and a matching track.
- Resonances, when identifiable.
- Jets could also be ReconstructedParticles.

ParticleType

- Encapsulates information about known types of particles, e.g.
  - name
  - mass
  - charge
  - pdg code
- Not limited to SM particles, could also simply consider “EFlow” particles, e.g. “Neutral EM”, “Neutral Hadron”, “Charged Hadron”, etc.

ParticleId

- Combines a ParticleType and the probability for the id given by a ParticleTypeIdentifier.
- Also contains a GUID to allow the identification to be reviewed at a later time.
- ReconstructedParticle should contain all the information needed by a ParticleTypeIdentifier to return a ParticleId.

ReconstructedParticle attributes

- Collection of calorimeter Cells and/or Clusters
- Collection of Tracks
- Collection of ParticleIds (sorted by probability)
- Mass
- Charge
- Kinematics
- Collection of ReconstructedParticles of which this is composed
- ReconstructedParticle of which this is a constituent
- Flag to indicate whether this is a final state

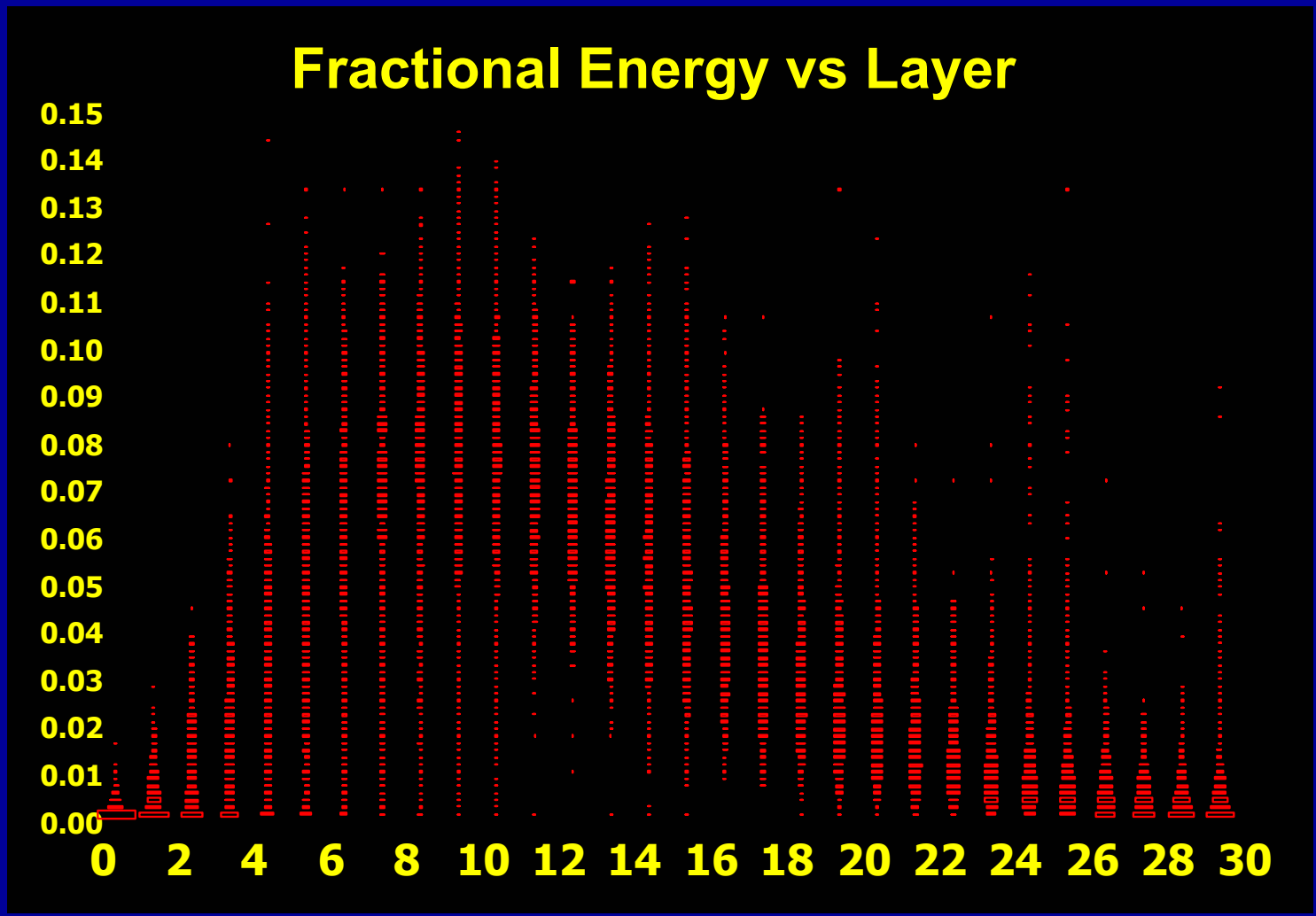
Photon Id

- Developed simple cone algorithm for finding EM clusters.
  - Fast, efficient.
- Implemented fully covariant χ^2 calculation for longitudinal shower shape analysis.
- Use shower width for γ-π^0 discrimination.
- In addition, use track-match and E/p for electron id.

Cone Algorithm

- Currently using fixed cone radius of 0.03
  - Based on energy contained within cone.
  - Based on number of clusters.
- Could also use a cone radius based on energy of seed cell.
- Currently split clusters whose cones overlap by associating cells to nearest cone axis.
- Could also search for NN clusters within cone.

Longitudinal Energy Distribution



Longitudinal HMatrix

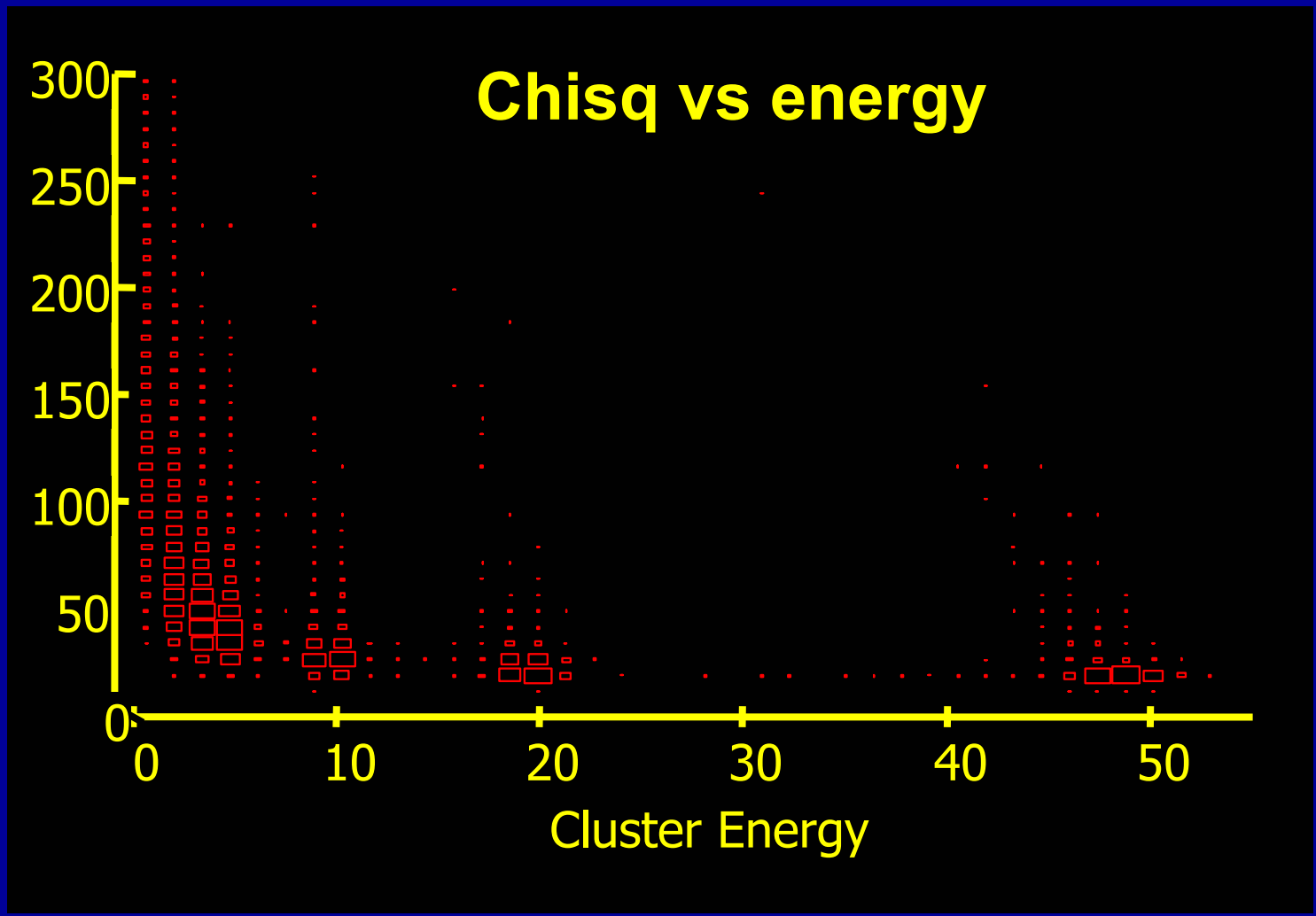
- Use longitudinal energy depositions and their correlations to create a cluster χ^2.

