Reconstruction of neutral particles in the GlueX detector

Feb. 27, 2009

Mihajlo Kornicer

Indiana University

Overview

- Data flow
- Photon reconstruction
- Two-photon factories: π^o and η

Data Flow

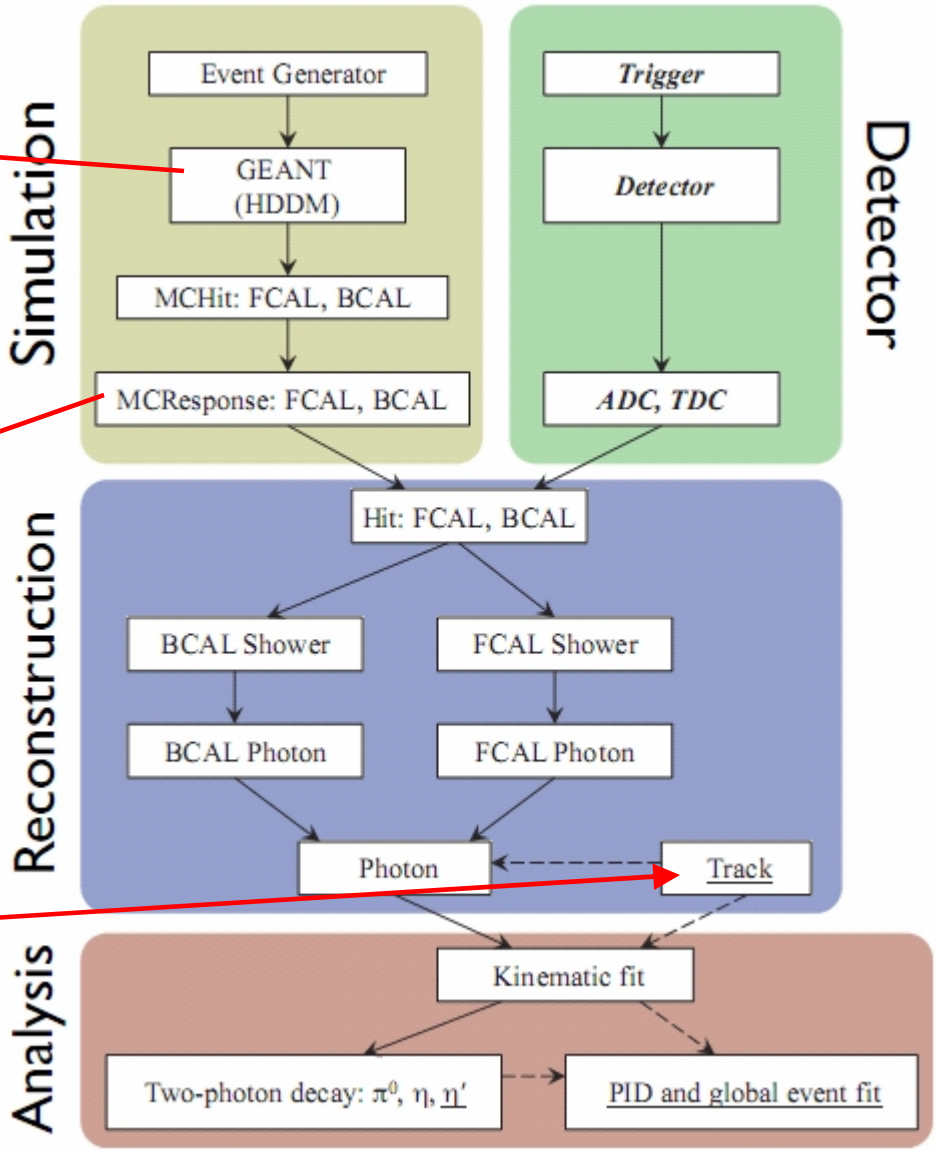
Simplified Monte Carlo:

- signal ~ ***dE/dx***
- FCAL: **no Cerenkov photons**
- BCAL: **no fibers** or **scintillation photons**

More sophisticated detector-response done here

Showers from charged particles:

- identification based on MC info **at this stage**



Photon reconstruction

- **FCAL MCResponse:** photon-counting statistics
- **BCAL MCResponse:** sampling-fluctuations, time delay and signal attenuation
- **FCAL Clusters:** code ported from RADPHI[1]
- **BCAL Showers:** code ported from KLOE[2]
- **FCAL Photons:** non-linear energy and shower depth corrections[1]
- **BCAL Photons:** non-linear z-dependent energy correction[2]
- **PID Photons – user object:** a list of FCAL/BCAL photons, inherits from *DKinematicData*, covariant error-matrix calculated

