KOTO Data and Analysis Run 49

Chris Rymph

10-28-2015

Introduction

Analysis in the KOTO experiment revolves primarily around two C libraries written by members of the KOTO collaboration: the “E14 Library” and “Analysis Library”. The E14 Library, based on the software library developed for the E391A experiment, provides the basic framework for analysis, containing class descriptions for various particle objects, analysis algorithms, and the code necessary to tie it all together. The E14 Library also contains the Geant4 based experiment simulation code. The Analysis Library contains additional code such as detector specific geometric considerations and calibration information as well as basic event reconstruction programs.

This document will introduce the usage of these libraries for basic data analysis and event reconstruction, as well as an overview of the data itself. To perform data analysis, some experience with C and ROOT is necessary. Though some basic root examples are included, these topics are beyond the scope of this document. For a more thorough introduction to ROOT, see:

https://root.cern.ch/root/html534/guides/primer/ROOTPrimer.html

https://root.cern.ch/root/htmldoc/guides/users-guide/ROOTUsersGuide.html

E14 Library Setup (KEKCC)

The following software and libraries are required to compile the e14 library: ROOT 5.32.00, CLHEP 2.1.1.0, Geant4 9.5.p02. They are already available on KEKCC, and are set up by the script env.sh within the KEKCC e14 library.

Available versions of the e14 library are located on KEKCC in */sw/koto/e14lib*

For the most recent and run-appropriate version of Analysis Library, I recommend asking someone in the analysis group.

The following required environment variables should be added to your ~/Env/bashrc_post file. The example shown uses the library versions I used during my own analysis on run 49 pro4 data. You should make sure you are using an appropriate version of e14lib and AnalysisLibrary if your own analysis is on data other than run 49.

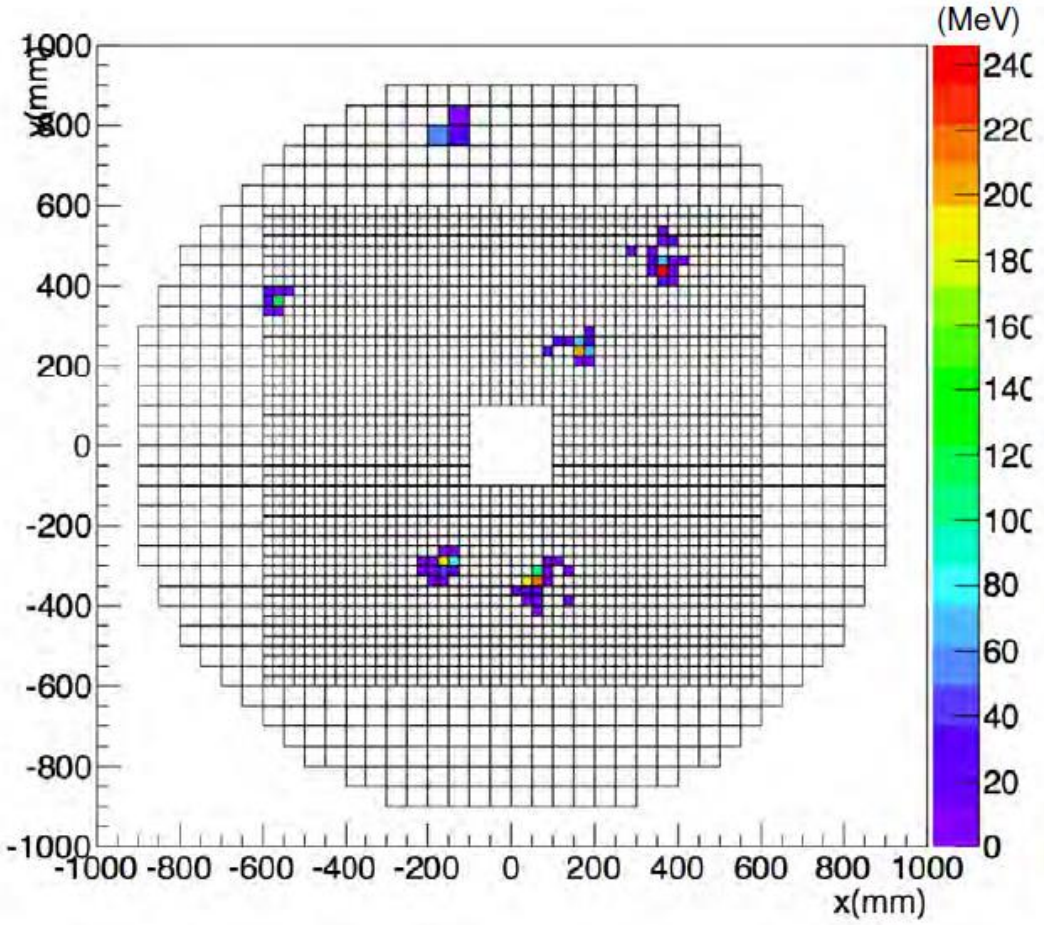
```
# e14lib and AnalysisLibrary related environment variables
source /sw/koto/e14lib/201305v2.4/env.sh >/dev/null
export E14_ANALYSIS_LIBRARY_PATH=/home/had/crymph/SCIENCE/analysis/AnalysisLib/AnalysisLibrary
export MY_TOP_DIR=/home/had/crymph/SCIENCE/analysis/AnalysisLib
export LD_LIBRARY_PATH=$MY_TOP_DIR/lib:$LD_LIBRARY_PATH
```

