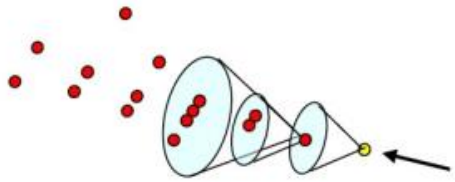
Pandora clustering algorithm

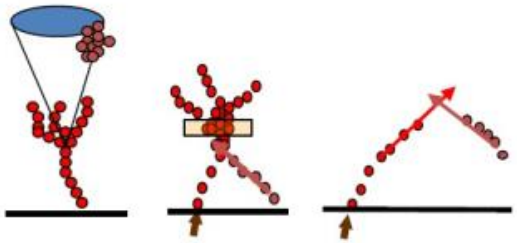
FANGYI GUO

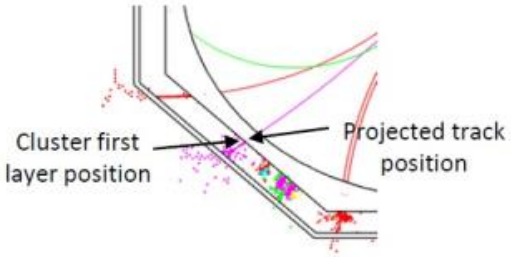
Pandora Algorithms (illustrated)



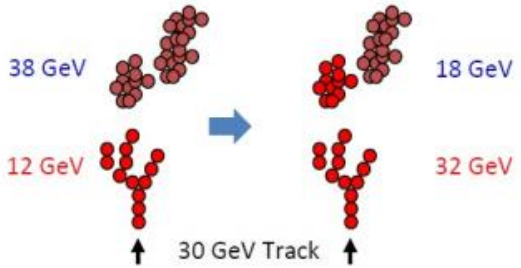
ConeClustering Algorithm

Topological Association Algorithms



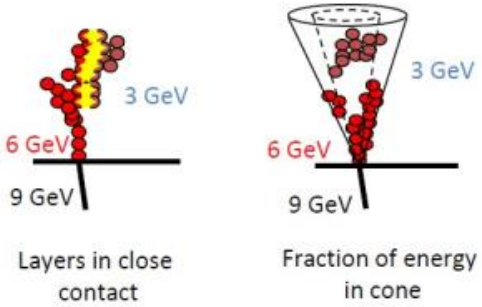


Track-Cluster Association Algorithms

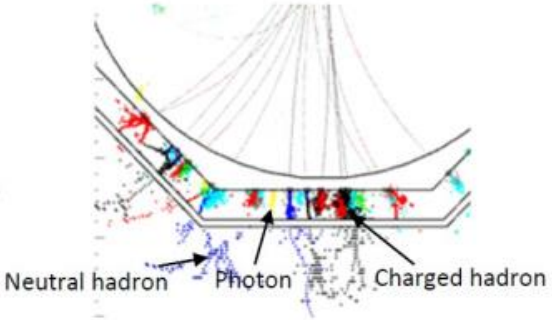


Reclustering Algorithms

Fragment Removal Algorithms



PFO Construction Algorithms



Reconstruction Process

<algorithm type = "EventPreparation"/>

<algorithm type = "MuonReconstruction">

<algorithm type = "PhotonReconstruction">

```
<algorithm type = "ClusteringParent">
    <algorithm type = "ConeClustering" description = "ClusterFormation"/>
    <algorithm type = "TopologicalAssociationParent" description = "ClusterAssociation">
        <associationAlgorithms>
            <algorithm type = "LoopingTracks"/>
            <algorithm type = "BrokenTracks"/>
            <algorithm type = "ShowerMipMerging"/>
            <algorithm type = "ShowerMipMerging2"/>
            <algorithm type = "BackscatteredTracks"/>
            <algorithm type = "BackscatteredTracks2"/>
            <algorithm type = "ShowerMipMerging3"/>
            <algorithm type = "ShowerMipMerging4"/>
            <algorithm type = "ProximityBasedMerging">
                <algorithm type = "TrackClusterAssociation"/>
            </algorithm>
            <algorithm type = "ConeBasedMerging">
                <algorithm type = "TrackClusterAssociation"/>
            </algorithm>
            <algorithm type = "MipPhotonSeparation">
                <algorithm type = "TrackClusterAssociation"/>
            </algorithm>
            <algorithm type = "SoftClusterMerging">
```

ClusteringParent

-Run initial cluster formation algorithm

```
    const ClusterList *pClusterList = NULL;
    PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, PandoraContentApi::RunClusteringAlgorithm(*this, m_clusteringAlgorithmName, pClu
sterList));
```

-Run topological association algorithm

```
    if (!pClusterList->empty() && !m_associationAlgorithmName.empty())
        PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, PandoraContentApi::RunDaughterAlgorithm(*this, m_associationAlgorithmName));
```

-Save new clusters list

Input:

```
    const CaloHitList *pCaloHitList = NULL;
    PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, PandoraContentApi::GetCurrentCaloHitList(*this, pCaloHitList));

    StatusCode PandoraContentApiImpl::GetCurrentCaloHitList(const CaloHitList *&pCaloHitList, std::string &caloHitListName) const
    {
        return m_pPandora->m_pCaloHitManager->GetCurrentList(pCaloHitList, caloHitListName);
    }
```

