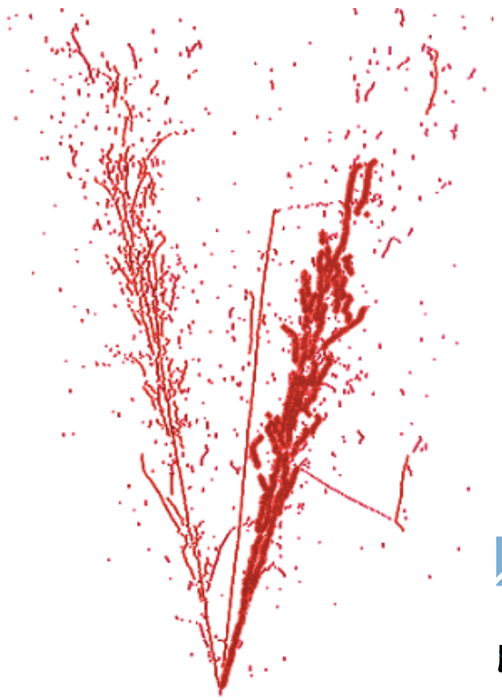
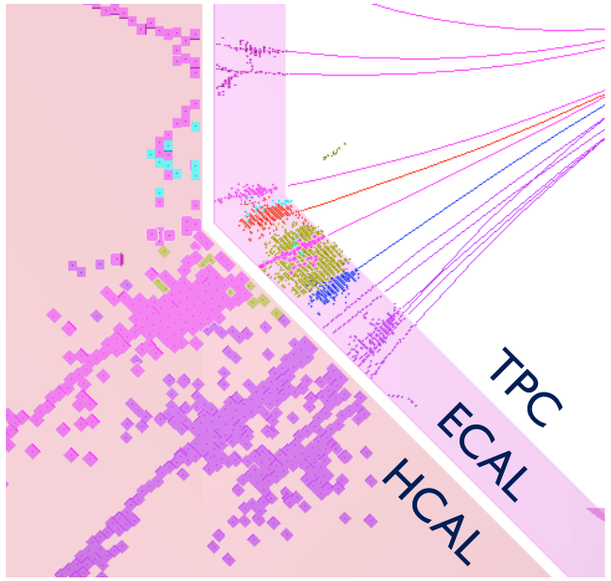


Pandora LC Reconstruction

J. S. Marshall, M. A. Thomson

3 November 2015



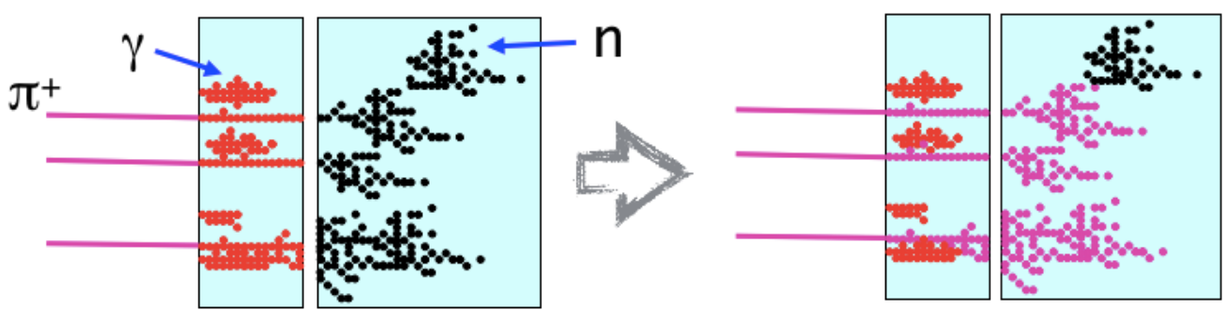
Overview

- This talk will discuss calorimeter pattern recognition and the particle flow approach to calorimetry.
- Functionality provided by the Pandora SDK allows implementation of novel, multi-algorithm approaches to solving pattern recognition problems.
- The Pandora SDK is a mature, well-documented software-engineering project with a growing number of use-cases in HEP.
- Today: review key features of the Pandora SDK, provide details of working with Pandora apps/algs and discuss recent LC development highlights.

Pattern Recognition

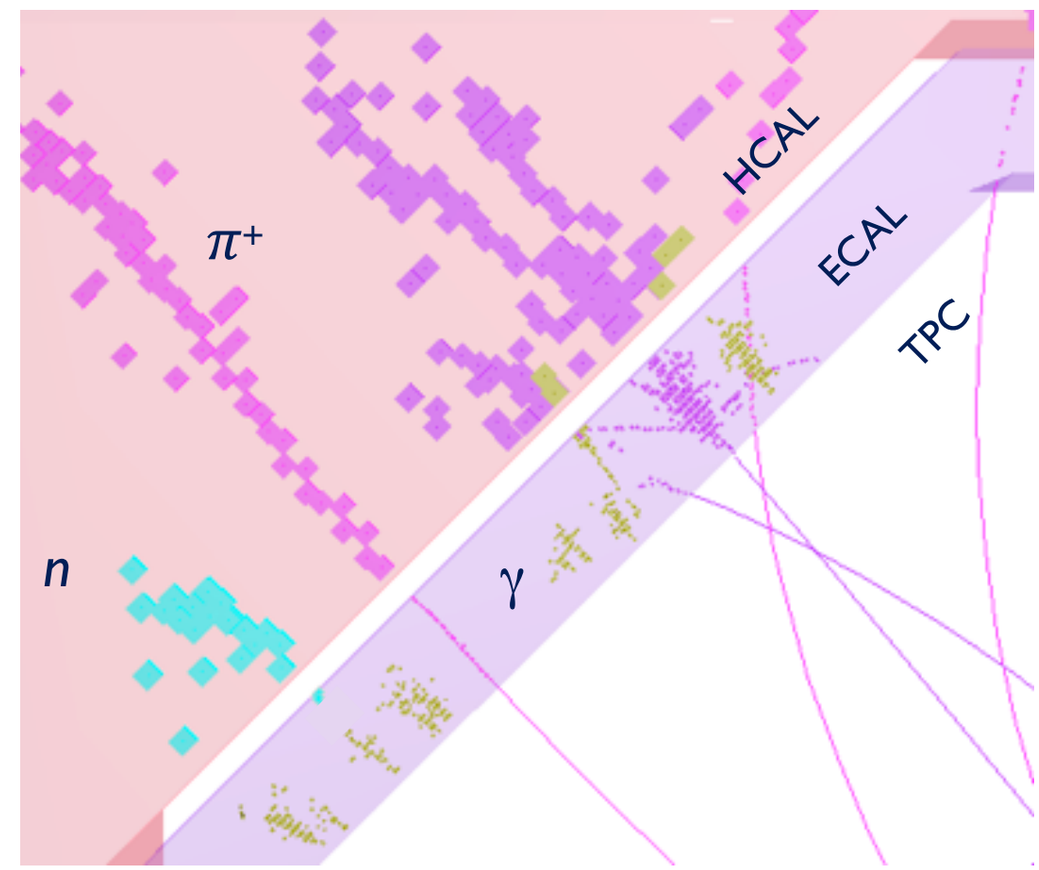
- The calorimeters designed for use at a future e^+e^- collider can image particle interactions in unprecedented detail. Recorded events contain wealth of information for use in physics analyses.
- The human brain is amazing at pattern recognition and can readily reconstruct most event topologies. This guides the **Pandora** approach to automated computer pattern recognition.

- Particle flow approach to calorimetry: just one key example of the advantages of fine granularity, imaging detectors.



$E_{JET} = E_{ECAL} + E_{HCAL}$

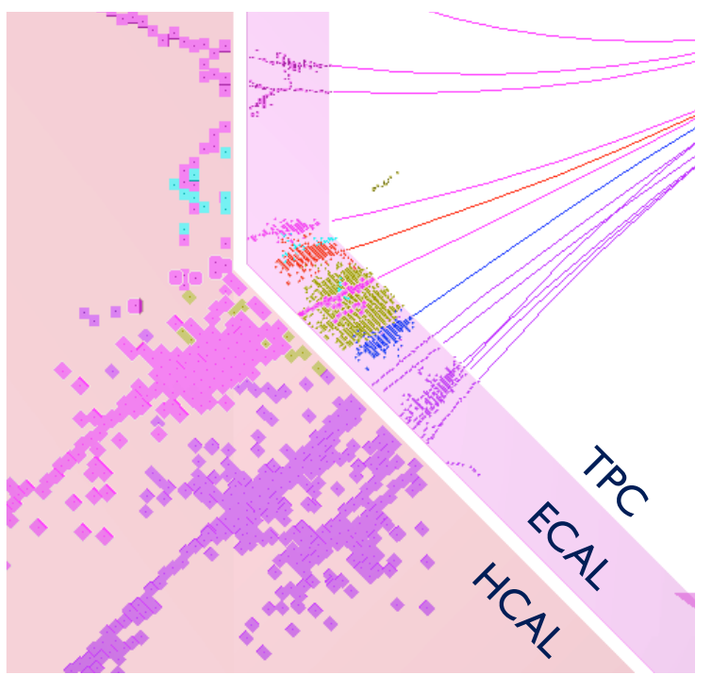
$E_{JET} = E_{TRACK} + E_{\gamma} + E_{n}$



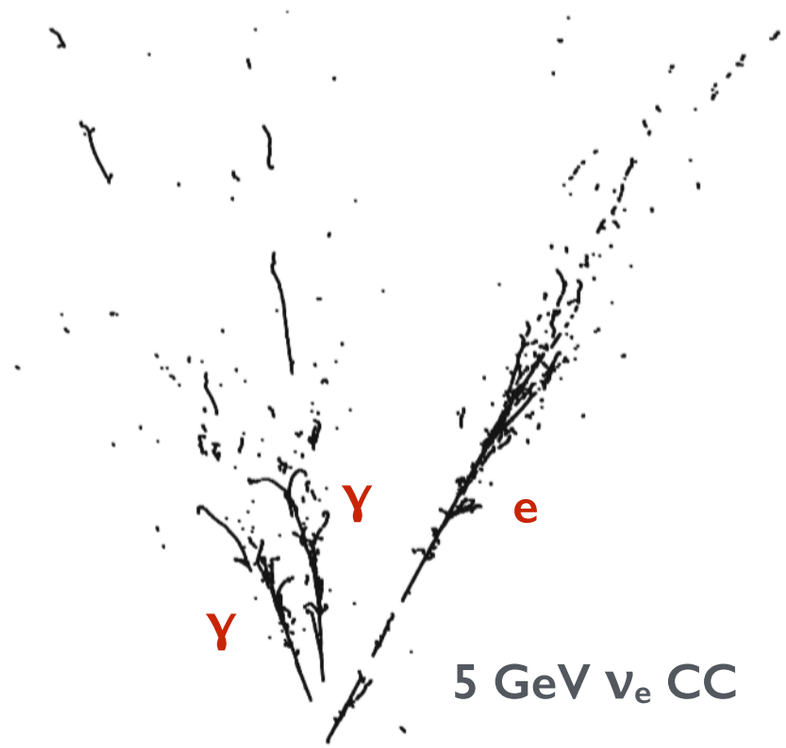
Multi-Algorithm Approach

- The **Pandora** approach to automated computer pattern recognition uses large numbers (70+) of algorithms, each carefully developed to address specific event topologies, without mistakes.
- Significant change from original "energy flow" approach, inferring presence of neutral hadrons via energy excesses. Now exploit calorimeter granularity to gradually build-up picture of events.

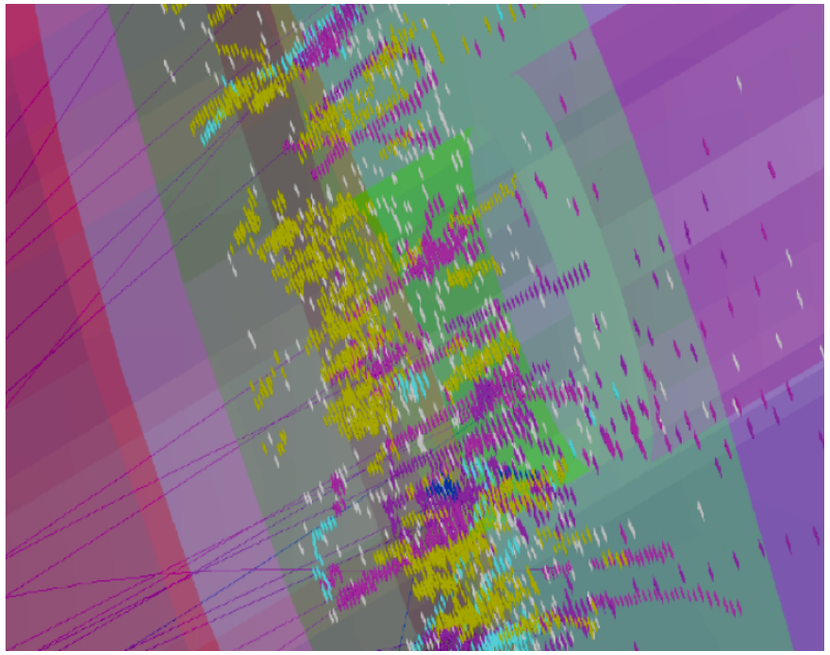
ILC/CLIC event topology



Example LAr TPC event topology



Showers in CMS HGCAL



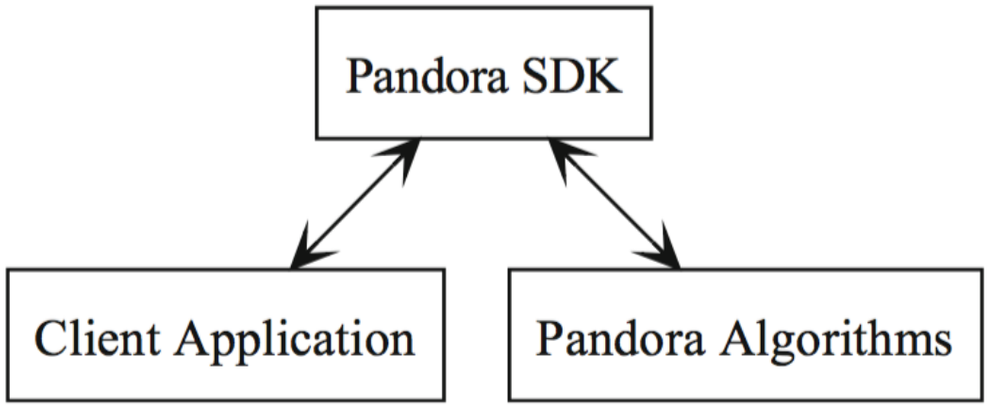
Current Pandora use-cases: ILC (NIMA.2009.09.009), CLIC (NIMA.2012.10.038), LAr TPC reco at DUNE/MicroBooNE (arXiv:1307.7335, 1506.05348) and CMS HGCAL upgrade (LHCC-P-008).

