Cluster Analysis

Cluster Analysis

The cluster analysis package is designed to analyze the output of a clustering algorithm run on a set of SimCalorimeterHits. Example drivers are provided in org.lcsim.contributed.calanal for clustering and analyzing using different clusterers. (NearestNeighbor, Cone, MST, and DirectedTree) At any point in a reconstruction algorithm where clustering is done, simply use the following code.

```
add(new ClusterAnalysisDriver(clusterlists,hitlists,[FSParticlelist],[plot folder]));
```

and a multitude of plots analyzing the clusters will be generated.

Definitions

Each MCParticle of interest contributes to n clusters and has m unclustered hits. The cluster containing the most energy from this particle is its MAX cluster.

Each cluster contains contributions from n particles. The particle with the most energy contributed is the MAX particle. A contributing particle is a "primary" contributor if this cluster is the MAX cluster for that particle. Else this contribtor is a "fragment" contibuter. The cluster is considered "primary" if its MAX particle is a "primary" contributor. Else the cluster is considered a "fragment".

For the default analysis plots, a particle is considered visible if it contibutes to any hit in the CalorimeterHit collections. A particle is considered "found" if it is the MAX particle for its MAX cluster. (Exception: overlapping photons from the same pi0 are both considered "found")

ClusterAnalysisDriver

ClusterAnalysis Driver constructor

type	name	description	default
String[]	Clusterlist	names in event	none
String[]	Hit	collection names in event	EcalBarrHits,EcalEndcapHits,HcalBarrHits,HcalEndcapHits
String	fsList	Final state particle list in event	SimFinalStateParticles
String	folder	Folder name for plots	Clusterlist[0]

processEvent

1. On first event, look for final state particle list. If it is in event, assume it will be in every event. If not, make the list and assume it needs to be created for each event.
2. Create the ClusterAnalysis lists.
3. Analyse the event.
    - setAnalyzer(ClusterAnalysis class): replace the default analysis package

CreateClusterAnalysisLists

String[] Clusterlist names in event

String[] Hit collection names in event

String Final state particle list in event

String pclist - name of List<MCPClusterInfo> to write to event

String cplist - name of List<ClusterMCPInfo> to write to event

CreateLists(Event): procedure

Get lists from event. If any are missing, write empty pclist and cplist to event.

Make a list of clusters, including all clusters from all cluster lists.

Make a list of CalorimeterHits, including all hits from all hit collections. Call it unclustered.

Add a ClusterMCPInfo object to cplist for each cluster.

Add a MCPClusterInfo object to pclist for each final state particle, and for any particle that

contributes to a hit but does not map to a final state particle.

Fill the MCPClusterInfo and ClusterMCPInfo classes, removing hits in clusters from

unclustered list.

MCPClusterInfo::

void AddCluster(Cluster c)
  Add a cluster this particle contributes to
void AddUnclusteredHit(SimCalorimeterHit h)
  Add an unclustered hit this particle contributes to

List<Cluster> getClusters()
  Return a list of clusters to which this particle contributes
double[] getEcorrectedContribution()
  Return an array of corrected energy contributions of this particle to the list of clusters
double[] getErawContribution()
  Return an array of raw energy contributions of this particle to the list of clusters

Cluster getMaxCluster()
  Return the cluster with the maximum energy contributed by this particle

ClusterMCPInfo getMaxCMCP()
  Return the ClusterMCPInfo object for the cluster with the maximum energy contributed by this particle

double getMaxEcorrected()
  Return the corrected energy contributed by this particle to the cluster with the maximum energy contributed by this particle
double getMaxEraw()
  Return the raw energy contributed by this particle to the cluster with the maximum energy contributed by this particle
int getMaxNHits()
  Return the number of hits contributed by this particle to the cluster with the maximum energy contributed by this particle

MCParticle getMCParticle()
  Return the MCParticle for which the cluster information is stored
int getNClusters()
  Return the number of clusters to which this MCParticle contributes
int[] getNhitContribution()
  Return an array of number of hits contributed by this particle to the list of clusters
double getTotalEcorrected()
  Return the total corrected energy in the CalorimeterHit collections contributed by this particle
double getTotalEraw()
  Return the total raw energy in the CalorimeterHit collections contributed by this particle
int getTotalHits()
  Return the total number of hits in the CalorimeterHit collections contributed by this particle

List<SimCalorimeterHit> getUnclusteredHits()
  Return a list of unclustered hits contributed to by this particle
boolean isFinalState()
  Return true if the MCParticle for which the cluster information is stored is a final state particle
void setMaxCMCP(ClusterMCPInfo ci)

ClusterMCPInfo::
void FillInfo(Map<MCParticle,MCPClusterInfo> m)
  Fill in the necessary information after the MCPClusterInfo objects are filled.

Cluster getCluster()

MCParticle[] getContributers()
  Return an array of MCParticles contributing to this cluster

MCParticle[] getFragments()
  Return an array of MCParticles contributing to this cluster having a different maximum cluster

MCParticle getMaxContributer()
  Return the MCParticle with the maximum contribution to this cluster
double getMaxCorrectedEnergy()
  Return the corrected energy from the MCParticle with the maximum contribution to this cluster

MCPClusterInfo getMaxMCPC()
  Return the MCPClusterInfo block for the maximum contributer
int getMaxNHits()
  Return the number of hits from the MCParticle with the maximum contribution to this cluster
double getMaxRawEnergy()
  Return the raw energy from the MCParticle with the maximum contribution to this cluster
int getNContributers()
  Return the number of MCParticles contributing to this cluster
int getNFragment()
  Return the number of MCParticles contributing to this cluster having a different maximum cluster
int getNPrimary()
  Return the number of MCParticles having this cluster as their maximum cluster

MCParticle[] getPrimaries()
  Return an array of MCParticles having this cluster as their maximum cluster