Pandora pattern recognition tutorial

24th June 2019 - Fermilab
LArSoft Summer Workshop 2019

Andrew Smith - For the Pandora team
Overview

- The aim of this talk is to answer the following questions:
  - What is pattern recognition and what is Pandora?
  - How does Pandora work?
  - How does Pandora fit into LArSoft?
  - How do I use the outputs of Pandora?
  - How can I learn more?
Pattern recognition in LArTPC experiments
**From images to physics**

- LArTPC detectors produce high resolution images of particle interactions that are rich with information we can exploit.

- To do physics with this data, we need to **reconstruct** these interactions from the raw images.

- One key component of the reconstruction process is **pattern recognition (patrec)**, in which we:
  - Identify the individual particles and their relationships to each other
  - Arrange these particles into hierarchies
  - Determine their 3D trajectories

LArTPCs provide us with:
- mm-scale resolution calorimetric imaging
- Low energy thresholds
- 3D imaging

*A neutrino interaction image from one wire plane in MicroBooNE*
From images to physics

- LArTPC detectors produce high resolution images of particle interactions that are rich with information we can exploit.
- To do physics with this data, we need to **reconstruct** these interactions from the raw images.
- One key component of the reconstruction process is **pattern recognition (patrec)**, in which we:
  - Identify the individual particles and their relationships to each other
  - Arrange these particles into hierarchies
  - Determine their 3D trajectories

- Human brain excels at pattern recognition
- An automated, algorithmic solution is required

LArTPCs provide us with:
  - mm-scale resolution calorimetric imaging
  - Low energy thresholds
  - 3D imaging
The scope of pattern recognition

The reconstruction pipeline

- Low-level image processing
- Hit finding
- Pattern recognition
- Particle fits: Tracks, Showers
- Calorimetric reconstruction
- Particle Identification
The scope of pattern recognition

The reconstruction pipeline

Low-level image processing → Hit finding → Pattern recognition → Particle fits: Tracks, Showers → Calorimetric reconstruction → Particle Identification

Cartoon wire responses to ionization charge drifting past a wire on induction and collection planes

The raw waveforms are processed to deconvolve detector effects and remove noise

A hit is produced for peaks in the processed waveforms. These form the input to the patrec stage
The scope of pattern recognition

The reconstruction pipeline

- Low-level image processing
- Hit finding
- Pattern recognition
- Particle fits: Tracks, Showers
- Calorimetric reconstruction
- Particle Identification

The main job of the patrec is to:

- Cluster the hits together to represent individual particles
- Identify the hierarchical relationship between particles

Cartoon wire responses to ionization charge drifting past a wire on induction and collection planes

Hits from the collection plane for a simulated neutrino interaction in MicroBooNE, before and after patrec

Pandora’s approach
The Pandora project

- General purpose open-source framework for pattern recognition
- Initially used for future linear collider experiments, but now well established on many LArTPC experiments too!

GitHub Repository
github.com/PandoraPFA

Software development kit

μBooNE Algorithms
The Pandora multi-algorithm approach

- Break the problem up into smaller well defined tasks and develop careful, targeted algorithms for each task
  - E.g. Cluster together two hits if …
- Algorithm complexity varies from simple cuts up to more advanced machine learning techniques
- The application runs many algorithms (~100) to gradually build our understanding until a complete picture of the event develops

- Iteration is used to allow 2-way information flow between algorithms
- Iteration provides powerful feedback loops - a technique that Pandora frequently utilizes
The event data model

- We encapsulate our current understanding of the event via the **event data model**
- After the patrec is finished, these are the objects which are available in LArSoft for downstream analysis

---

**Event data model**

- Hits
- 2D Clusters
- Interaction Vertices
- Particles
- Hierarchies
- 3D Trajectories

**Understanding of the event**

- Cluster of hits
- Neutrino interaction vertex
- Parent-Daughter link defines hierarchy
- EM shower
- Each colour represents a different particle

Reconstructed particles are the principal output

The three readout planes give three “views” of the same interaction

A cartoon LArTPC pattern recognition problem
Pandora’s algorithms for neutrino interactions

Final result

Protected track clusters

Candidate shower spines

Candidate shower branches

Neutrino interaction vertex

Simulated unresponsive channels

Vertex finding using SVMs

Matching clusters between views

U plane

V plane

W plane

2D clustering

Electromagnetic showers

Neutrino interaction vertex

Neutrino interaction vertex

MicroBooNE simulation

Pandora’s other algorithm chains

**Neutrino / test-beam chain**

- As described on previous slides, algorithms are designed for neutrino or test-beam interactions
- Identify the primary interaction vertex early in the patrec to inform later algorithms
- Includes special chains of algorithms for electromagnetic showers