Analysis Overview

- *Experiment:* Detector energy and timing information is saved in a timing window around a trigger
- *Clustering:* Convert CsI energy and timing info into hit clusters
- *Gamma Finding:* Convert clusters to photon hits
- *Reconstruction:* Reconstruction of orignal particle decay position, energy, and momentum
- *Cuts:* Discard background events and incorrectly reconstructed particles
- *Analysis:* Make lots of plots *(Label axes)*

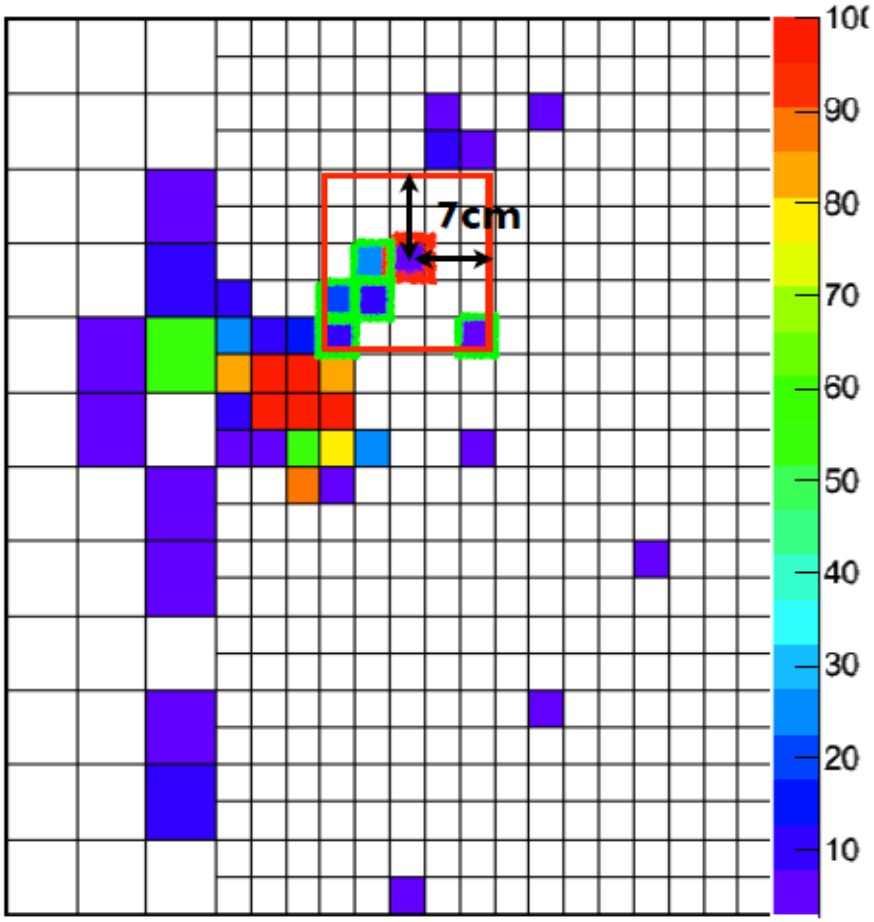
Experiment

- An event consisting of 64 samples (512 ns) of data from each detector is recorded when a trigger is generated.
- Physics events are triggered by a minimum energy level in the CsI detector.