ConeClusteringAlgorithm

```
//Get CaloHit list and give it into orderedCaloHitList (assump pCaloHitList is in order) #fangyi
//class OrderedCaloHitList : public std::map<PseudoLayer, CaloHitList *> #fangyi
OrderedCaloHitList orderedCaloHitList;
PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, orderedCaloHitList.Add(*pCaloHitList));

//Create a cluster beginning with adding a track.
ClusterVector clusterVector;
PANDORA_RETURN_RESULT_IF(STATUS_CODE_SUCCESS, !=, this->SeedClustersWithTracks(clusterVector));
```

SeedClustersWithTracks(){

Create an empty cluster

Read in tracks

if(use track) PandoraContentApi::Cluster::Create(*this, pTrack, pCluster));

}

//Loop the calohit list in each pseudo layer (orderedCaloHitList)

In each layer:

FindHitsInPreviousLayers

FindHitsInSameLayer

UpdateClusterProperties

ConeClusteringAlgorithm

FindHitsInPreviousLayers (PseudoLayer, CaloHitList, &clusterVector){

For each hit in the hitlist, find the cluster with:

smallest distance;

same distance but highest hadronic energy.

//can choose cluster in all 3 stepback layer or current search layer.

If(find cluster) add hit into cluster; remove hit from hitlist.

}

FindHitsInSameLayer(PseudoLayer, CaloHitList, &clusterVector){

//clustering the remain hits. For those hits belong to no cluster, create a new cluster. Finally erase all hits in the CaloHitList.

}

UpdateClusterProperties(&clusterVector)

Fit cluster to get chi2 and direction.

ConeClusteringAlgorithm

Conclusion:

- High freedom
- 2 set of parameters: fine and coarse
- Geometry information only from pseudo layer and calohit position (CaloHit::getPosition()).
- Energy is only used in clustering.
- Shower shape: cone size + pseudo layer number

TopologicalAssociation

Main process:

- Sort the cluster, mainly from inner pseudo layer to outer pseudo layer.
- Fit cluster to get direction (may not in all pseudo layer)
- Loop over all combinations, compare fit results to determine whether the cluster should be merged:

  Fit direction

  Centroid vector in 2 layer

Each process has it's logic, designed for different topological condition.

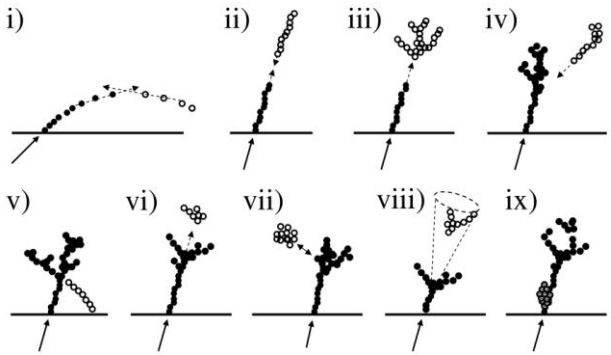


Figure 4: The main topological rules for cluster merging: i) looping track segments; ii) track segments with gaps; iii) track segments pointing to hadronic showers; iv) track-like neutral clusters pointing back to a hadronic shower; v) back-scattered tracks from hadronic showers; vi) neutral clusters which are close to a charged cluster; vii) a neutral cluster near to a charged cluster; viii) cone association; and ix) recovery of photons which overlap with a track segment. In each case the arrow indicates the track, the filled points represent the hits in the associated cluster and the open points represent the hits in the neutral cluster.

backup

ConeClusteringAlgorithm

1. GetDistance ∪

if (第一层). 返回 hit 与 track 到 Calo 投影点的距离.

if (hit position 与 track/cluster direction 角度 < θ_{min})

return UNCHANGE.

↓ fit 成功 → fit 方向

fit 失败 → initial 方向.

if (hit 在 search layer 中) // if 算本层的 hit - cluster 距离.

distance = $\frac{d_{hit}}{D_{cut}}|_{min}$. d_{hit}: 与其他 hit 间距.

D_{cut} = PadWidth × CellLength.

Fine/coarse 2套值.

else:

定义3种距离:

initialDirectionDistance: hit 与 cluster 初方向的距离/S

currentDirectionDistance: hit 与 cluster fit 方向距离/S

trackSeedDistance: hit 到 track 投影点距离/S.

最终取3者最小值.

ConeClusteringAlgorithm

ConeApprochDistance: Loop cluster中所有hit，返回最小d.

Input: cluster中一个hit位置，cluster方向

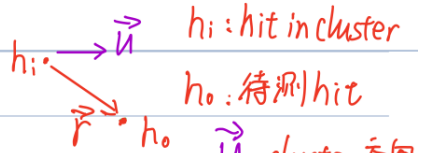
$d_{\parallel} = |\vec{r} \cdot \vec{u}|$

$d_{\perp} = |\vec{r} \times \vec{u}|$

$d_{cut} = d_{\parallel} \tan A + b \cdot D_{pad}$

PadWidth　CellLength.

$d = d_{\perp} / d_{cut}$.



DistanceToTrackSeed:

Input: cluster, hit.

$f = 1 + D_{trk} \cdot |\vec{r}| / S_{trk}^{max}$

$d_{cut} = f \cdot b \cdot D_{pad}$

$d_{\perp} = |\vec{u} \times \vec{r}|$

$d = d_{\perp} / d_{cut}$