Pandora Introduction

A multi-algorithm event reconstruction can be difficult to implement. The **Pandora Software Development Kit** has been carefully engineered to provide a software environment in which:

1. It is easy for users to provide the building-blocks that define a pattern recognition problem.
2. Logic required to solve pattern recognition problems is cleanly implemented in algorithms.
3. Operations to access or modify building-blocks, or to create new structures, are requested by algorithms and performed by the Pandora framework.

It actively promotes use of large numbers of algorithms, each addressing specific event topologies.



https://www.github.com/PandoraPFA

EPJC.75.439

The Pandora Software Development Kit for Pattern Recognition

J. S. Marshall[a,*], M. A. Thomson[a]

[a] *Cavendish Laboratory, University of Cambridge, Cambridge, United Kingdom*

Abstract

The development of automated solutions to pattern recognition problems is important in many areas of scientific research and human endeavour. This paper describes the implementation of the Pandora Software Development Kit, which aids the process of designing, implementing and running pattern recognition algorithms. The Pandora Application Programming Interfaces ensure simple specification of the building-blocks defining a pattern recognition problem. The logic required to solve the problem is implemented in algorithms. The algorithms request operations to create or modify data structures and the operations are performed by the Pandora framework. This design promotes an approach using many decoupled algorithms, each addressing specific topologies. Details of algorithms addressing two pattern recognition problems in High Energy Physics are presented: reconstruction of events at a high-energy e^+e^- linear collider and reconstruction of cosmic ray or neutrino events in a liquid argon time projection chamber.

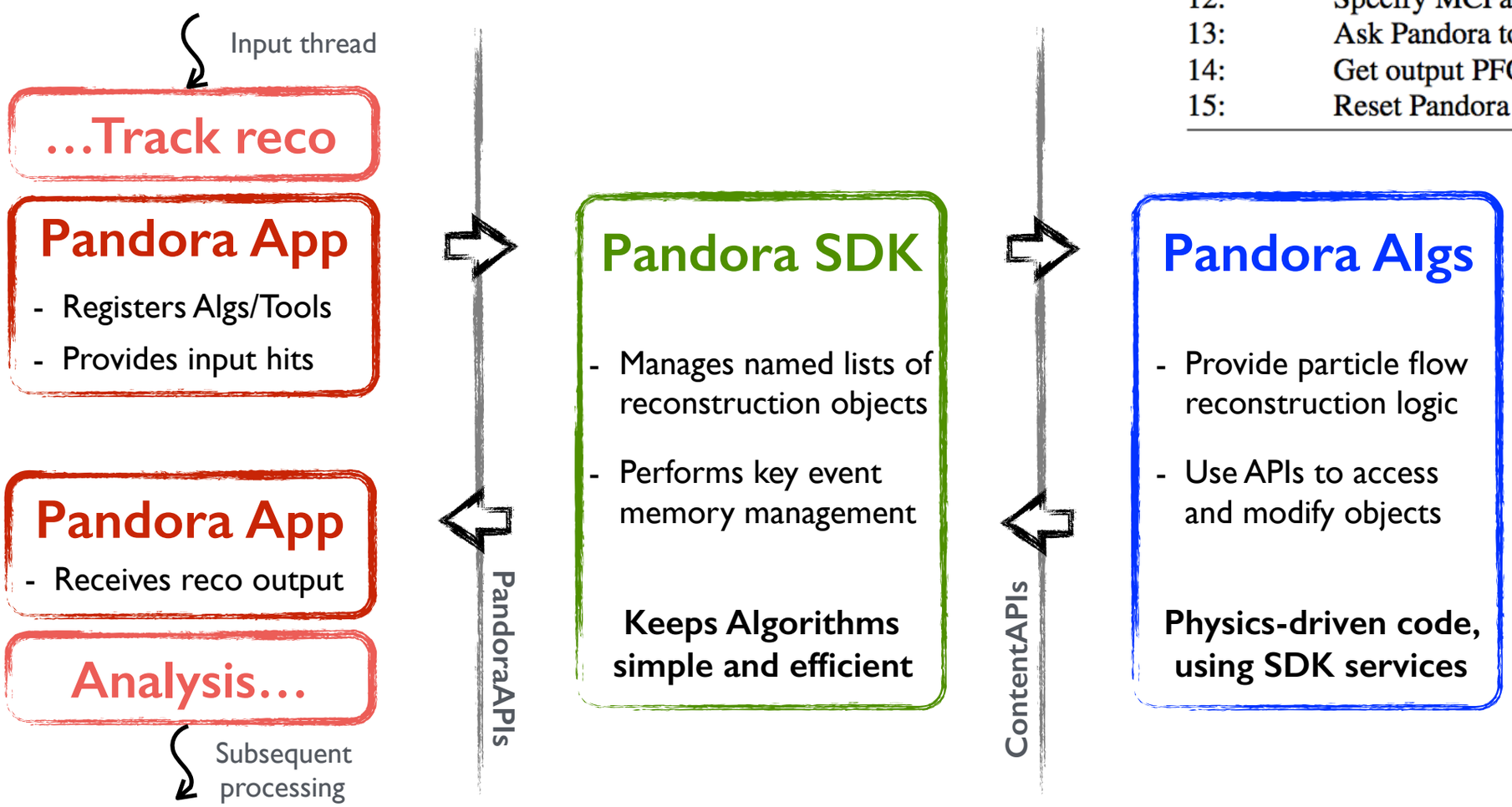
Keywords: Software Development Kit, Pattern recognition, High Energy Physics

Pandora in iLCSoft

- Pandora is **not** a replacement/alternative to iLCSoft. It is an ideal framework for pattern recognition. Carefully designed APIs enable multi-alg approach.
- Client App creates Pandora instance(s), registers algs and provides alg config. Each event, it passes details about Hits, Tracks to Pandora and receives Particles.

Algorithm 3 Pseudocode description of a client application used for Linear Collider event reconstruction.

```
procedure MAIN
    Create a Pandora instance
    Register Algorithms and Plugins
    Provide detector geometry description
    Ask Pandora to parse XML settings file
    for all Events do
        Create Track instances
        Create CaloHit instances
        Create MCParticle instances
        Specify Track-Track relationships
        Specify MCParticle-Track relationships
        Specify MCParticle-CaloHit relationships
        Ask Pandora to process the event
        Get output PFOs and write to file
        Reset Pandora before next event
```

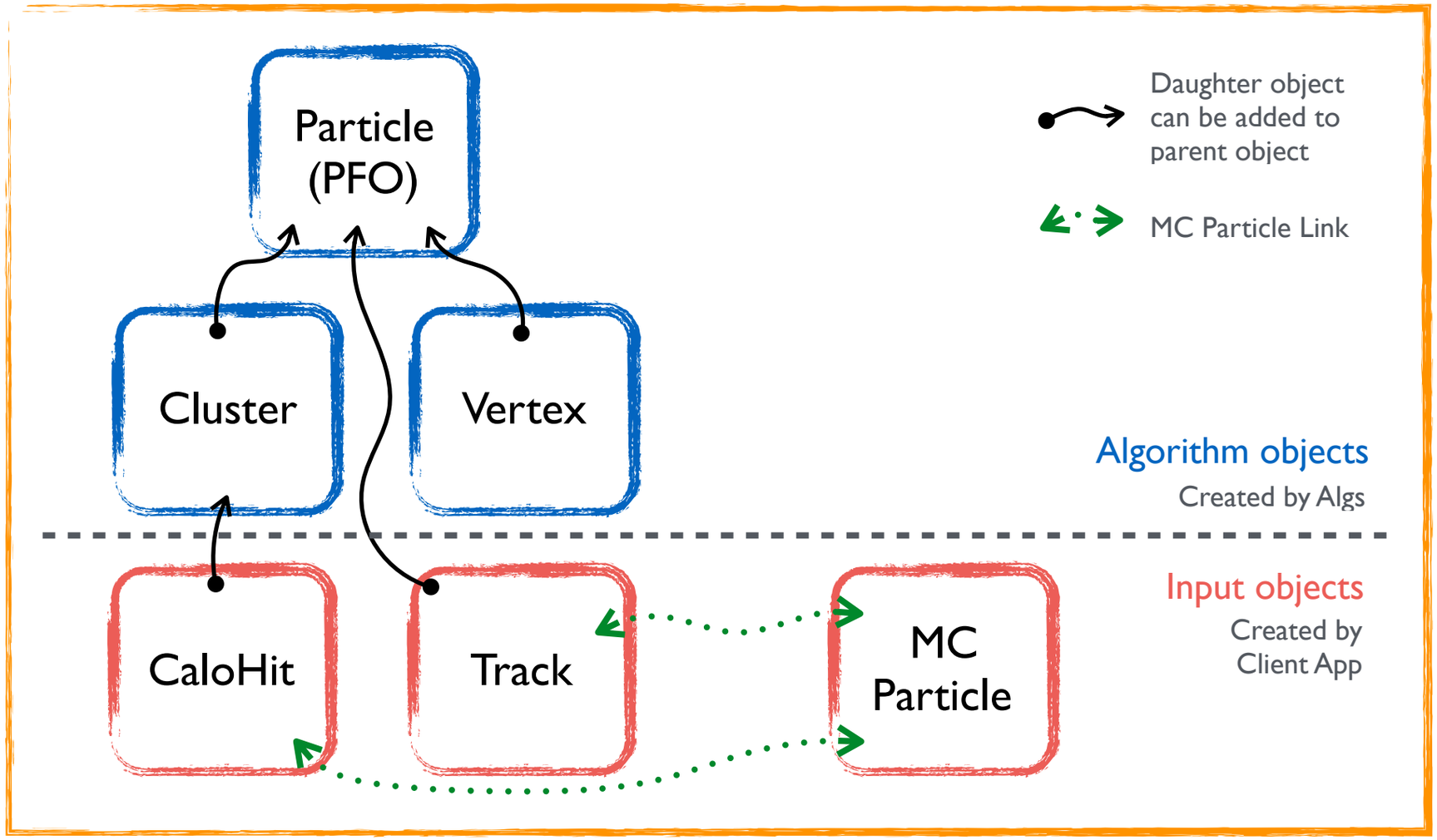


https://www.github.com/PandoraPFA

Pandora SDK

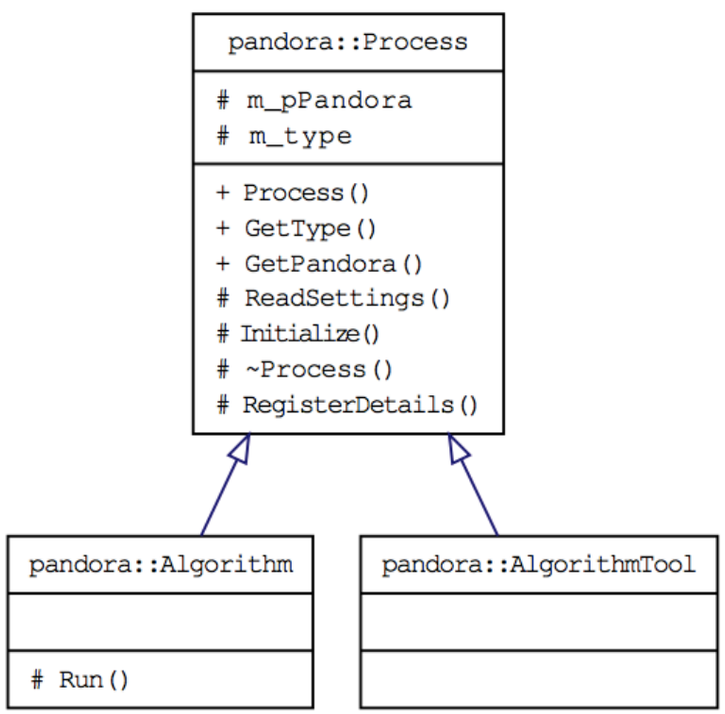
- The Pandora SDK provides a comprehensive **Event Data Model** (EDM) for managing pattern recognition problems. Instances of objects in the EDM are owned by **Pandora Managers**.
- The object instances are stored in named lists and the Managers are able to create new objects, delete objects, create and save new lists and move objects between lists.
- The Managers provide a complete set of low-level operations that allow the high-level operations requested by pattern recognition algorithms to be satisfied.

```
pandora::Pandora
- m_pAlgorithmManager
- m_pCaloHitManager
- m_pClusterManager
- m_pGeometryManager
- m_pMCManager
- m_pPfoManager
- m_pPluginManager
- m_pTrackManager
- m_pVertexManager
- m_pPandoraSettings
- m_pPandoraApiImpl
- m_pPandoraContentApiImpl
- m_pPandoraImpl
+ Pandora()
+ ~Pandora()
+ GetPandoraApiImpl()
+ GetPandoraContentApiImpl()
+ GetSettings()
+ GetGeometry()
+ GetPlugins()
- PrepareEvent()
- ProcessEvent()
- ResetEvent()
- ReadSettings()
```



Pandora Algorithms

- Pandora algorithms contain the step-by-step instructions for finding patterns in the provided data.
- They use the APIs to access objects and to request the Managers to make new objects or modify existing objects.
- They inherit from the **Process** class, which provides functionality for handshaking with Pandora, XML config and function callbacks.