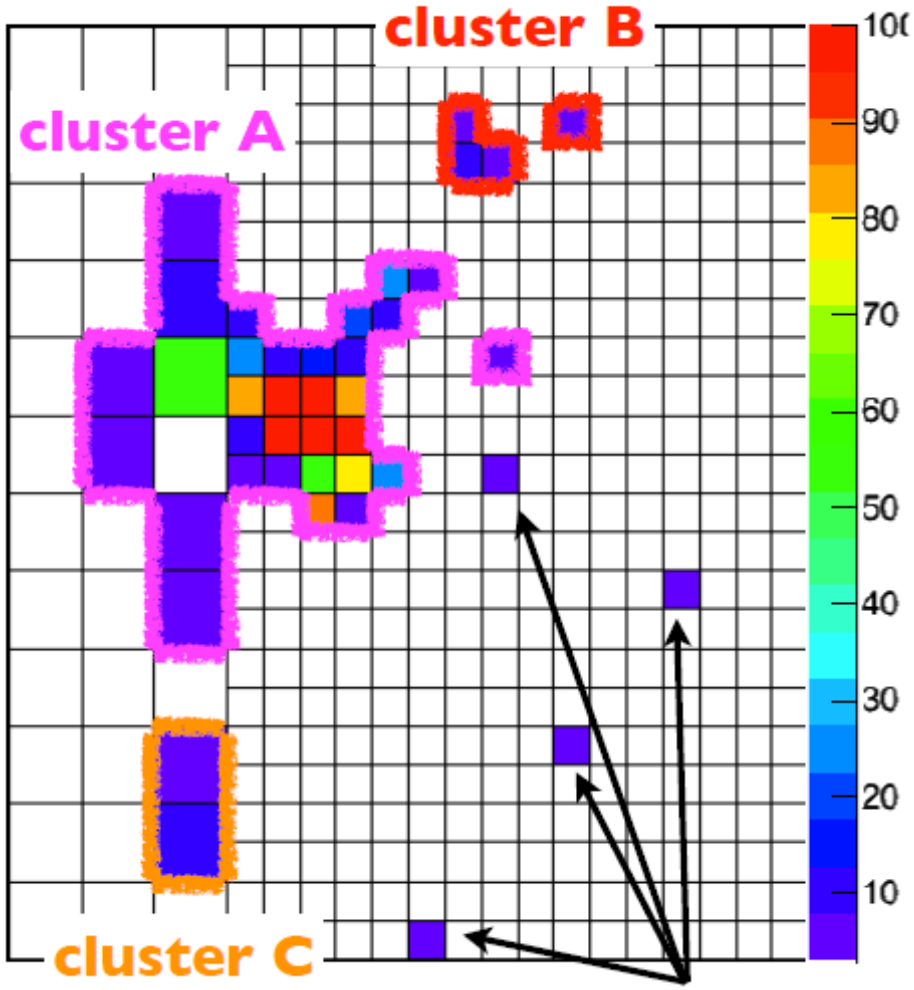
Energy distribution in the CsI detector for one event

Clustering



- Energy in one crystal is selected at random as a cluster seed.
- Simultaneous energy deposits within a 7cm radius of the seed are added to the cluster.
- Crystals within 7cm of these are also added to the cluster.

Clustering



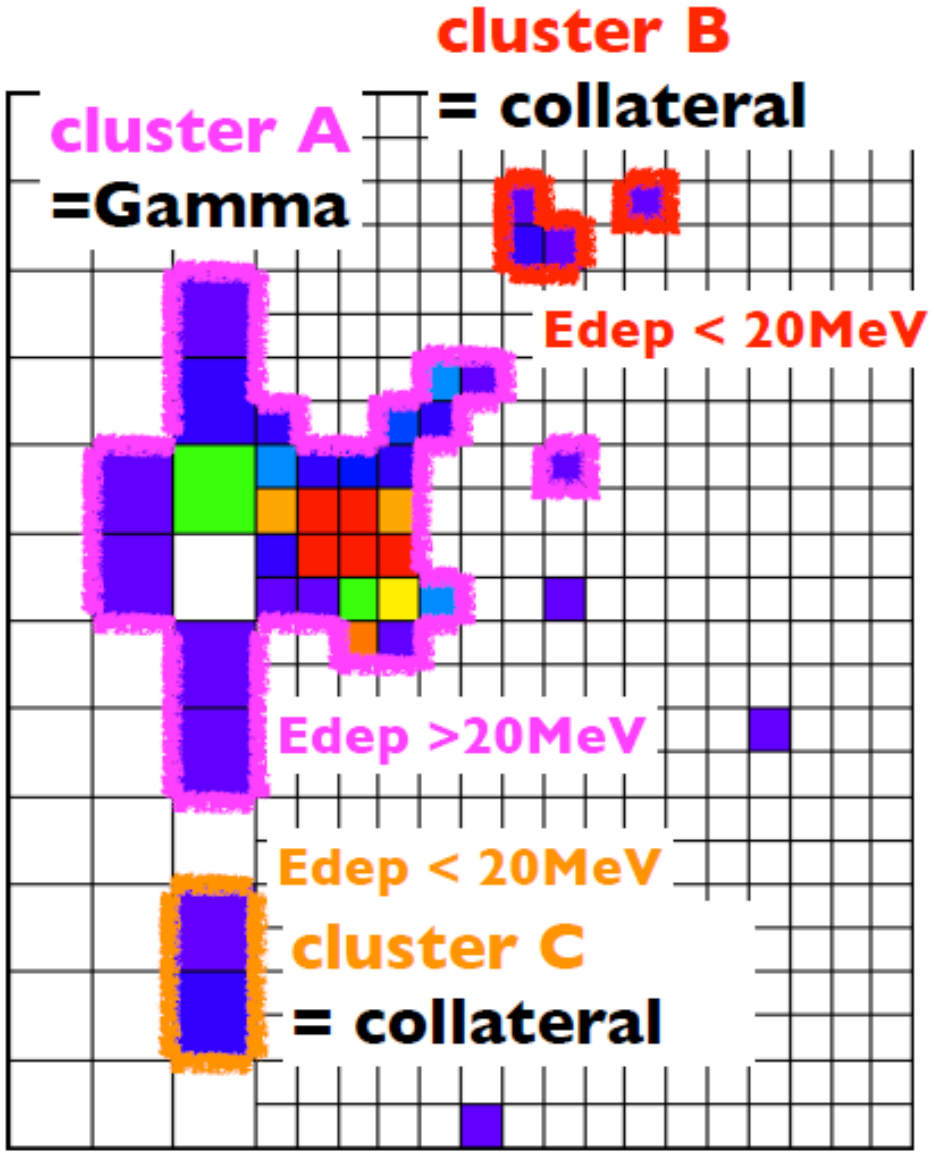
- Process is repeated until all clusters are found.
- Cluster energy, timing, and center of energy position are saved.

$$x_{COE} = \frac{\sum_i^n [E_i \cdot x_i]}{\sum_i^n E_i}$$

Isolated crystals are not saved.