Documentation:

1 - GlueX-docs 823, 1093

2 - GlueX-docs 569, 849

3 - GlueX-doc 989 – Calorimeter Simulation

Kinematic Data

```
#  DKinematicData.h
#  Hall D
#
#  Created by Matthew Shepherd on 5/28/07.
#
# Class largely borrowed from the KTKinematicData class used in CLEO.  Where applicable CLHEP objects have been
# converted into Hall D ROOT-based typdefs.  A portion of the original CLEO documentation appears below:

# Describes kinematic properties of charged tracks, photons, and virtual particles such as pi0, Ks, so that a user can carry
# out standard operations such as calculating masses, adding 4-momenta together, etc. The basic information consists of
# the 3-momentum, 3-position, mass and charge. 7x7 error matrix is stored for the quantities (Px,Py,Pz,E,x,y,z). …
```

... the rest of it can be found in …/src/libraries/PID/DKinematicData.h

Example:

```
vector<const DPhoton*> photons;
eventLoop->Get(photons);
for (unsigned int j = 0; j < photons.size() ; j++) {
    unsigned int T =  photons[j]->getTag()
    TLorentzVector p4 = photons[j]->lorentzMomentum()
    DVector3 vertex = photons[j]->position()
}
```

Photon tags:

1 = FCAL

2 = BCAL

3 = charged particle*

* **based on Δθ from closest charged MC thrown object**

Two-gamma Fit Factory

TwoGammaFit_factory_PI0(π^0) {
  FitTwoGammas(π^0)
}

TwoGammaFit_factory(mass) {
  FitTwoGammas(mass)
}

Constrains all photon pairs to given mass and provides:
- χ^2
- confidence level
- pulls
- IDs to look-up children momenta …

Getting π^0 and η:

```
vector<const DTwoGammaFit*> fit2g;
loop->Get( fit2g ,"PI0");
```
… or
```
loop->Get( fit2g ,“ETA");
```

Useful functions:

- lorentzMomentum() ($\gamma_1+\gamma_2$ before fit)*
- getProb()
- getChi2()
- getPull(i), i=0,5 (p_{x1}, … , p_{z2})
- getChildID(i), i=0.1 (use to look-up γ)
- getChildTag(i), i=0,1
- getChilMom(i), i=0,1 (after fit)*

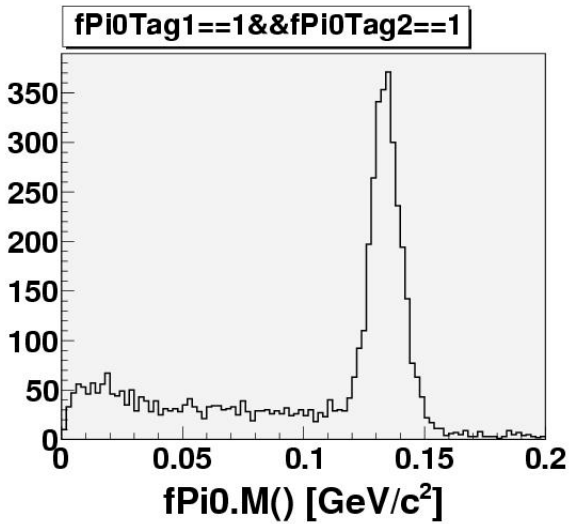
*** for now, will fix it later**

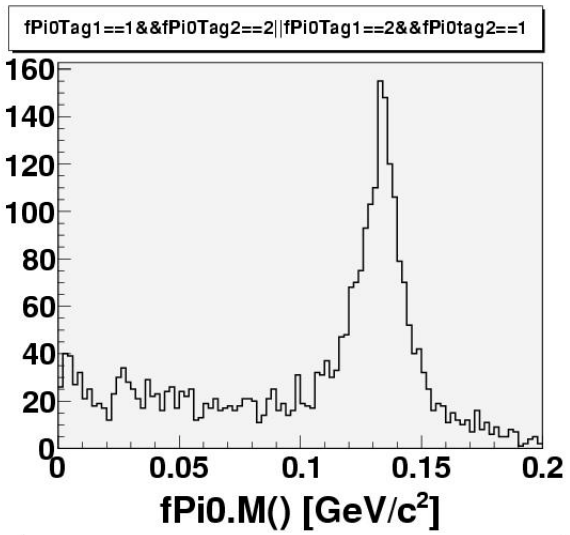
State of the art: π^0 and η from particle gun