Algorithm 1 Cluster creation pseudocode. The logic determining when to create new Clusters and when to extend existing Clusters will vary between algorithms.

```
procedure CLUSTER CREATION
    Create temporary Cluster list
    Get current CaloHit list
    for all CaloHits do
        if CaloHit available then
            for all newly-created Clusters do
                Find best host Cluster
            if Suitable host Cluster found then
                Add CaloHit to host Cluster
            else
                Add CaloHit to a new Cluster
    Save new Clusters in a named list
```

Algorithm 2 Cluster merging pseudocode. The logic governing the identification of suitable parent Clusters and daughter Clusters will vary between algorithms.

```
procedure CLUSTER MERGING
    Get current Cluster list
    for all Clusters do
        if Cluster is suitable parent then
            for all Clusters do
                Find best daughter Cluster
            if Suitable daughter Cluster found then
                Merge daughter Cluster into Parent
```

Pandora LC Algorithms

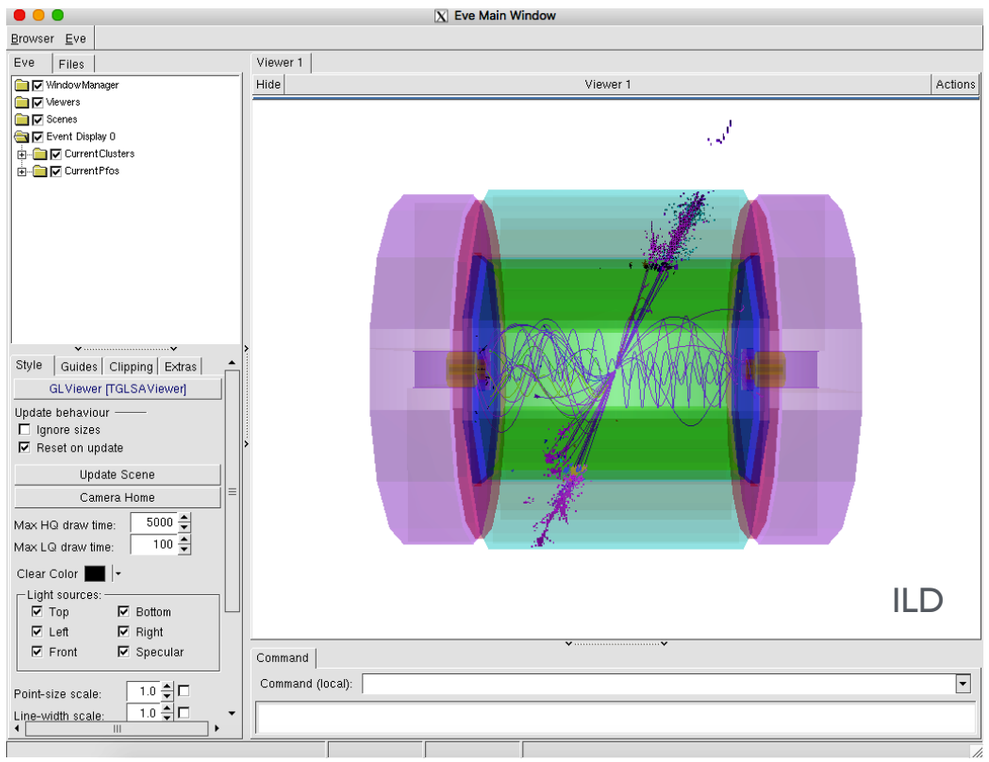
- Idea is that algorithms focus on physics, with Pandora performing memory-management, etc. Rapid development is possible in a standalone environment, with visual debugging.
- We currently have around 80 algorithms, tools, plugins, helper functions, etc. for LC (and CMS HGCAL) reconstruction, grouped in the LCPandoraContent library.
- Some algorithms are complex and sophisticated, whilst others are pleasingly simple. The algorithms gel together to provide a coherent and robust reconstruction.

- Alg selection and config specified via XML file. Currently available for LC:
  - Basic/core Pandora reconstruction
  - Plus standalone muon reco
  - Plus standalone photon reco
- For development use only:
  - Options to cheat pattern recognition for specific particle types
  - Fully cheated pattern recognition

```xml
<!-- Standalone photon reconstruction -->
<algorithm type = "PhotonReconstruction">
    <algorithm type = "ConeClustering" description = "ClusterFormation">
        <ClusterSeedStrategy>0</ClusterSeedStrategy>
        <ShouldUseTrackSeed>false</ShouldUseTrackSeed>
        <ShouldUseOnlyECalHits>true</ShouldUseOnlyECalHits>
        <ConeApproachMaxSeparation>250.</ConeApproachMaxSeparation>
    </algorithm>
    <algorithm type = "RecoPhotonFragmentMerging" description = "FragmentMerging">
        <NonPhotonClusterListName>NonPhotonClusters</NonPhotonClusterListName>
        <associationAlgorithms>
            <algorithm type = "ProximityBasedMerging">
                <algorithm type = "TrackClusterAssociation"/>
            </algorithm>
            <algorithm type = "SoftClusterMerging">
                <algorithm type = "TrackClusterAssociation"/>
            </algorithm>
        </associationAlgorithms>
    </algorithm>
    <ClusterListName>PhotonClusters</ClusterListName>
    <ReplaceCurrentClusterList>false</ReplaceCurrentClusterList>
    <ShouldMakePdfHistograms>false</ShouldMakePdfHistograms>
    <NEnergyBins>9</NEnergyBins>
    <HistogramFile>PandoraLikelihoodData9EBin.xml</HistogramFile>
    <ShouldDeleteNonPhotonClusters>false</ShouldDeleteNonPhotonClusters>
</algorithm>
```

Example XML snippet - Photon reco

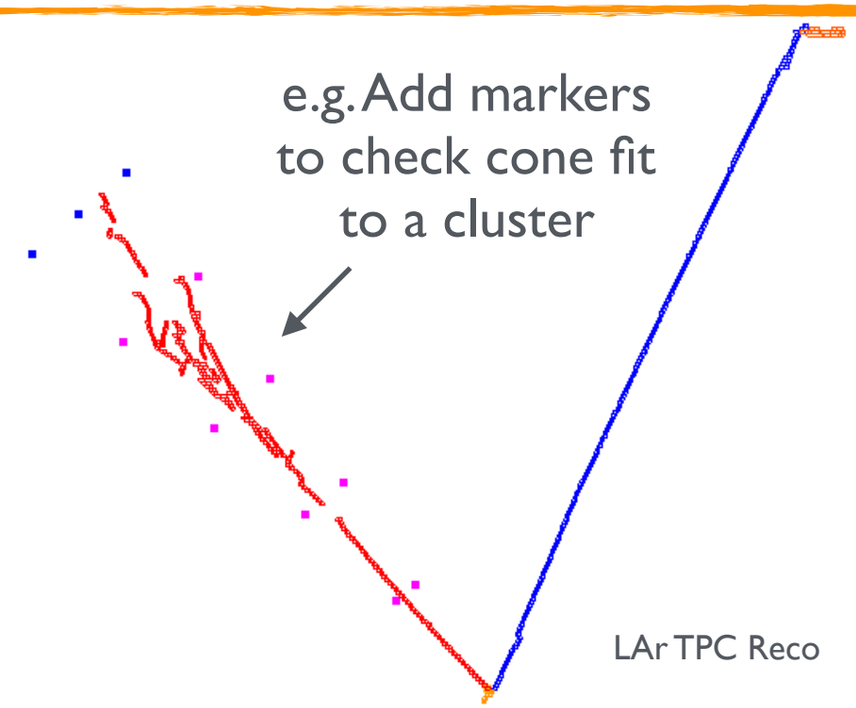
Pandora Visualisation




```
. . .
<algorithm type = “LoopingTracks"/>
<algorithm type = "VisualMonitoring">
    <ShowCurrentClusters>true</ShowCurrentClusters>
</algorithm>
<algorithm type = “BrokenTracks"/>
<algorithm type = "VisualMonitoring">
    <ShowCurrentClusters>true</ShowCurrentClusters>
</algorithm>
<algorithm type = "ShowerMipMerging"/>
<algorithm type = "ShowerMipMerging2"/>
<algorithm type = "BackscatteredTracks"/>
<algorithm type = "BackscatteredTracks2"/>
<algorithm type = "ShowerMipMerging3"/>
<algorithm type = "ShowerMipMerging4"/>
<algorithm type = “ProximityBasedMerging”/>
. . .
```

e.g. Add two event display algs to examine changes as reconstruction progresses

- Pandora algorithms can choose to use the Pandora Monitoring library, which has a ROOT dependency.
- ROOT Event Visualisation Environment provides 2D or 3D displays of hits, clusters, particles or vertices.
- Reusable Pandora event display alg can be added to XML to view status of reconstruction at any point.
- Alternatively, algs can use visualisation APIs to provide custom visual debugging - **rewarding way to work.**



Pandora Key Points

- Pandora is not trying to be iLCSoft. It is a framework for pattern recognition algorithms.
- It is generic and is used successfully across multiple (very different) projects.
- Its powerful functionality enables complex algorithms using e.g. recursion or reclustering.
- Design philosophy: support multi-algorithm approach, gradually build-up picture of event.

Next: Working with Pandora…

Pandora Client App Steering

```xml
<processor name="MyMarlinPandoraDefault" type="PandoraPFANewProcessor">
  <parameter name="PandoraSettingsXmlFile" type="String">PandoraSettingsDefault.xml</parameter>
  <!-- Collection names -->
  <parameter name="TrackCollections" type="StringVec">MarlinTrkTracks</parameter>
  <parameter name="ECalCaloHitCollections" type="StringVec">ECALBarrel ECALEndcap ECALOther</parameter>
  <parameter name="HCalCaloHitCollections" type="StringVec">HCALBarrel HCALEndcap HCALOther</parameter>
  <parameter name="LCalCaloHitCollections" type="StringVec">LCAL</parameter>
  <parameter name="LHCalCaloHitCollections" type="StringVec">LHCAL</parameter>
  <parameter name="MuonCaloHitCollections" type="StringVec">MUON</parameter>
  <parameter name="MCParticleCollections" type="StringVec">MCParticle</parameter>
  <parameter name="RelCaloHitCollections" type="StringVec">RelationCaloHit RelationMuonHit</parameter>
  <parameter name="RelTrackCollections" type="StringVec">MarlinTrkTracksMCTruthLink</parameter>
  <parameter name="KinkVertexCollections" type="StringVec">KinkVertices</parameter>
  <parameter name="ProngVertexCollections" type="StringVec">ProngVertices</parameter>
  <parameter name="SplitVertexCollections" type="StringVec">SplitVertices</parameter>
  <parameter name="V0VertexCollections" type="StringVec">V0Vertices</parameter>
  <parameter name="ClusterCollectionName" type="String">PandoraClustersDefault</parameter>
  <parameter name="PFOCollectionName" type="String">PandoraPFOsDefault</parameter>
  <!-- Calibration constants -->
  <parameter name="ECalToMipCalibration" type="float">160.0</parameter>
  <parameter name="HCalToMipCalibration" type="float">34.8</parameter>
  <parameter name="ECalMipThreshold" type="float">0.5</parameter>
  <parameter name="HCalMipThreshold" type="float">0.3</parameter>
  <parameter name="ECalToEMGeVCalibration" type="float">1.007</parameter>
  <parameter name="HCalToEMGeVCalibration" type="float">1.007</parameter>
  <parameter name="ECalToHadGeVCalibrationBarrel" type="float">1.075</parameter>
  <parameter name="ECalToHadGeVCalibrationEndCap" type="float">1.075</parameter>
  <parameter name="HCalToHadGeVCalibration" type="float">1.027</parameter>
  <parameter name="MuonToMipCalibration" type="float">10.0</parameter>
  <parameter name="DigitalMuonHits" type="int">0</parameter>
  <parameter name="MaxHCalHitHadronicEnergy" type="float">1.</parameter>
  <!-- Absorber properties -->
  <parameter name="AbsorberRadLengthECal" type="float">0.2854</parameter>
  <parameter name="AbsorberIntLengthECal" type="float">0.0101</parameter>
  <parameter name="AbsorberRadLengthHCal" type="float">0.0569</parameter>
  <parameter name="AbsorberIntLengthHCal" type="float">0.0060</parameter>
  <parameter name="AbsorberRadLengthOther" type="float">0.0569</parameter>
  <parameter name="AbsorberIntLengthOther" type="float">0.0060</parameter>
  <!--Whether to calculate track states manually, rather than copy stored fitter values-->
  <parameter name="UseOldTrackStateCalculation" type="int">0</parameter>
</processor>
```

Pandora alg steering

Input and output collection names

Pandora calibration constants

Additional geometry information

Support for old tracking software

Pandora Algorithm Steering

- **Pandora is configured via an XML file, provided by the client application.**
- It looks for algorithm XML tags within the top level Pandora tags, creating instances of any algorithms found. It will run these algorithms, in order, for each event.
- Each algorithm receives a ReadSettings callback, with a provided XML handle. Algorithms can have mandatory or optional parameters (override default values).
- Algorithms can use the ReadSettings callback to control the creation of daughter Algorithms or AlgorithmTools. Allows for use of (multiple) alternative approaches to solving a problem.

```
<!-- Pandora settings xml file -->
<pandora>
    <!-- GLOBAL SETTINGS -->
    <IsMonitoringEnabled>true</IsMonitoringEnabled>
    <ShouldDisplayAlgorithmInfo>false</ShouldDisplayAlgorithmInfo>
    <ShouldCollapseMCParticlesToPfoTarget>true</ShouldCollapseMCParticlesToPfoTarget>

    <!-- PLUGIN SETTINGS -->
    <HadronicEnergyCorrectionPlugins>CleanClusters ScaleHotHadrons</HadronicEnergyCorrectionPlugins>
    <EmShowerPlugin>LCEmShowerId</EmShowerPlugin>
    <PhotonPlugin>LCPhotonId</PhotonPlugin>
    <ElectronPlugin>LCElectronId</ElectronPlugin>
    <MuonPlugin>LCMuonId</MuonPlugin>

    <!-- ALGORITHM SETTINGS -->

    <!-- Set calo hit properties, then select tracks and hits to use for clustering -->
    <algorithm type = "CaloHitPreparation"/>
    <algorithm type = "EventPreparation">
        <OutputTrackListName>Tracks</OutputTrackListName>
        <OutputCaloHitListName>CaloHits</OutputCaloHitListName>
        <OutputMuonCaloHitListName>MuonYokeHits</OutputMuonCaloHitListName>
        <ReplacementTrackListName>Tracks</ReplacementTrackListName>
        <ReplacementCaloHitListName>CaloHits</ReplacementCaloHitListName>
    </algorithm>
    …SNIP…
```

Pandora XML tag opened here

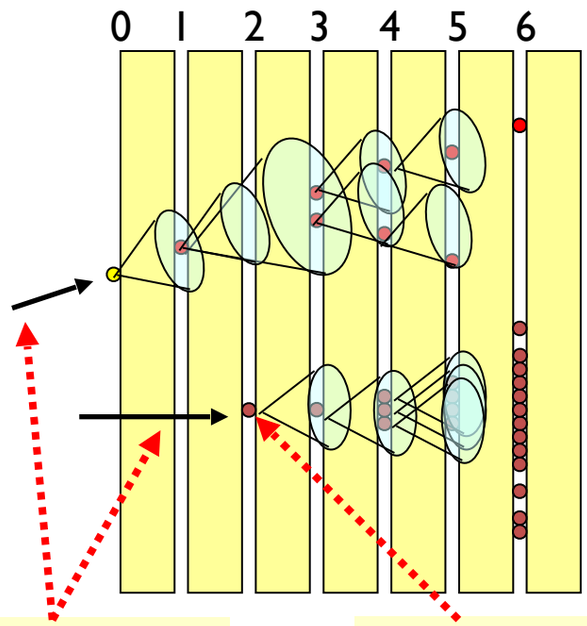
Non-default global parameters

Particle Id and Energy correction Plugins

First two algorithms, with required parameters

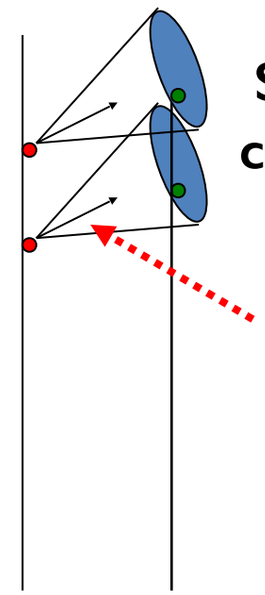
Pandora Clustering Example

- **Philosophy:** "It's easier to put clusters together, than to split them up again."
- Clustering algorithm very careful to avoid accidentally merging energy deposits from separate particles.