Gamma Finding

- Clusters satisfying an energy threshold are considered photon hit candidates.
- Energy corrections are applied to compensate for leakage.
- Shape of energy distribution is used to distinguish valid photons.



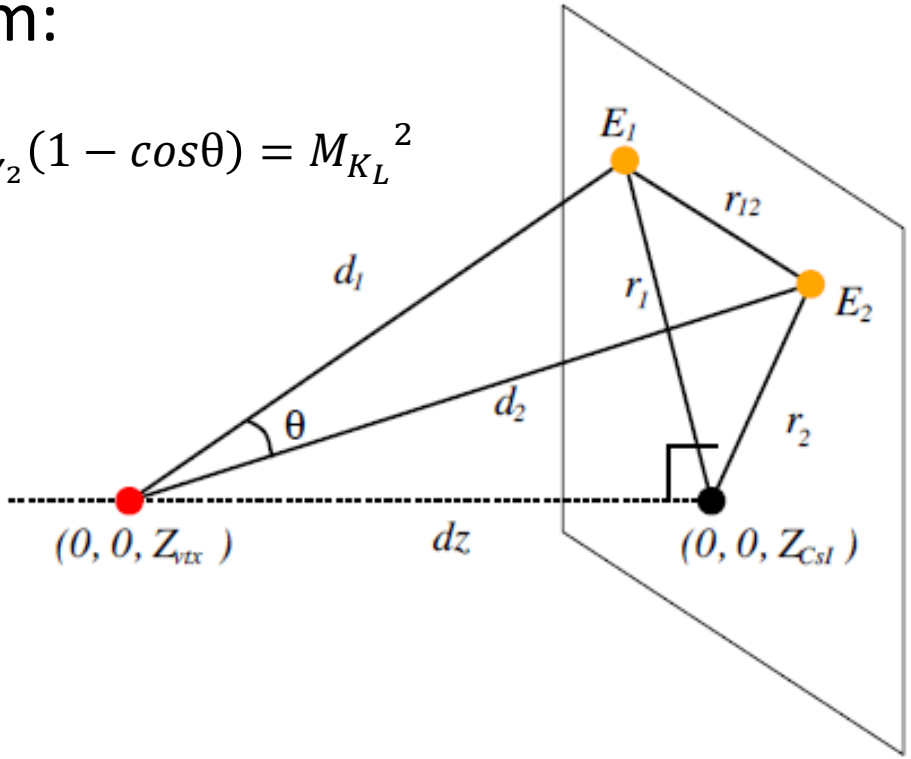
Reconstruction: K_L -> γγ

- Select events with two simultaneous photon hits.
- Assume K_L mass and decay on the beam axis (x=y=0)
- From Lorentz invariance, conservation of energy, and conservation of momentum:

$$P^{\alpha}{}_{K_L} P_{\alpha\, K_L} = \left(P^{\alpha}{}_{\gamma_1} + P^{\alpha}{}_{\gamma_2}\right)^2 = 2E_{\gamma_1}E_{\gamma_2}(1 - cos\theta) = {M_{K_L}}^2$$

$$\cos\theta = 1 - \frac{{M_{K_L}}^2}{2E_{\gamma_1}E_{\gamma_2}}$$

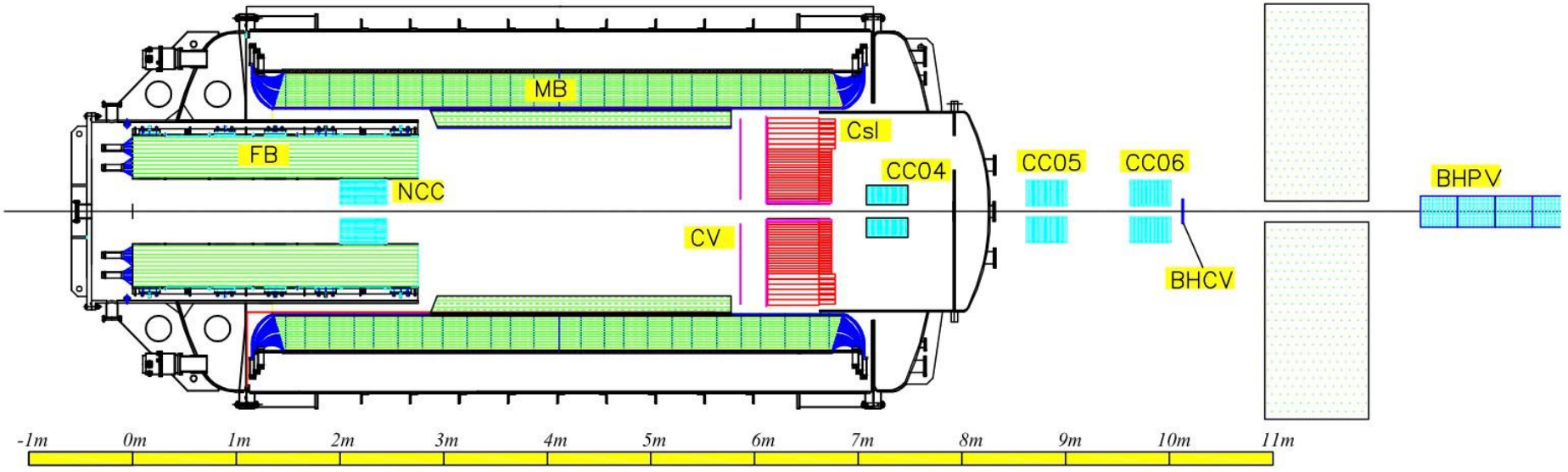
- Decay vertex is then calculated from the geometry shown here.