$$\mathbf{M_{ij}} = \frac{1}{\mathbf{N}} \sum_{\mathbf{n=1}}^{\mathbf{N}} (\mathbf{E_i^{(n)}} - \mathbf{\overline{E}_i})(\mathbf{E_j^{(n)}} - \mathbf{\overline{E}_j})$$

$$\mathbf{H} \equiv \mathbf{M^{-1}}$$

$$\zeta_{\mathbf{m}} \equiv \sum_{\mathbf{i,j=1}}^{\mathbf{N}} (\mathbf{E_i^{(m)}} - \mathbf{\overline{E}_i})\mathbf{H_{ij}}(\mathbf{E_j^{(m)}} - \mathbf{\overline{E}_j})$$

Cluster χ^2



Muon Id

- Using Rich Markeloff's package to identify muons.
- Works well in central (barrel) region for high pT muons.
- Needs to be extended into endcaps
- Needs to be augmented to find muons which range out before making it to the muon system.
- Need to characterize efficiencies, fakes, etc.

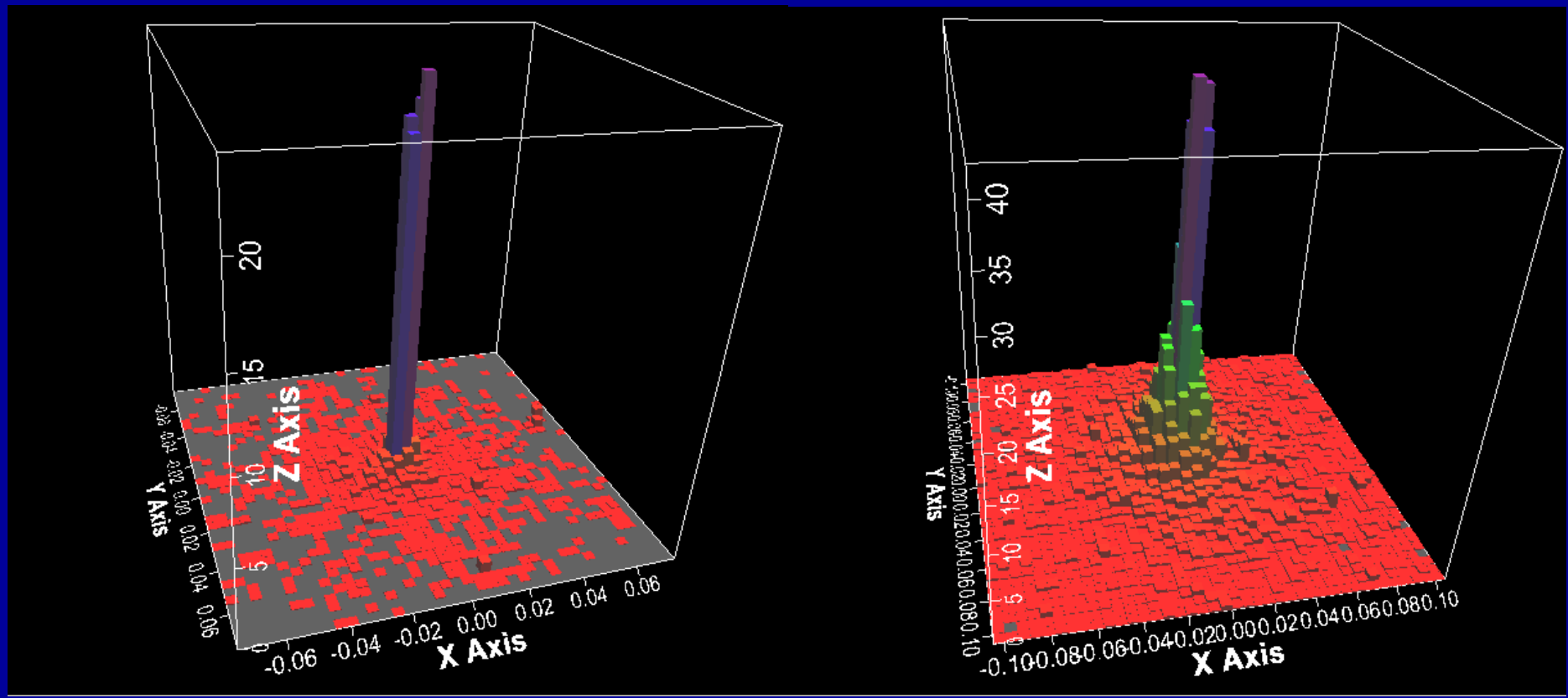
Charged Hadron Id

- Continuing to characterize pion shower shapes in calorimeters as function of momentum and direction.
- PionShower class being developed to encapsulate the association of hit calorimeter cells with extrapolated tracks.
  - Follows MIP trace to shower start.
  - Characterize hit-track association with χ^2.
  - Will allow association to proceed until a limit is reached on either match χ^2 or E/p.

Hadronic Shower Shapes

EM Layer 1

HAD Layer 13



Prototype Reconstruction

```
public ReconstructedParticleJob(double radius, double
    seedEmin, double clusEmin, String hmxName, double
    clusEmin, double chisqmin, double trackdistmin)
    {
        // Smear Tracker hits with resolution
        add(new SmearDriver());
        // Find tracks
        add(new TrackReco());
        // build up the eflow event
        // sets up and populates the CalorimeterHitMap
        add(new EflowEventBuilder());
```

Prototype Reconstruction

```
    // Find muons
    add(new MuonFinder());
    // Find EM clusters using a simple cone algorithm
    add(new EMConeClusterBuilder(radius, seedEmin,
clusEmin));
    // Construct and identify the ReconstructedParticles
    // Photons, electrons, pi0
    add(new
EMParticleFinder(hmxName,clusEmin,chisqmin,trackdistmin));
    // charged hadrons
    add(new ChargedParticleFinder());
    // neutral hadrons
    add(new NeutralHadronFinder());
    // Physics!
    add(new EventAnalyzer());
}
```

To do

- Establish use cases for ReconstructedParticles
  - Try out a few example analyses, improve.
- Clean interface at this level allows much closer collaboration.
- Where do we draw the line between reconstruction and analysis?
  - We currently think of "EFlow Objects" as end deliverable, and let user do jet-finding, etc.
- Iterate!