Initial cluster direction

Unmatched hit seeds new cluster

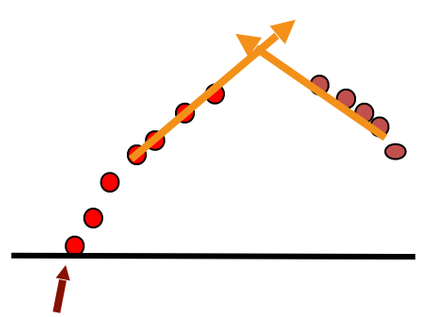


Simple cone-based clustering algorithm

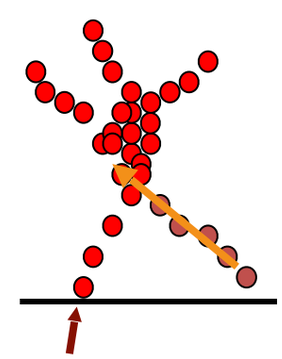
Cones based on either: initial direction or current direction

Topological Associations

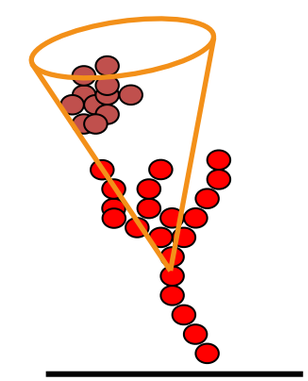
- Fine granularity of the calorimeters exploited to merge cluster fragments that are clearly associated.
- Very few mistakes made.



Looping tracks



Back-scattered tracks



Cone associations

Pandora Clustering Config

Parent clustering algorithm

Cluster formation daughter algorithm

```
<algorithm type = "ClusteringParent">
    <algorithm type = "ConeClusteringFast" 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 = "TrackClusterAssociationFast"/>
            </algorithm>
            <algorithm type = "ConeBasedMerging">
                <algorithm type = "TrackClusterAssociationFast"/>
            </algorithm>
            <algorithm type = "MipPhotonSeparation">
                <algorithm type = "TrackClusterAssociationFast"/>
            </algorithm>
            <algorithm type = "SoftClusterMergingFast">
                <algorithm type = "TrackClusterAssociationFast"/>
                <AdditionalClusterListNames>PhotonClusters</AdditionalClusterListNames>
            </algorithm>
            <algorithm type = "IsolatedHitMerging">
                <AdditionalClusterListNames>PhotonClusters</AdditionalClusterListNames>
            </algorithm>
        </associationAlgorithms>
    </algorithm>
    <ClusterListName>PrimaryClusters</ClusterListName>
    <ReplaceCurrentClusterList>true</ReplaceCurrentClusterList>
</algorithm>
```

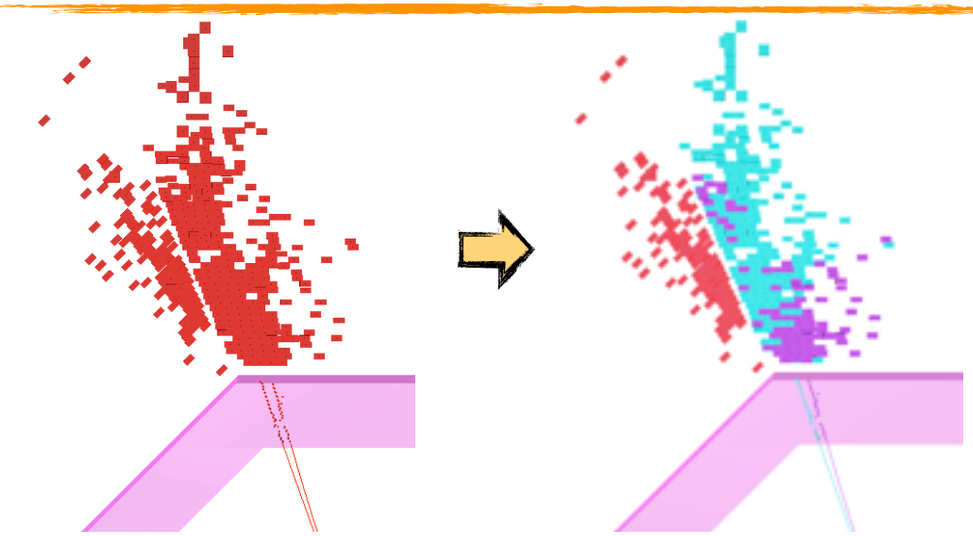
List of topological association daughter algorithms

Some topological association algs use daughter algs

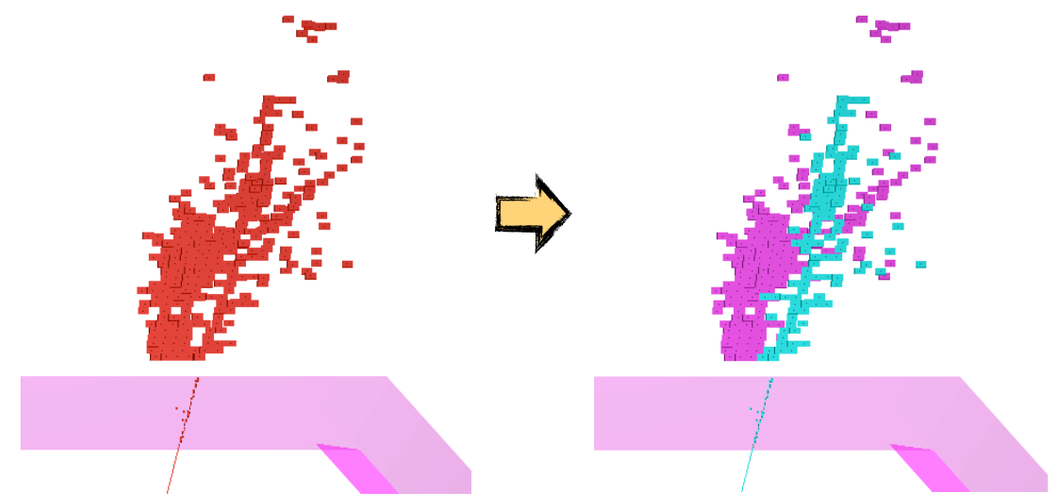
Configuration of output Cluster list

Pandora Reclustering Example

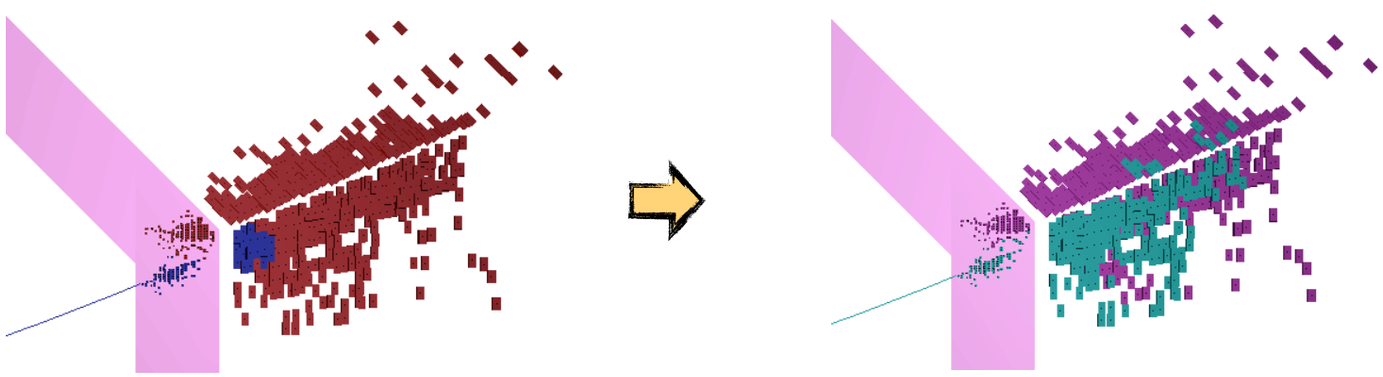
If identify significant discrepancy between cluster energy and associated track momentum, choose to recluster. Alter clustering parameters until cluster splits to obtain track-cluster consistency.



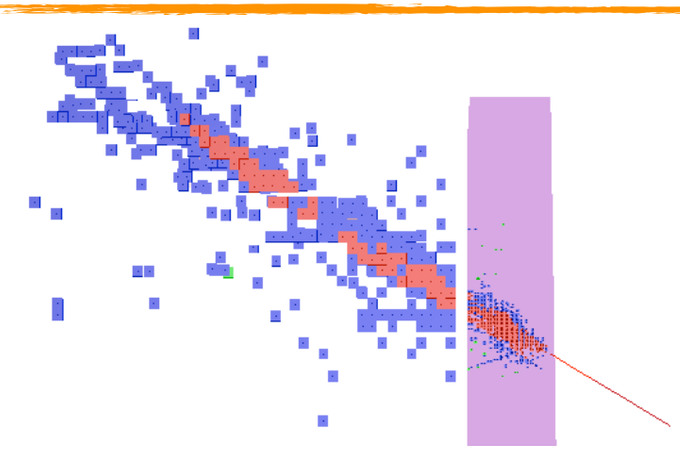
1. Multiple tracks associated to single cluster – split cluster.



2. Cluster energy much greater than track momentum – split cluster.



3. Track momentum much greater than cluster energy – bring in nearby clusters and reconfigure.



4. If, and only if, no E/p match emerges, can force track-cluster consistency ⇒ energy flow.

Pandora Reclustering Config

Parent reclustering algorithm, provides overall logic

```
<algorithm type = "SplitMergedClusters">
    <clusteringAlgorithms>
        <algorithm type = "ConeClusteringFast" instance = "Reclustering1">
            <TanConeAngleFine>0.24</TanConeAngleFine>
            <TanConeAngleCoarse>0.4</TanConeAngleCoarse>
            <AdditionalPadWidthsFine>2</AdditionalPadWidthsFine>
            <AdditionalPadWidthsCoarse>2</AdditionalPadWidthsCoarse>
            <SameLayerPadWidthsFine>2.24</SameLayerPadWidthsFine>
            <SameLayerPadWidthsCoarse>1.44</SameLayerPadWidthsCoarse>
            <MaxTrackSeedSeparation>100</MaxTrackSeedSeparation>
            <MaxLayersToTrackSeed>0</MaxLayersToTrackSeed>
            <MaxLayersToTrackLikeHit>0</MaxLayersToTrackLikeHit>
            <TrackPathWidth>0</TrackPathWidth>
        </algorithm>
        <algorithm type = "ConeClusteringFast" instance = “Reclustering2"/>
        <algorithm type = "ConeClusteringFast" instance = "Reclustering3"/>
        <algorithm type = "ConeClusteringFast" instance = "Reclustering4"/>
        <algorithm type = "ConeClusteringFast" instance = "Reclustering5"/>
        <algorithm type = "ConeClusteringFast" instance = "Reclustering6"/>
        <algorithm type = "ConeClusteringFast" instance = "Reclustering7"/>
        <algorithm type = "ConeClusteringFast" instance = "Reclustering8"/>
        <algorithm type = "ConeClusteringFast" instance = "Reclustering9"/>
        <algorithm type = "ConeClusteringFast" instance = "Reclustering10"/>
        <algorithm type = "ConeClusteringFast" instance = "Reclustering11"/>
        <algorithm type = "ConeClusteringFast" instance = "Reclustering12"/>
    </clusteringAlgorithms>
    <algorithm type = "TopologicalAssociationParent" description = "ClusterAssociation" instance = "reclusterAssociation"/>
    <algorithm type = "TrackClusterAssociationFast" description = "TrackClusterAssociation"/>
    <algorithm type = "ForcedClustering" description = “ForcedClustering"/>
    <UsingOrderedAlgorithms>true</UsingOrderedAlgorithms>
    <ShouldUseForcedClustering>true</ShouldUseForcedClustering>
</algorithm>
```

Alternative clustering daughter algorithm (full config)

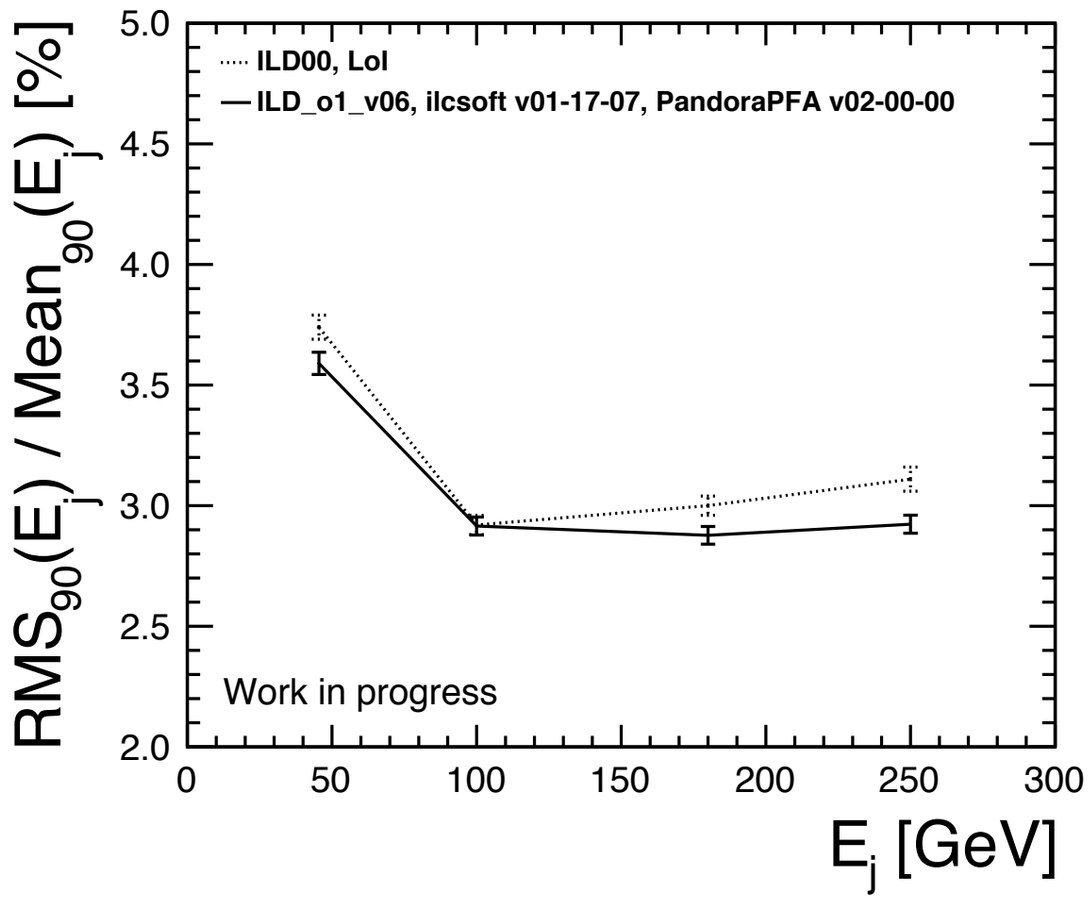
Alternative clustering daughter algorithms (reuse existing instances)

Daughter algorithms providing specific functionality to parent

Steering options for parent algorithm

Pandora Performance

- Pandora particle flow performance is typically characterised by the **Jet Energy Resolution**, measured using Z' particles, which decay into light quarks, producing two mono-energetic jets.
- Performance depends on pattern recognition *and* the estimators used to obtain energy measurements for EM and hadronic showers. The calibration procedure is very important.
- Performance plots shown here use same calibration approach used for ILD LoI. Results show ***consistency*** with and ***progress*** with respect to the LoI results, with clear high E improvement.