$${r_{12}}^2 = {r_1}^2 + {r_2}^2 + 2(Z_{CSI} - Z_{vtx})^2 - 2\sqrt{{r_1}^2 + (Z_{CSI} - Z_{vtx})^2}\sqrt{{r_2}^2 + (Z_{CSI} - Z_{vtx})^2}cos\theta$$

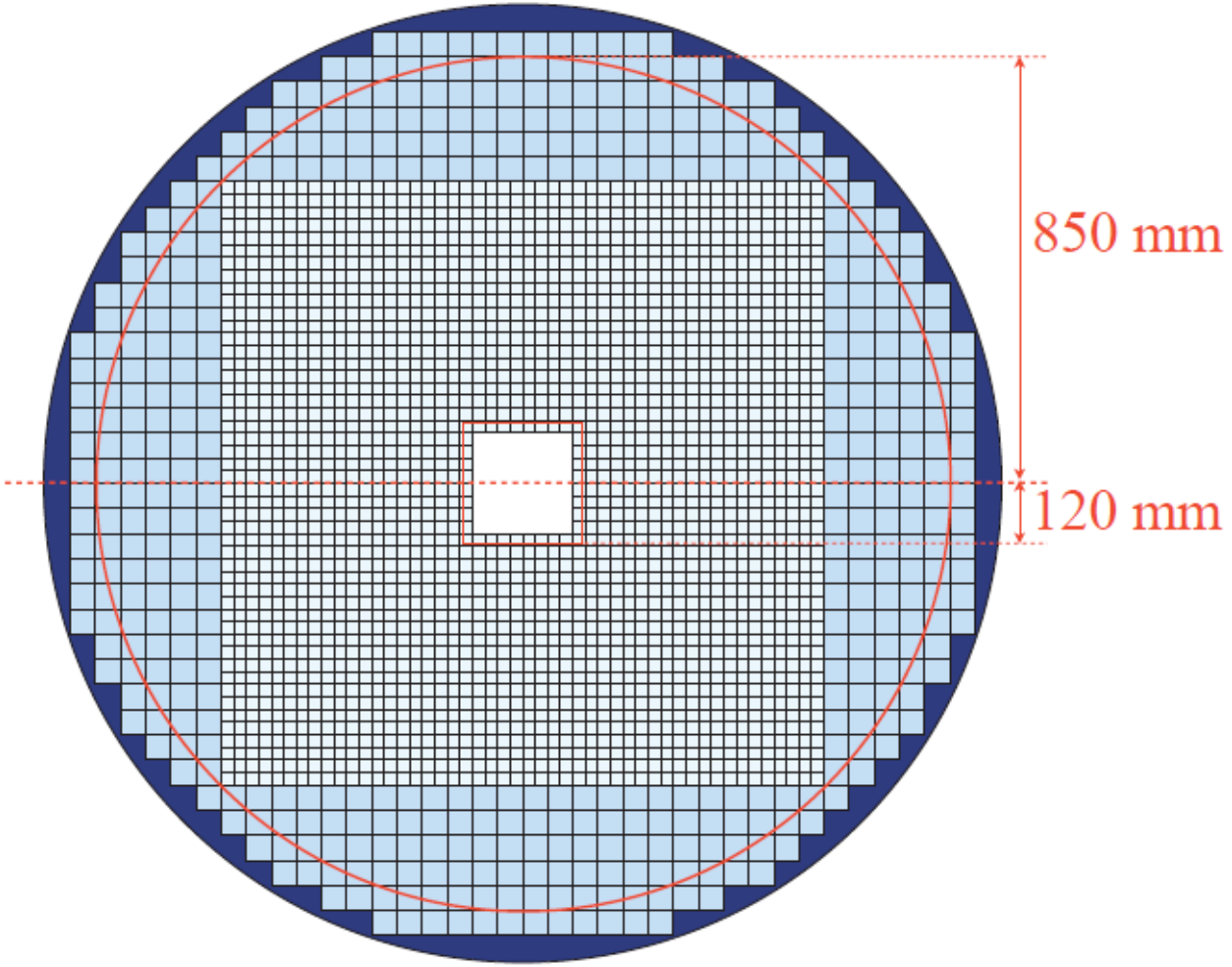
Cuts – Veto Detectors

- Energy deposits inside a timing window around K_L photon hits are summed for each veto detector.
- Events over detector thresholds are discarded to remove background decay modes with more than 2 photons in which some did not hit the CsI detector.