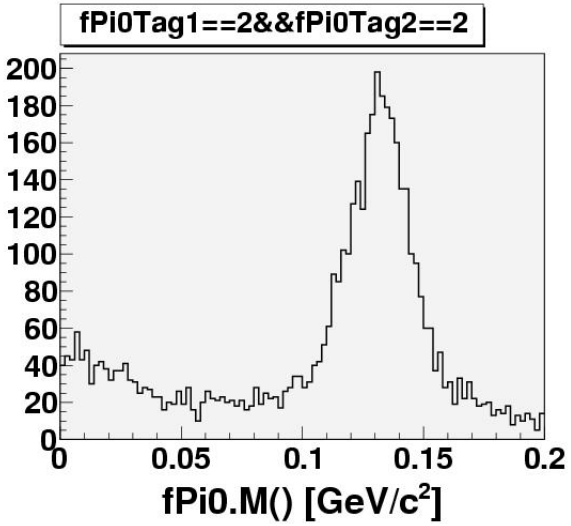
$0 < P < 8$ GeV , $0 < \theta < 40^0$

FCAL | **FCAL / BCAL** | **BCAL**

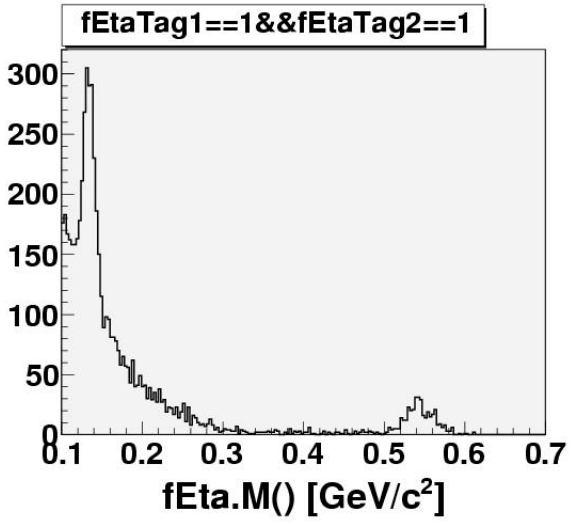
π^0

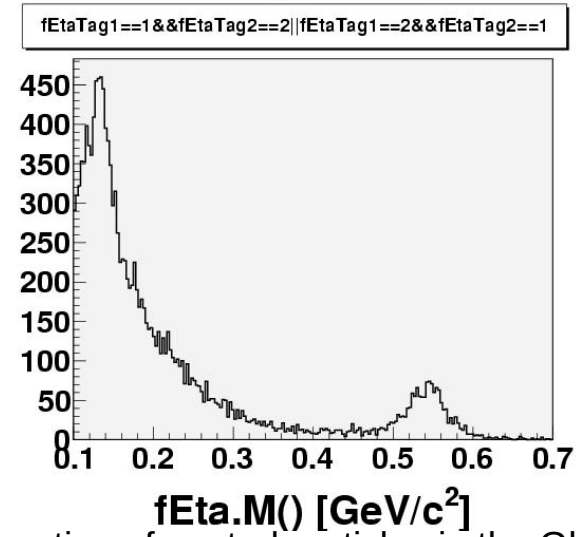


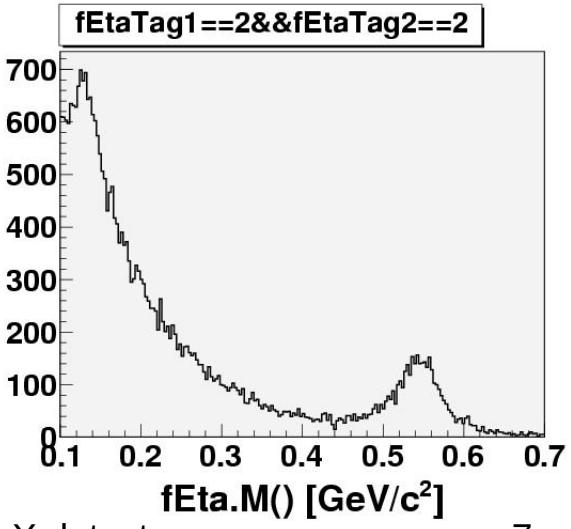




η







Happy Neutral Reconstruction!