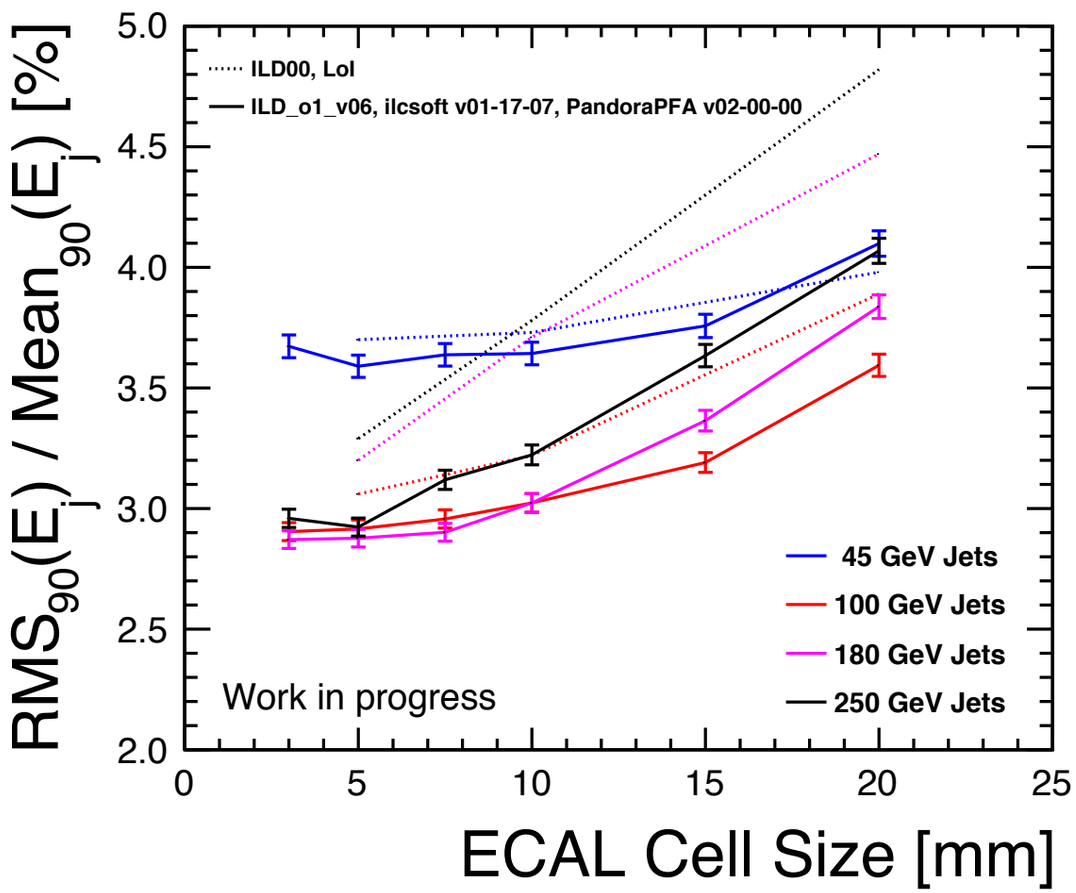
Event type	200 GeV Z'	200 GeV Z'
Background	none	CLIC 3 TeV
Event Time	0.22 ± 0.02 s	12.5 ± 0.4 s
VMem	256 MB	480 MB
RSS	43 MB	266 MB
# Tracks	27 ± 1	650 ± 7
# CaloHits	$4,200 \pm 60$	$39,900 \pm 700$

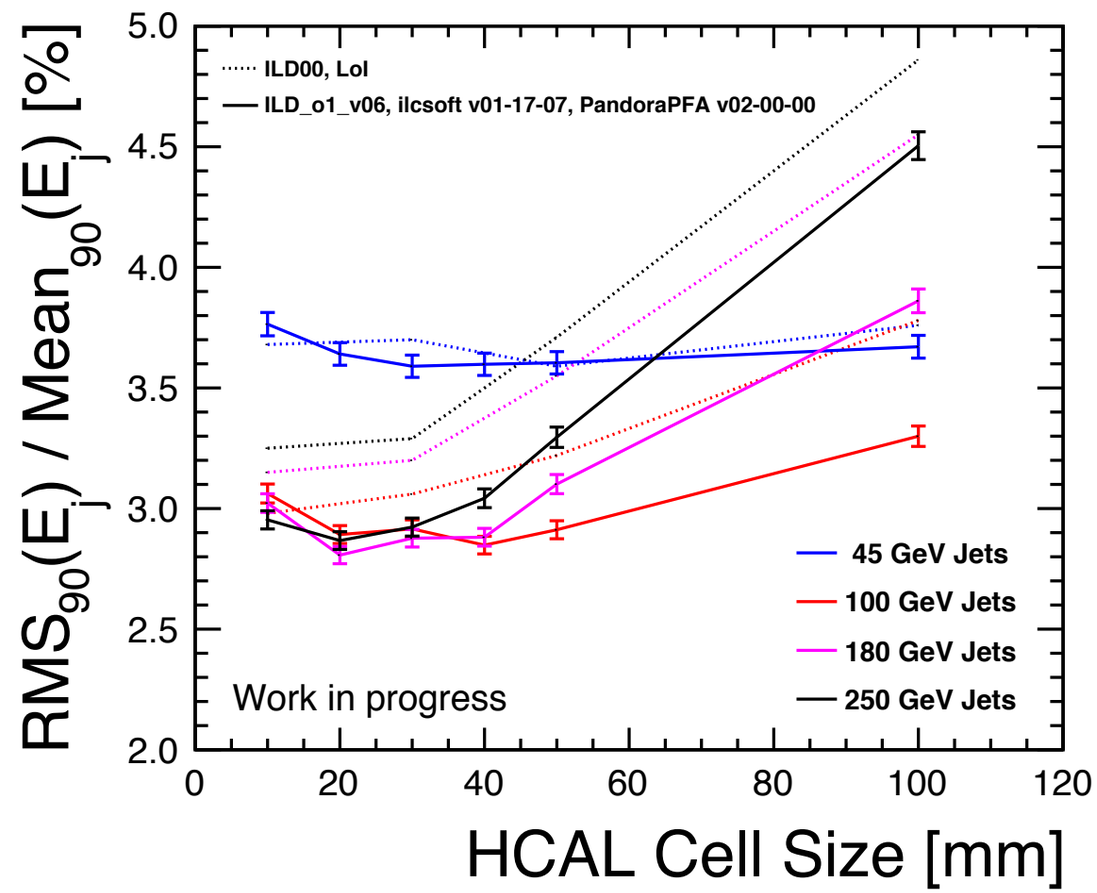
Algorithms	Input Objects	Monitoring	VMem [MB]	RSS [MB]
✗	✗	✗	19	3
✓	✗	✗	19	4
✗	✓	✗	64	47
✓	✓	✗	244	229
✓	✓	✓	480	266

Indicative CPU times and memory footprint

Pandora Performance

- Detector optimisation plots here compare LoI results with those obtained using ilcsoft v01-17-07 (PandoraPFA v02-00-00). The LoI calibration *approach* was applied (DBD constants).
- The optimum calibration approach is still under investigation, so these plots should not yet be used to draw any conclusions about the most appropriate detector parameters.
- **Take-home message:** PandoraPFA v02-00-00+ offers our best ever jet energy resolutions, but users should appreciate the importance of the energy calibration for each detector model.





Pandora Calibration

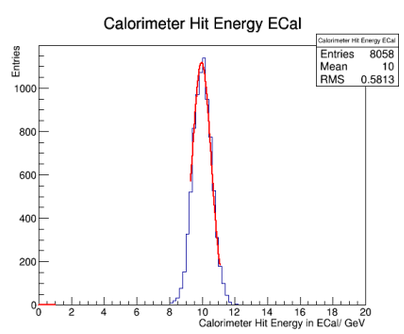
S. Green

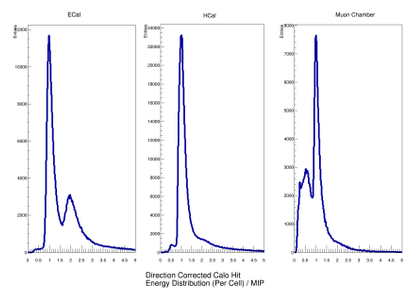
Simulation Output
Energy Deposited in Active Material of Calorimeters

Calorimeter Energy
Energy Deposited in the Total Calorimeter Volume

Particle Energies
Energy of individual reconstructed particles

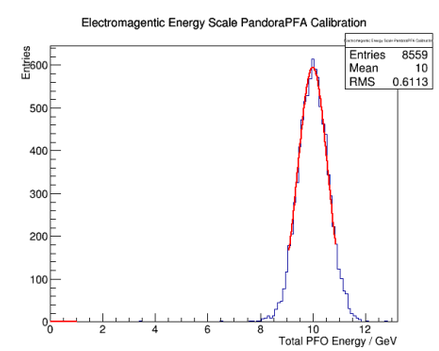
Digitisation
Set by looking at contained kaonL and γ events throughout the calorimeters.

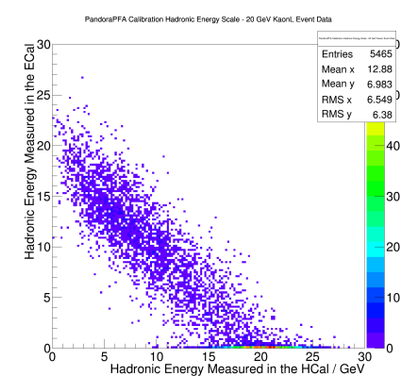




Minimum Ionising Particle (MIP) Detector Response
Determine the response of each part of the detector to a MIP, μ^- events. MIP scale used in PandoraPFA.

Electromagnetic and Hadronic Scale
Electromagnetic/Hadronic scale set using PFO energy of contained γ and kaonL events.





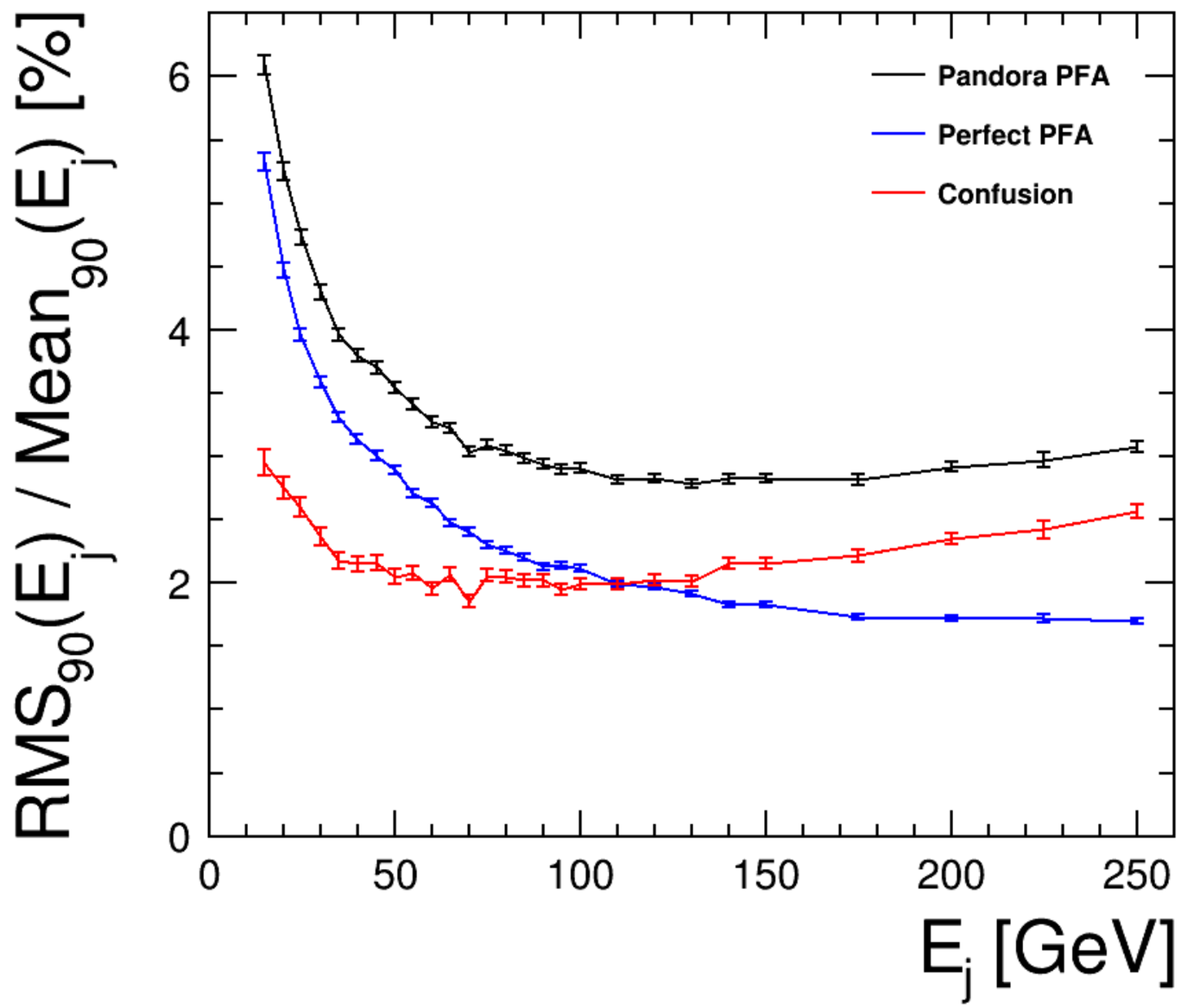
The PandoraAnalysis toolkit has scripts designed for setting the digitisation and calibration constants. The user has to provide samples of kaon0L, γ and μ^-

These scripts make semi-automated calibration possible. Procedure is fully documented.