Cuts - Fiducial

- Due to energy leakage in outer crystals, hits must be outside a 240mm square and within a radius of 850mm.



Data Overview

- *Binary:* Raw binary data stored by Level 3.
  (See FileFormat6DataStructure.pdf for details)
- *Converted:* Same information as raw binary data; converted to ROOT format for easier access.
- *Data Summary Tape (DST):* Typical starting point of analysis; contains calibrated detector energy and timing information.
- *Gsim:* Simulated data generated by Geant4 Monte Carlo.
- *Clustering:* Contains energy and timing of groups of CsI energy deposits along with original detector energy and timing data.
- *Reconstructed:* Reconstructed kaon decay events containing photon, pion, and kaon kinematic information.

Converted Data

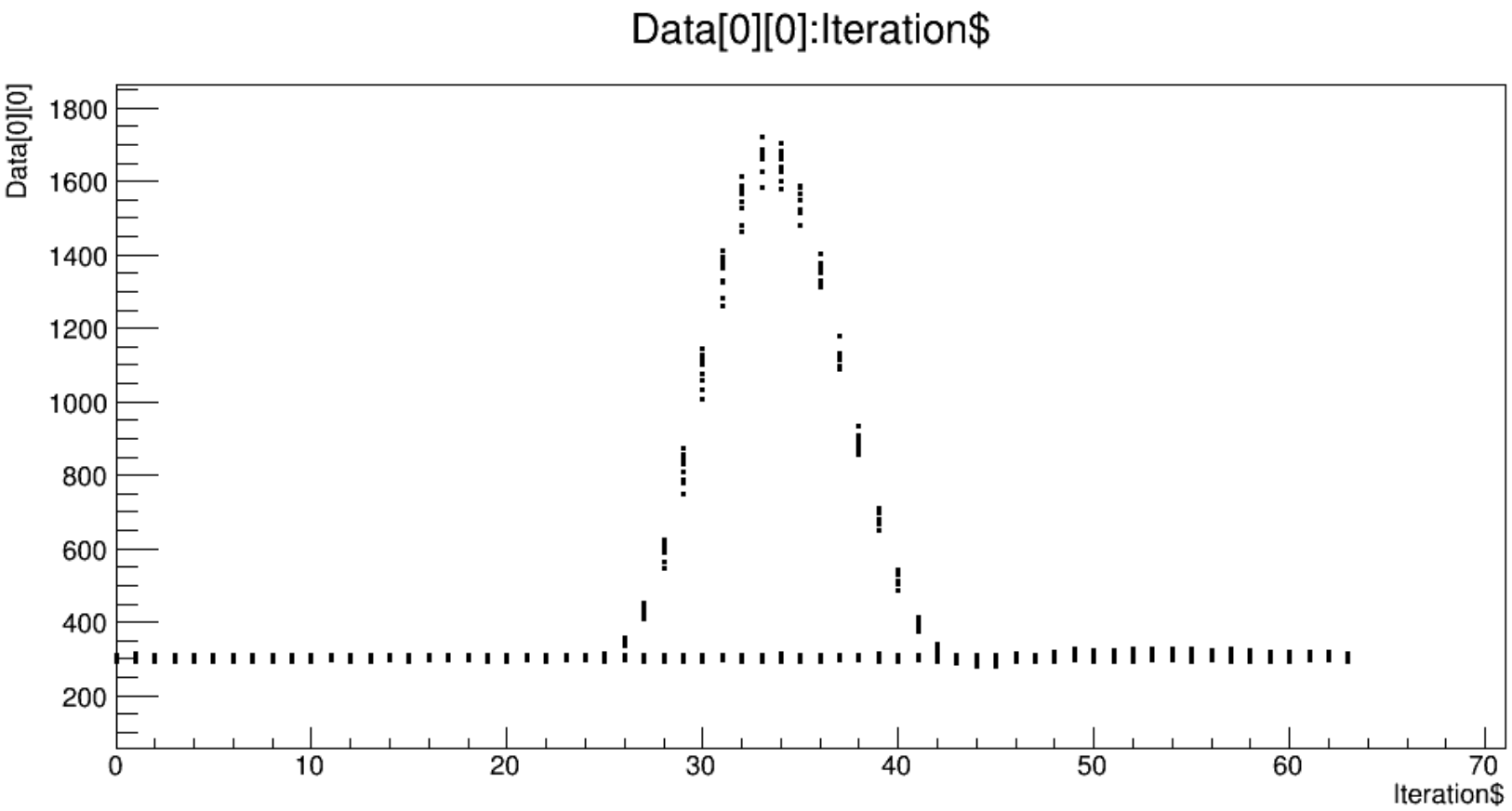
Examples:
/gpfs/fs03/had/koto/ps3/klea/work/togawa/production/run49/pro1/work/fiber_out/conv_data

ROOT file containing event header information: spill number, trigger information, waveform data, etc. Data is separated into trees by crate; Crate0, Crate1, …, Crate15.