Pandora JER Studies

- With adoption of a clear calibration scheme, able to investigate jet energy resolutions for a range of digitisation and detector configurations, and perform detector optimisation studies.
- Studies performed very carefully and fully described in **S. Green's talk:** Sim/Reco, 13:30, Wed.

S. Green

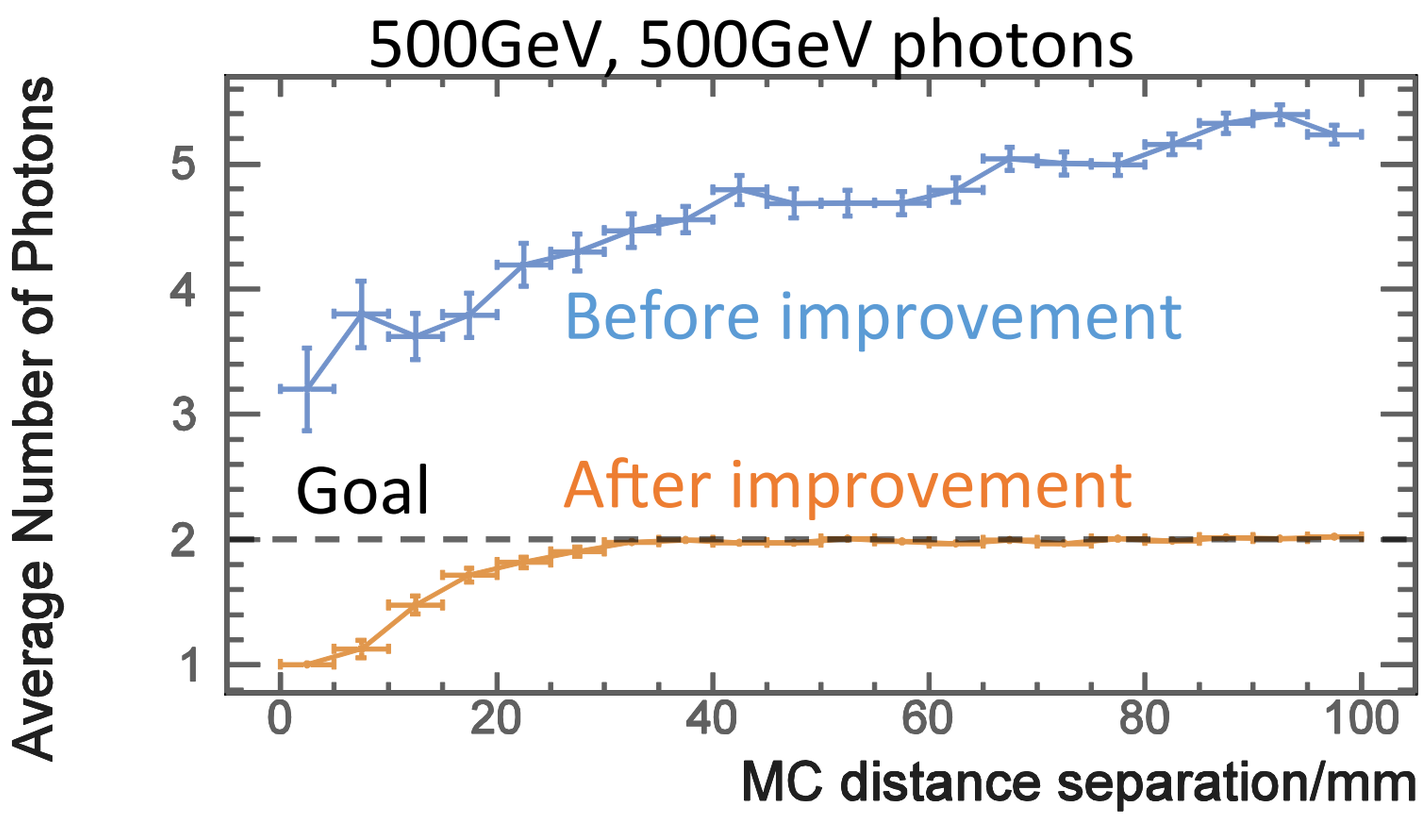


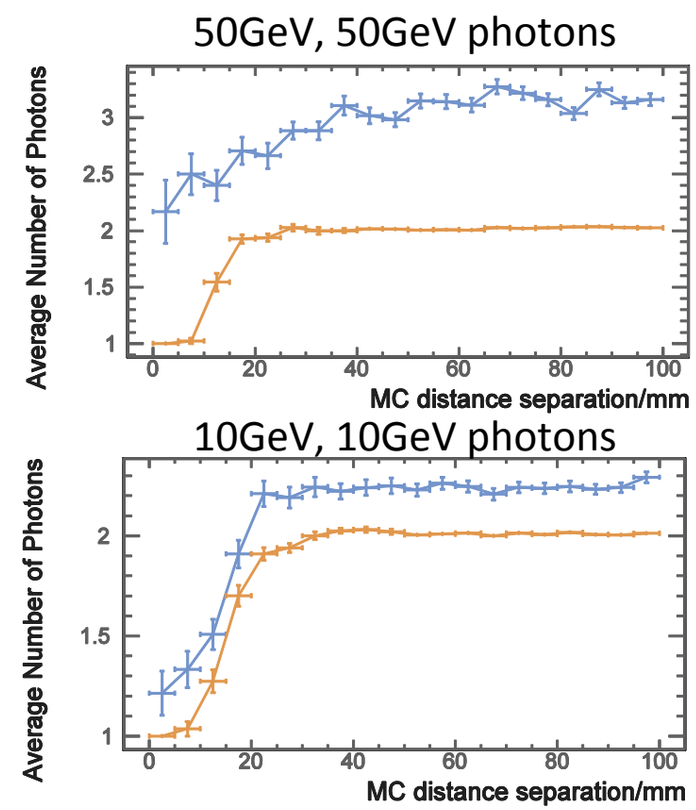
- **iLCSoft_v01_17_07**
  - **PandoraPFA v02-00-00**
- **Digitisation: ILDCaloDigi**
  - **Realistic ECal and HCal options**
  - **100 ns ECAL and HCAL Timing Cuts**
- **1 GeV HCAL cell E_{HAD} truncation**
- **PandoraAnalysis Calibration Tools**

Pandora Photon Reconstruction

- Improve completeness of reconstructed photons, particularly at high energies, where small fragments of EM showers could often be reconstructed as separate particles.
- Two new Pandora algorithms carefully compare candidate photon clusters, collecting evidence of association, based on cluster separation and energy profiles.
- Performance plots below show average number of reconstructed photons (as a function of true separation) for samples consisting of two photons, generated with random directions.

B. Xu



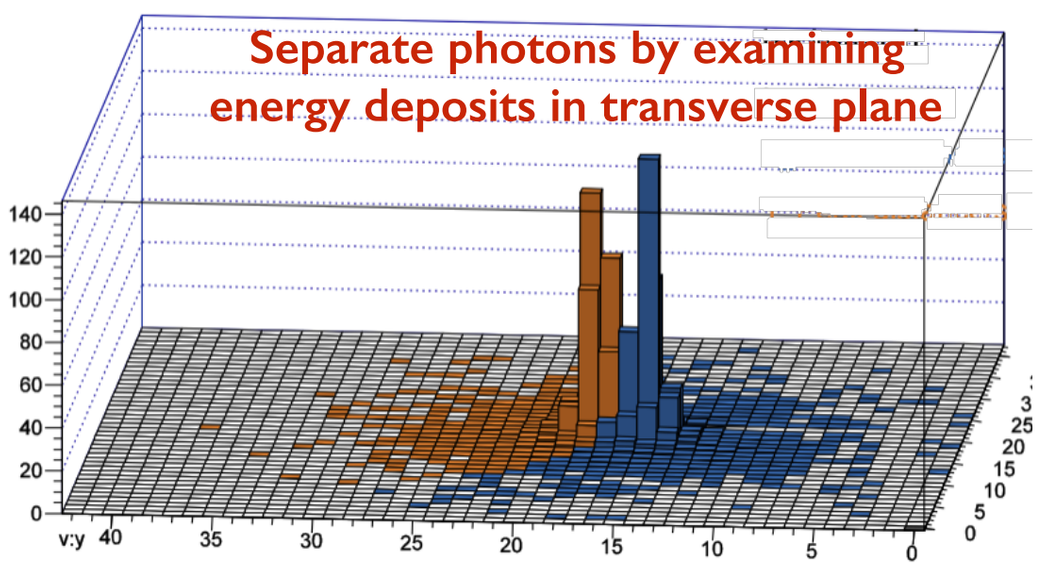


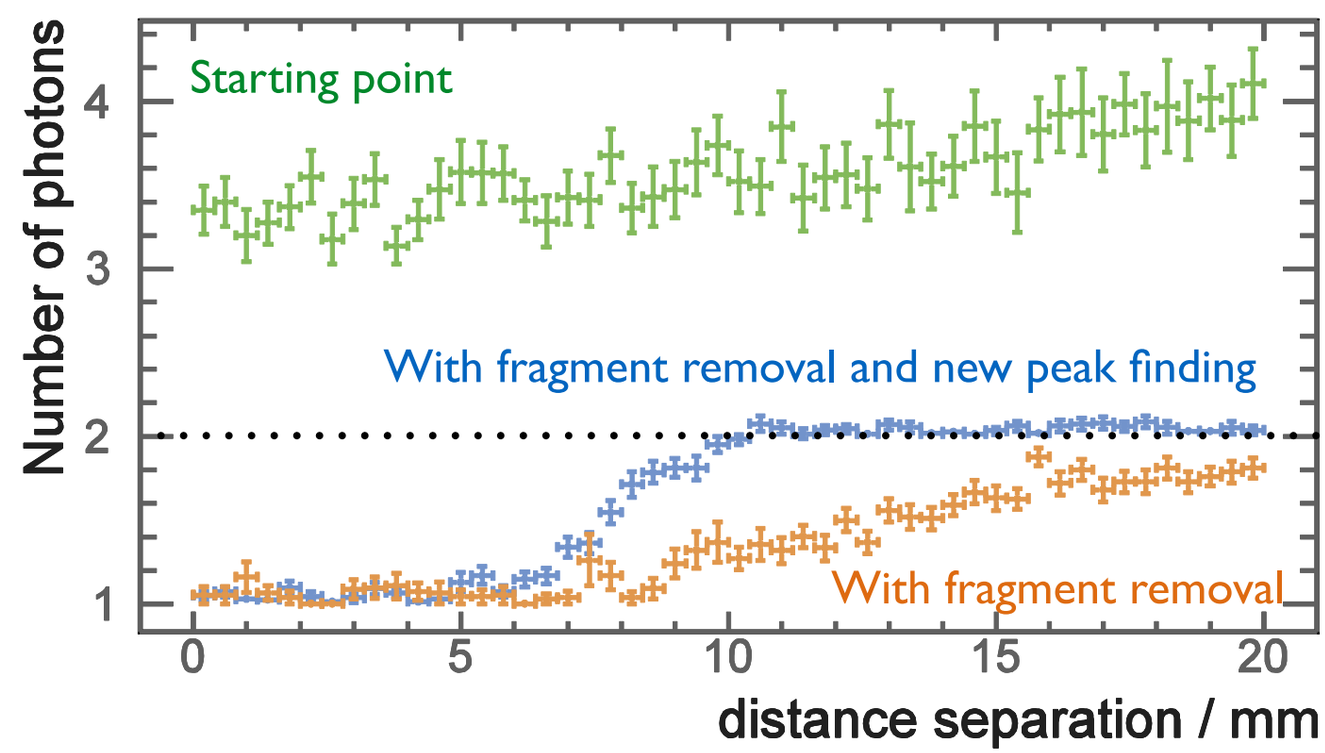
Pandora Photon Reconstruction

- Improvements to standalone reconstruction algorithms, which run at start of Pandora LC reconstruction, aim to carefully separate nearby electromagnetic showers.
- Identify EM shower cores by projecting ECAL energy deposits into transverse plane. Apply algorithm to identify energy deposition peaks and to collect hits contributing to each peak.
- Performed carefully, this approach should improve photon purity and completeness, reduce confusion in jet reconstruction and improve e.g. π^0 reco. **B. Xu talk:** Sim/Reco, 14:00, Wed.

B. Xu

E.g. Separation of nearby high energy (500GeV) photons:





Pandora Summary

- Pandora provides an easy and fast development platform. It has no external dependencies beyond ROOT, which it uses purely for event visualisation and monitoring purposes.
- Visualisation APIs provide simple access to user-customised event displays in algorithms, enabling a rapid and rewarding visual approach to debugging/development.
- Pandora can provide a complete standalone environment for rapid development. Need only run the iLCSoft app once to persist input Hits in Pandora binary or XML formats.
- Pandora is ideally suited for distributed development i.e. people can work on standalone algorithms, which can then be slotted into the reconstruction via simple XML config.

https://www.github.com/PandoraPFA

EPJC.75.439

Pandora Documentation

Pandora Documentation

http://arxiv.org/abs/1506.05348

or EPJC.75.439 or PandoraPFA github page

The Pandora Software Development Kit for Pattern Recognition

J. S. Marshall[a,*], M. A. Thomson[a]

[a] *Cavendish Laboratory, University of Cambridge, Cambridge, United Kingdom*

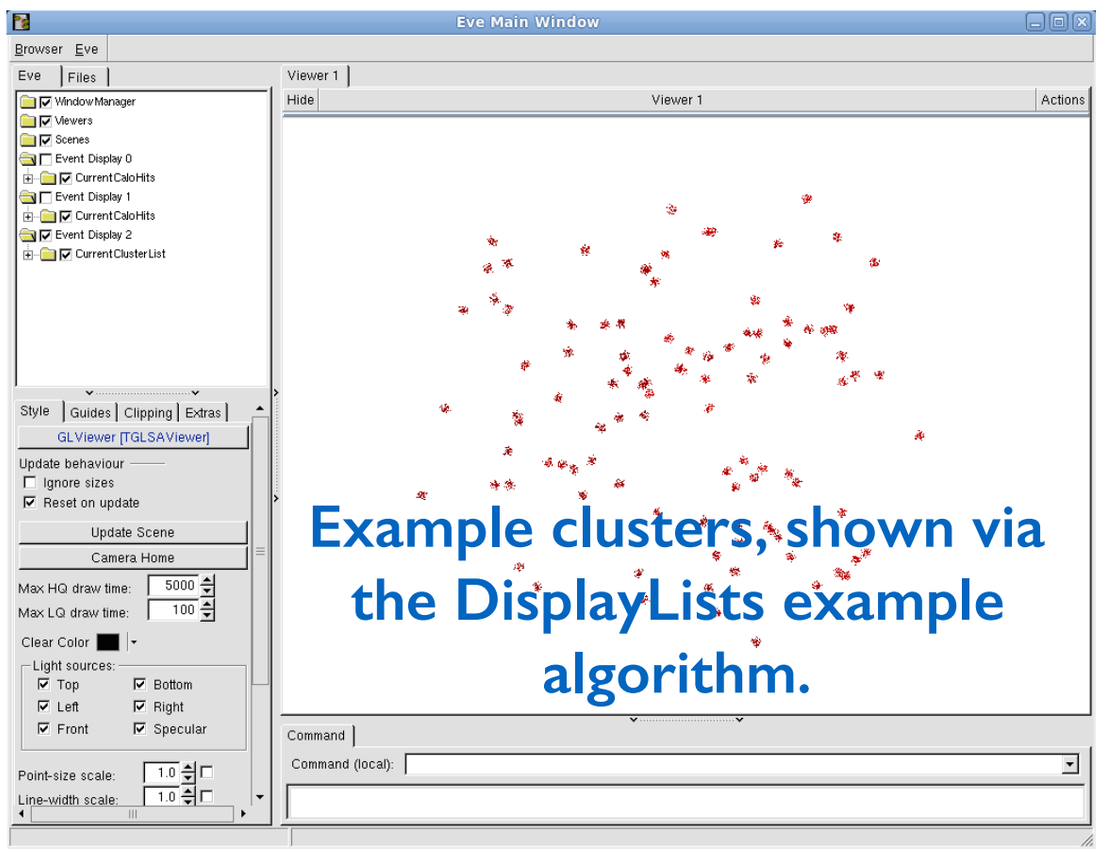
Abstract

The development of automated solutions to pattern recognition problems is important in many areas of scientific research and human endeavour. This paper describes the implementation of the Pandora Software Development Kit, which aids the process of designing, implementing and running pattern recognition algorithms. The Pandora Application Programming Interfaces ensure simple specification of the building-blocks defining a pattern recognition problem. The logic required to solve the problem is implemented in algorithms. The algorithms request operations to create or modify data structures and the operations are performed by the Pandora framework. This design promotes an approach using many decoupled algorithms, each addressing specific topologies. Details of algorithms addressing two pattern recognition problems in High Energy Physics are presented: reconstruction of events at a high-energy e^+e^- linear collider and reconstruction of cosmic ray or neutrino events in a liquid argon time projection chamber.

Keywords: Software Development Kit, Pattern recognition, High Energy Physics

Pandora Learning Library

- Pandora algorithms create and/or modify clusters, vertices and PFOs. Their decisions (the algorithm logic) whether to proceed with operations can be complex and use-case specific.
- The aim of the Pandora ExampleContent library and test application is to demonstrate the key Pandora functionality in a very simple testing and learning environment.
- The ExampleContent library is structured in exactly the same manner as the LCContent and LArContent libraries, currently in use for Linear Collider and LAr TPC reconstruction.

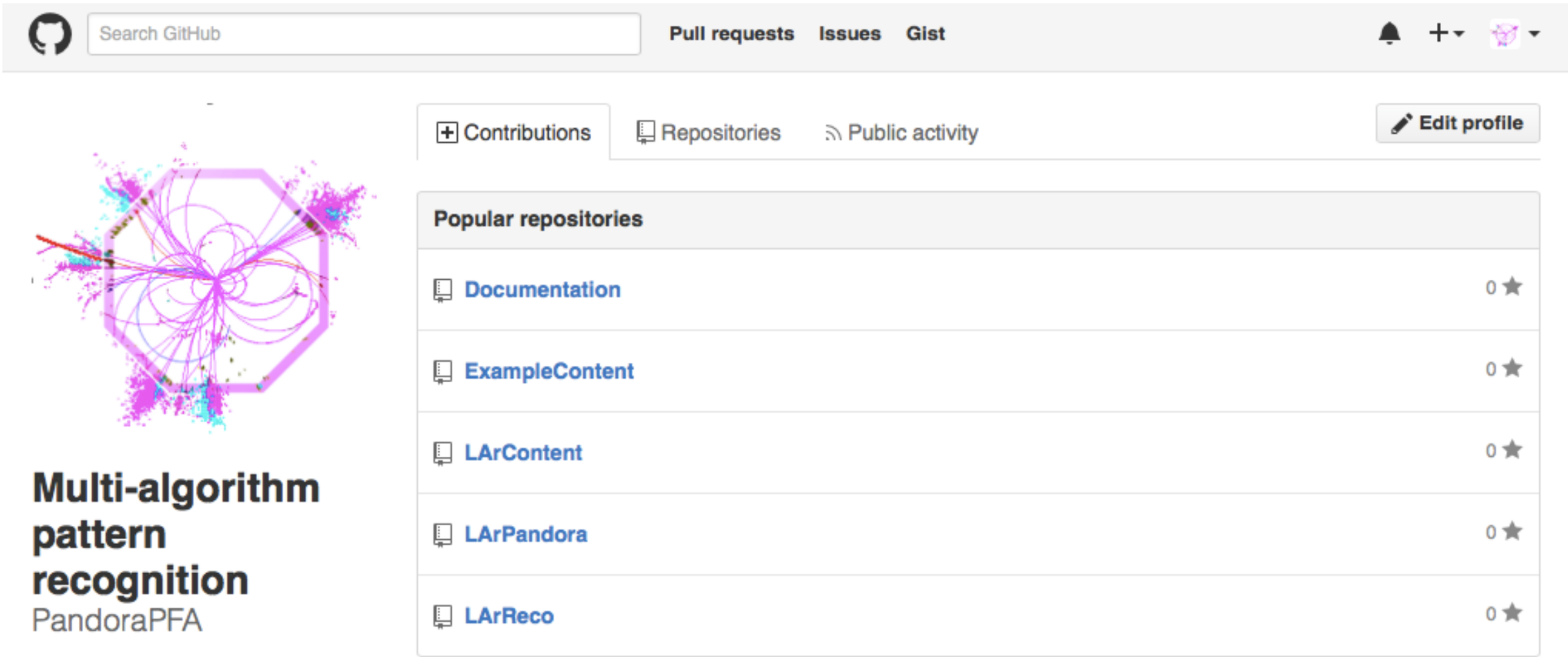


Example clusters, shown via the DisplayLists example algorithm.

- The library consists of example Algorithms, AlgorithmTools, Plugins and Helper functions:
  - Example list access and display
  - Example Cluster, Vertex and PFO creation
  - Cluster manipulation, including merging, deletion, fragmentation and reclustering
  - Creating and saving new lists of objects
  - Using Algorithm Tools and Plugins
  - Writing a tree using PandoraMonitoring.

https://github.com/PandoraPFA/Documentation

Pandora Distribution



Pandora now uses git to provide a distributed version control system.

Distributed approach is perfect for balancing development on multiple Pandora projects.

Source code is officially distributed from the github remote repository:

https://github.com/PandoraPFA

Pandora in iLCSoft

- PandoraPFA v02-03-01: ← Metadata/build package, collects consistent tags
  - Pandora SDK v02-01-00 ← Pandora framework
  - Pandora Monitoring v02-01-00 ← ROOT EVE visualisation, plus tree writing
  - Pandora LCContent v02-01-01 ← Algorithms for use at ILC or CLIC
- MarlinPandora v02-01-00 ← Client App in Marlin framework
- PandoraAnalysis v01-01-00 ← Marlin processors for analysis and calibration

- **It is recommended that you use the DESY reference installations, via cvmfs (or afs).**
- The gcc48 installations include C++11-specific functionality, including fast versions of some key algorithms (see LCContentFast.h), which use KD-trees and new unordered containers.
- Recommended for use if there is significant beam-related background in events (x15-20 speed-up). Just append "Fast" to relevant algorithm types in PandoraSettings XML file.

Pandora Backup Slides

Pandora EDM Object Extensions

Object creation API

Note optional PandoraObjectFactory argument

```
/**
 *  @brief  Object creation helper class
 */
template <typename PARAMETERS, typename OBJECT>
class ObjectCreationHelper
{
public:
    typedef PARAMETERS Parameters;
    typedef OBJECT Object;

    /**
     *  @brief  Create a new object from a user factory
     *
     *  @param  pandora the pandora instance to create the new object
     *  @param  parameters the object parameters
     *  @param  factory the factory that performs the object allocation
     */
    static pandora::StatusCode Create(const pandora::Pandora &pandora, const Parameters &parameters,
        const pandora::ObjectFactory<Parameters, Object> &factory = pandora::PandoraObjectFactory<Parameters, Object>());
};
```

Enhanced object

```
/**
 *  @brief  Example calo hit class - add an additional property to pandora calo hit
 */
class ExampleCaloHit : public pandora::CaloHit
{
public:
    /**
     *  @brief  Constructor
     *
     *  @param  parameters the example calo hit parameters
     */
    ExampleCaloHit(const ExampleCaloHitParameters &parameters);

    /**
     *  @brief  Get the additional property string
     *
     *  @return the additional property string
     */
    const std::string &GetAdditionalProperty() const;

private:
    std::string     m_additionalProperty;       ///< The additional property string
};
```

- Advanced users can add additional functionality to Pandora objects.
- Provide implementation of a custom PandoraObjectFactory.
- Specify non-default ObjectFactory at point of object creation.
- PandoraSDK will work with pointers to base objects.
- Algorithms can access derived object content via dynamic_cast

Pandora Multiple Instances

PANDORA

AlgorithmManager	Algorithms and AlgorithmTools
CaloHitManager MCManager TrackManager	Input object Managers
ClusterManager PfoManager VertexManager	Algorithm object Managers
GeometryManager	SubDetectors and DetectorGaps
PluginManager	Pseudo-Layer, BField, Particle Id
PandoraSettings	Instance config.
PandoraApiImpl PandoraContentApiImpl PandoraImpl	Hidden implementation

Marlin steering

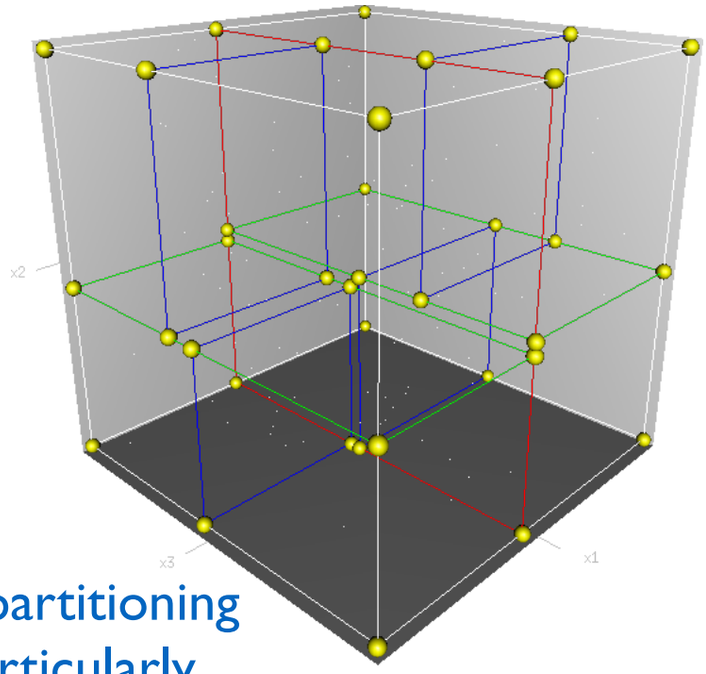
```
<execute>
  <processor name="MyMarlinPandoraMuon"/>
  <processor name="MyPfoAnalysisMuon"/>
  <processor name="MyMarlinPandoraDefault"/>
  <processor name="MyPfoAnalysisDefault"/>
  <processor name="MyMarlinPandoraPerfectPhoton"/>
  <processor name="MyPfoAnalysisPerfectPhoton"/>
  <processor name="MyMarlinPandoraPerfectPhotonNK0L"/>
  <processor name="MyPfoAnalysisPerfectPhotonNK0L"/>
  <processor name="MyMarlinPandoraPerfectPFA"/>
  <processor name="MyPfoAnalysisPerfectPFA"/>
</execute>
```

- Older Pandora releases: configurable static members, so enforce can only be one Pandora instance per system process.
- From Pandora v00-17-00, each Pandora instance stands alone and contains all it needs for a complete reconstruction.
- Pandora instance is a container of Manager instances, API Impl instances and a Settings instance.
- Algorithms and Algorithm Tools created (as requested by XML config) and owned by Algorithm Manager.
- All Pandora Process instances (includes Algorithms and AlgorithmTools) know with which Pandora they are associated.
- Can e.g. run multiple Pandora passes (with different algs) in one Marlin job.

Pandora LCContent FAST

- Recent **successful** application of Pandora LC algorithms to reconstruction of events in CMS HGCAL upgrade geometry.
- Challenging environment with significant pile-up; algorithms need to deal with large numbers of hits and tracks.
- Tested and improved LC algorithm performance using events at CLIC with sizeable $\gamma\gamma \rightarrow$hadrons background.

Example 3-dimensional k-d tree:



k-d trees are space-partitioning data structures, particularly suited to range searches using multidimensional keys.

- **Significant** gains made using **k-d trees** in cone clustering, fragment removal, track-cluster and topological association algs.
- **Typical CLIC pile-up events:**
  from ~60 s / event
  to 12.5 s / event

- k-d tree implementation uses C++11, so LCContent contains FAST versions of algorithms, only built if C++11 supported.
- To use, append "Fast" keyword to alg types in PandoraSettings XML file. Will become default once ilcsoft uses C++11 as standard.