Some key features:

- Data branch contains original 64 sample ADC waveforms, in units of ADC counts, in a 20x16x64 array of short integers: Data[Module][Channel][Sample]
- IntegratedADC branch contains the pedestal-subtracted integrated waveform (which is proportional to total deposited energy) in units of ADC counts in a 20x16 array of doubles IntegratedADC[Module][Channel].
- COE_Ex, COE_Ey, COE_Esum branches contain the values used in online COE calculation.
- ScaledTrigBit contains the type of trigger generating each event. ScaledTrigBit is described in more detail later in this document.

Converted Data



Example: Viewing all 64 samples across all events in one file for Crate 0, Slot 0, Channel 0 in a converted data file.

Crate0->Draw("Data[0][0]:Iteration$")

DST

Examples:
/gpfs/fs03/had/koto/ps3/klea/work/togawa/production/run49/pro4/work/fiber_out/dst_data

Contains calibrated energy and timing information for each detector and channel. Although header information (trigger tag) for each event is copied from original converted data, waveform shape information is lost. New versions of this data are produced as calibration is completed or improved. Final version for run49 is pro4.

Some key features:

- For each detector (CSI, CV, etc.), DETECTOREne branch contains energy deposited in each channel in an array of floats: DETECTOREne[DETECTORNumber]
- DETECTORTime branch contains the peak time of energy deposit in an array of floats: DETECTORTime[CSINumber]
- “DETECTORNumber” relates to specific detector modules as described in the Analysis Library channel map files: AnalysisLib/data/mapFiles/ch_map_(DETECTOR).txt

See also:
http://koto.quark.kj.yamagata-u.ac.jp/wiki/index.php?plugin=attach&refer=Beam%20time%2FMay.2013&openfile=AnalysisInformation_run49_v1.pdf

Gsim

Example simulation macros: $E14_TOP_DIR/examples/gsim4test/

More recent simulation macros and data by Hajime Nanjo:
/gpfs/fs03/had/koto/ps3/klea/work/nanjo/MC

Simulated events generated by the e14 library using Geant4. Kaons are generated one at a time at the detector entrance with a given momentum spectrum and then proceed to decay by a given decay mode. Particle interactions within detector materials are simulated and energy deposits are recorded.

Convert gsim data to DST format before reconstruction/analysis using gsim2dst from the Analysis Library:

gsim2dst (input gsim file) (output dst file) (accidental seed)

“Accidental seed” is a random seed used to apply an accidental event overlay to the simulated data. Accidental energy deposits recorded during a special run are randomly time shifted and added to simulated events. No argument or -1 will disable accidental overlay.

Gsim

Generating Simulated Data

Create a working directory:

mkdir KLpi0pi0

cd KLpi0pi0

Make/copy a simulation macro:

cp /gpfs/fs03/had/koto/ps3/klea/work/nanjo/MC/201305v2.14/KLpi0pi0/scripts/e14_201305_KLpi0pi0.mac .

Execute gsim4test with the following arguments:

gsim4test (macrofile) (outputfile) (number of events) (random seed) (run number)

For example:

$E14_TOP_DIR/bin/gsim4test e14_201305_KLpi0pi0.mac KLpi0pi0-gsim0.root 10000 42 0

Convert gsim data to DST format:

$E14_ANALYSIS_LIBRARY_PATH/gsim2dst/bin/gsim2dst KLpi0pi0-gsim0.root KLpi0pi0-simdst0.root 0

Clustering

Generated by running clusteringFromDst on DST data. Contains energy, position, and timing of CsI clusters, along with previous DST information.

Some key features:

- ClusterNumber branch contains the number of clusters in an event as an integer.
- ClusterDepE branch contains the total energy of each cluster in an event, in units of MeV, in an array of doubles: ClusterDepE[ClusterNumber]
- ClusterTime branch contains the energy-weighted cluster time of each cluster in an event, in units of ns, in an array of doubles: ClusterTime[ClusterNumber]
- ClusterCoePos branch contains the CoE position of each cluster in an event, in units of mm, in an array of doubles: ClusterCoePos[ClusterNumber][x=0,y=1,z=2]

Clustering

Generating Clustering Data

Run clusteringFromDst from the Analysis Library on DST format data as follows:

clusteringFromDst (inputfile) (outputfile) (run number) (node number) (file number)

With node number referring to the corresponding Mandolin node for run data or an arbitrary number for simulation data.

Example (pro4 run data):

$E14_ANALYSIS_LIBRARY_PATH/clusteringFromDst/bin/clusteringFromDst /gpfs/fs03/had/koto/ps3/klea/work/togawa/production/run49/pro4/work/fiber_out/dst_data/node004/run00016931_node004_file2.root ~/run00016931_node004_file2_clustering.root 16931 4 2

Example (simulation, continued from gsim example):

$E14_ANALYSIS_LIBRARY_PATH/clusteringFromDst/bin/clusteringFromDst KLpi0pi0-simdst0.root KLpi0pi0-simclus0.root 0 999 0

Reconstructed

Generated by running an event reconstruction program on DST data. Contains photon, pion, and kaon energy, timing, positions, etc. Basic reconstruction programs for several decay modes are available in the Analysis Library:

g2anaMay2013; events with 2 photons; one (K_L->$\pi^0\nu\nu$) or zero (K_L->$\gamma\gamma$) pions.

g4anaMay2013; events with 4 photons; 2 pions

g6anaMay2013; events with 6 photons; 3 pions

Some key features:

- KlongNumber branch contains the number of reconstructed kaons in an event. In events with more than 2 photons, multiple combinations may be used to reconstruct kaons. Klong information is sorted by a "z vertex chisquare," so KlongNumber = 0 is in general "the best" and used for analysis.
- KlongMom branch contains reconstructed kaon momentum, in units MeV/c, in an array of doubles: KlongMom[KlongNumber][x=0,y=1,z=2]
- KlongPos branch contains the reconstructed decay vertex position for each kaon, in units of mm, in an array of doubles: KlongPos[KlongNumber][x=0,y=1,z=2]

Photon/Pion information is stored in equivalent Gamma/Pi0 branches.

Reconstructed

Trigger Bit

ScaledTrigBit branch contains the type of trigger which originally triggered each event, as a binary word integer. Trigger types are:

- (0x0001) : Physics
- (0x0002) : Normalization
- (0x0004) : Minimum bias
- (0x0008) : Calibration
- (0x0010) : Off spill (NCC)
- (0x0020) : Off spill (OEV)
- (0x0100) : Spill Gate
- (0x0200) : External 1 (clock, laser, LED)
- (0x0400) : External 2 (TMon)

The interesting trigger types in physics run analysis are Physics, Normalization and Minimum Bias. Physics triggers have a minimum CsI energy threshold, maximum CoE radius, and maximum energy threshold in CV, CBAR, CC03, and NCC. Normalization triggers have minimum CsI and CV, CBAR, CC03, NCC threshholds and are recorded with a prescale of 1/30 events in run49. Minimum bias events have the minimum CsI energy requirement only with a prescale of 1/300 in run49.

See Jia Xu's PhD thesis for more information on trigger conditions.

Reconstructed

Generating Reconstructed Data

Run g(n)anaMay2013 from the Analysis Library on clustering output data as follows:

g(n)anaMay2013 (inputfile) (outputfile)

Example (simulation, continued from gsim example):

$E14_ANALYSIS_LIBRARY_PATH/g4anaMay2013/bin/g4anaMay2013 KLpi0pi0-simclus0.root KLpi0pi0-simrec0.root

Example: Analysis ROOT Macro

Continued from simulation example. Go to KLpi0pi0 directory. Execute the following commands or use the root macro "KLpi0pi0_example.c".

```
// Recommend setting root to batch mode to disable graphical interface. Transmitting the GUI over the internet from KEKCC is extremely slow.
gROOT->SetBatch(1);

// Create a TChain to access the tree "RecTree“ in the reconstructed data file.
TChain *KLpi0pi0 = new TChain("RecTree");
KLpi0pi0->Add("KLpi0pi0-simrec0.root");

// Create a canvas to contain a histogram and a histogram fro the longitudinal momentum spectrum.
TCanvas *PzCanvas = new TCanvas("Pz");
TH1D *hPz = new TH1D("hPz","hPz",100,500,6000);

// Draw Z component of the momentum branch into the created histogram, applying the default veto and kinematic cuts.
KLpi0pi0->Draw(“KlongMom[0][2]>>hPz","MyVetoCondition==0&&MyCutCondition==0");

//Set title, label axes
hPz->SetTitle("Reconstructed K^{0}_{L} P_{Z}");
hPz->GetXaxis()->SetTitle("P_{Z} (MeV/c)");

// Redraw updated histogram with error bars and logarithmic y axis, making sure correct canvas is active.
PzCanvas->cd();
hPz->Draw("e");
gPad->SetLogy();

// Save canvas as a png file to avoid KEKCC graphical interface lag.
PzCanvas->SaveAs("hPz.png");

// Histogram will be quite boring as most events will miss CsI or be rejected. Try this process with more sample data.
```

Analyzing Large Data Sets: LSF

Processing the large datasets generated by the experiment (hundreds of TB) takes a lot of computing time. To process many files at once, use the KEKCC LSF Job system. Up to 1500 jobs may be submitted at a time, and they will run simultaneously on a separate computer cluster as resources are available. Text output of executed jobs will be saved in ~/.lsf upon job completion.

Submit a job to an appropriate queue. The "s" queue for jobs under 6 hours is usually a good choice.

Bsub -q (queue) (command)

For example, running the previous simulation as a job:

bsub -q s $E14_TOP_DIR/bin/gsim4test e14_201305_KLpi0pi0.mac KLpi0pi0-gsim0.root 10000 42 0

Monitoring Jobs:

bjobs

busers –w

Cancel a job:

bkill (job number)

Cancel all of your jobs:

bkill 0

For more information, see:

http://kekcc.kek.jp/service/kekcc/html/Eng/JobExecution.html