Pandora Arbor Content

Current status of Arbor(-like)PFA in PandoraSDK (R. Ete, IPNL)

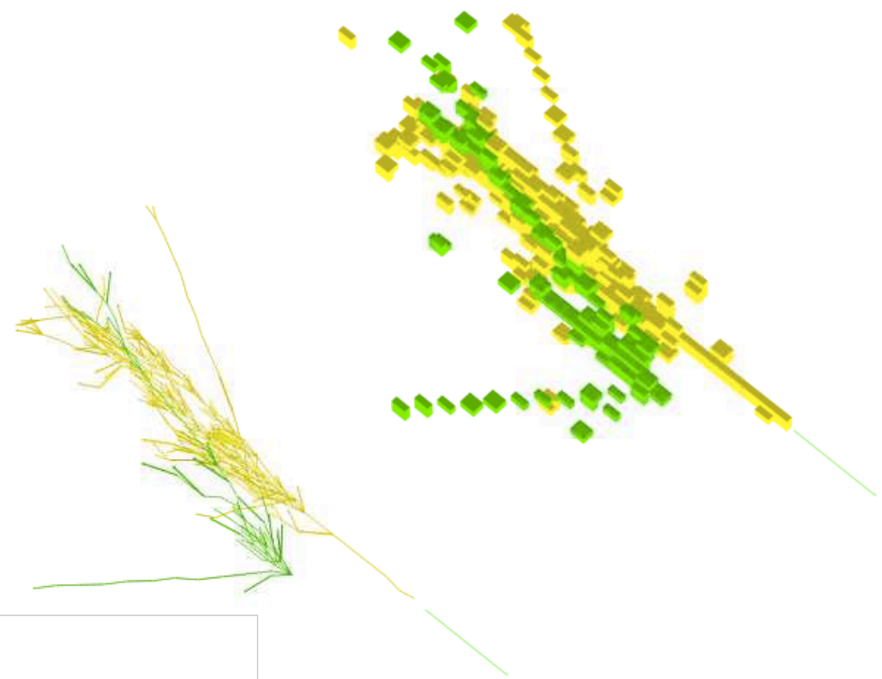
Implementation for the SDHCAL prototype

* Package ArborPFA
→ https://github.com/SDHCAL/ArborPFA

* Developed for SDHCAL test beam data

* Single particle study

* Separation of close-by hadronic showers



Implementation for full detector (i.e ILD)

* Package ArborContent (algorithms) + MarlinArbor (marlin interface)
→ https://github.com/rete/ArborContent
→ https://github.com/rete/MarlinArbor

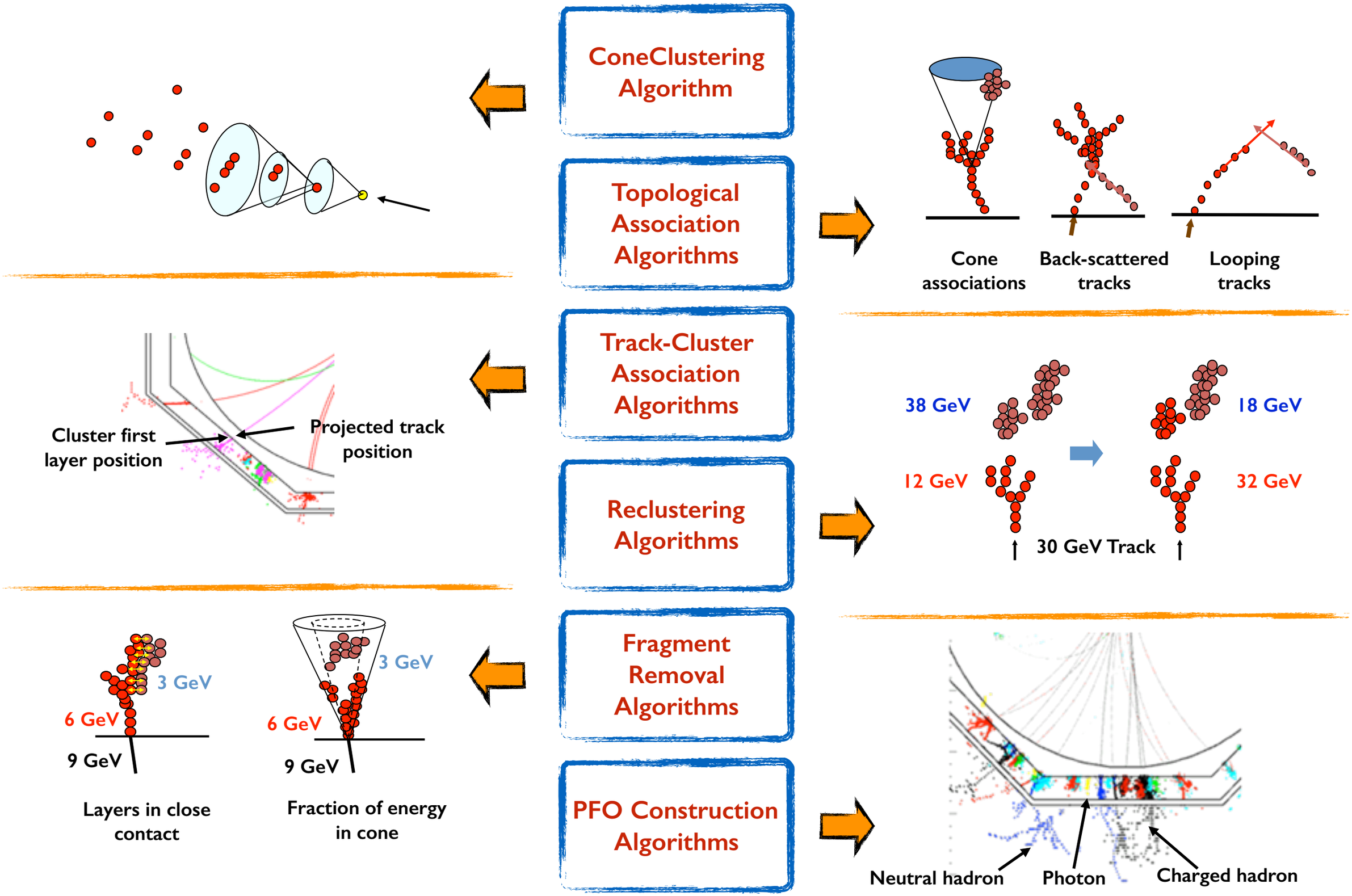
* Uses the new PandoraSDK (v01.00.01) object extension functionality → extension of CaloHit (arbor_content ::CaloHit)

* Algorithms currently being reviewed and new ones added for full detector implementation
→ Ecal specific impl, energy estimators, Ecal-Hcal link, gap handling, reclustering, etc...

* Refactorization of algorithms with AlgorithmTools to test new ideas on key points (specific seeding, cleaning, branching, etc ...)

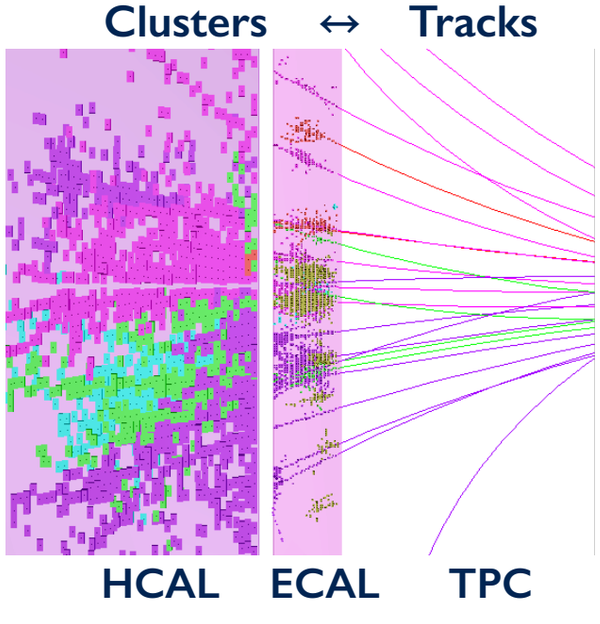
R. Ete

Pandora LC Algorithms

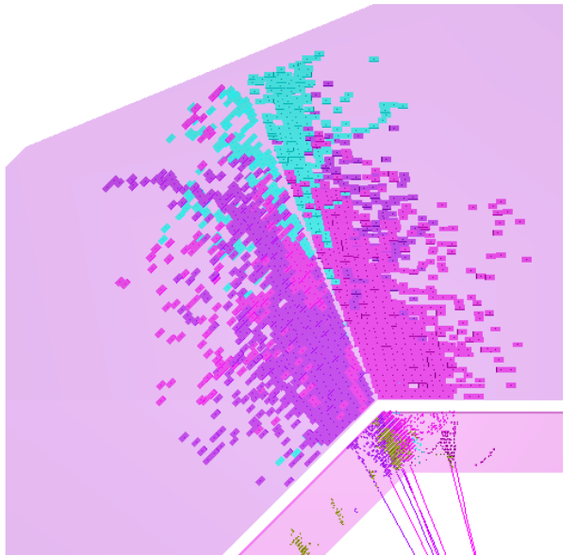


Pandora LC Reclustering

- Key aspect of particle flow calorimetry is association of calorimeter clusters to inner detector tracks.
- Look for consistency between cluster properties and helix-projected track state at front face of calorimeter:
  - Close proximity between cluster and track positions.
  - Consistent track and initial cluster directions.

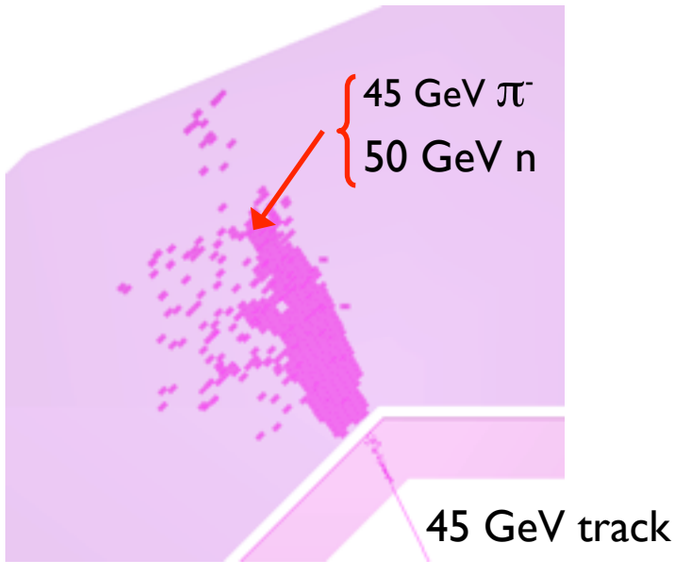


- At some point, in high energy jets, cannot cleanly resolve neutral hadrons in hadronic showers.
- Use information from track-cluster associations to identify pattern-recognition problems:



After topological association

Compare E/p values to find problems

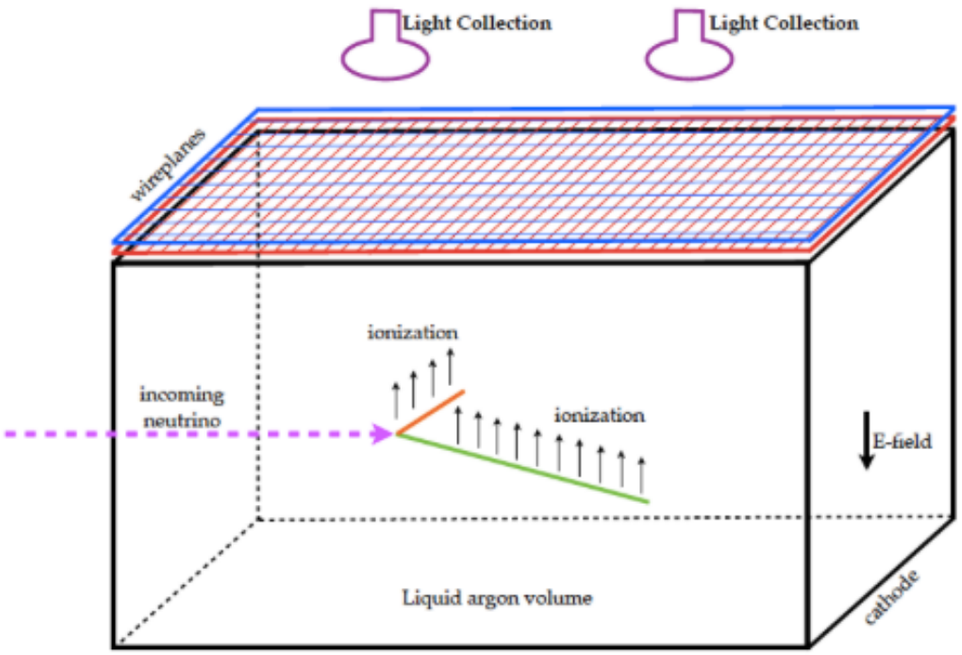


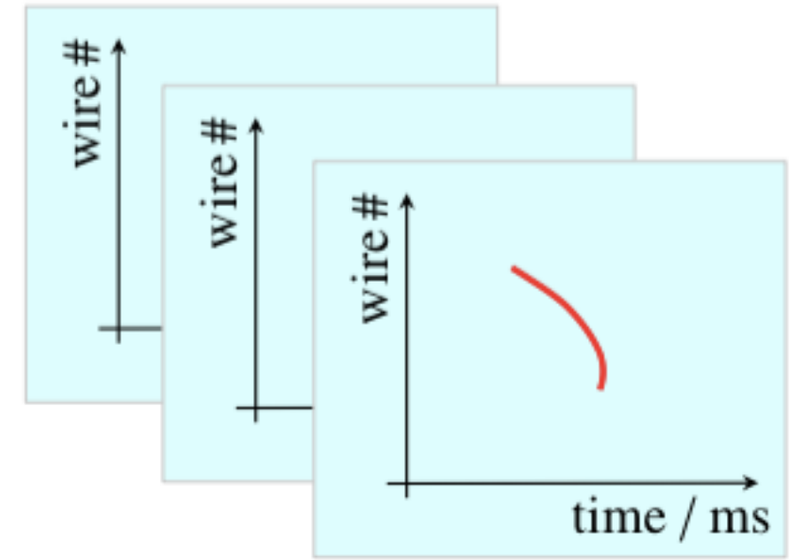
Find n absorbed into π^- cluster

e.g. 45GeV track associated to 95GeV cluster:

identify and address clustering problem

Pandora LAr TPC Reconstruction





Picture from M. Soderberg

- Ionisation electrons detected by series of wire planes, enabling particle tracking and calorimetry.
- Reveal neutrino interactions in unprecedented detail. Obtain 3 "images": wire no. vs drift time.
- **Software challenge:**
  - 3x2D reconstruction, combine results to obtain 3D image of neutrino interaction.
  - Many 'hits', diverse event topologies, 2D views with features often obscured in 1+ view.