Each algorithm chain works well on the types of interactions it’s designed for

For surface detectors, we need a way of dealing with events that contain both neutrino/test-beam interactions and cosmic-rays

**Solution:** “Consolidated approach”

**Cosmic-ray chain**

- Optimised to reconstruct cosmic-ray muons
- More strongly track-oriented than the neutrino / test-beam hypothesis
- Includes algorithms to identify and reconstruct delta-rays of energetic cosmic-rays

A Pandora-reconstructed cosmic-only data event in MicroBooNE
The consolidated approach - 1

Input hits → Cosmic-ray chain

Reconstruct all input hits under the cosmic hypothesis
Identify unambiguous cosmic-rays and remove their hits
The consolidated approach - 3

Input hits → Cosmic-ray chain → Unambiguous cosmic-ray tagging + removal → Remaining hits → Event slicing

Group the remaining hits into topologically associated “slices”

Group based on proximity and pointing
The consolidated approach - 4

- **Neutrino / test beam chain**
  - Reconstruct each slice separately using both algorithm chains
  - Now have an either/or choice

- **Cosmic-ray chain**

**Slice ID**

**Choose** which chain to keep on each slice
- Identify the slice(s) that contains the neutrino / test beam interaction - if any

**The details of this step are experiment specific**
Custom slice ID via external event building module

- In some cases, it’s necessary to make this decision outside of the Pandora framework e.g. to use optical information, cosmic ray tagging system, etc.

- Pandora’s “all outcomes” collection contains the slices as reconstructed under BOTH chains allowing downstream users to make slice-ID decision with full power of LArSoft

- Must be used carefully, as you don’t want to read both outcomes directly into your analysis - each slice is necessarily counted twice!

- “External event building” LArSoft module does the bookwork to safely interface with this collection to produce a new consolidated output. No detailed knowledge of Pandora required

If you are interested in writing an external slice ID, let us know for more technical details
The consolidated output hierarchy

An example output (neutrino experiment)

- **Primary Particles**
  - Neutrino
  - CR muon

- **Daughter Particles**
  - Track-like final state
  - Shower-like final state
  - Shower-like delta ray

- **Daughters of Daughter Particles**
  - Shower-like secondary

**Output from the neutrino chain**

**Output from the cosmic chain**
The consolidated output hierarchy

Primary Particles

The reconstructed neutrino doesn't have any associated hits, but has an interaction vertex position available.

An example output (neutrino experiment)

- **Primary Particles**
  - Neutrino
  - CR muon
  - CR muon
  - Track-like final state
  - Shower-like final state
  - Shower-like delta ray
  - Shower-like secondary

Reconstructed primary cosmic-ray muons
The consolidated output hierarchy

An example output (neutrino experiment)

- Neutrino
- CR muon
- CR muon
  - Track-like final state
  - Shower-like final state
  - Shower-like delta ray

Daughter Particles

The daughters of the neutrino are the reconstructed final state particles.

Reconstructed delta rays are the daughters of CR muons.
The consolidated output hierarchy

An example output (neutrino experiment)

- **Primary Particles**
  - Neutrino
  - CR muon
  - CR muon

- **Daughter Particles**
  - Track-like final state
  - Shower-like final state
  - Shower-like delta ray

- **Daughters of Daughter Particles**
  - Shower-like secondary

Reconstructed particles from secondary interactions are daughters of the final states.
Pandora in LArSoft
Pandora in LArSoft

- Pandora can be used as a standalone piece of software or integrated into a larger framework, such as LArSoft

- The larpandora application, contains the producer module that makes the collections and associations that are stored in the output root files

larpandora
LArSoft repository that contains the application that runs pandora - in charge of inputs and outputs - shared amongst all LArTPC experiments

pandora (external)
External package shared amongst all experiments - contains Pandora's core software development kit - Provides application ↔ algorithm interface

larpandoracontent
LArSoft repository that contains the algorithms themselves - shared amongst all LArTPC experiments
Pandora’s outputs to LArSoft

- PFParticles
  - Tracks
    - Clusters
      - Slices
      - Vertices
      - Metadata
  - SpacePoints
  - Slices
  - Vertices
  - Metadata
- Hits
- PCAxes
- Showers
- T vs. S
Pandora’s outputs to LArSoft

The core output is the **PFParticle** - these are the reconstructed particles identified by Pandora.

Each PFParticle is assigned a PDG code which is Pandora’s best guess at the particle’s type, limited to track-like, shower-like, cosmic-ray or neutrino / test-beam particle - the full PID is done downstream.

A primary particle is one without a parent PFParticle, e.g. primary CR muon, neutrino, test-beam particle.

Each PFParticle has a unique ID which is accessed using the **Self()** method.

The **Parent()** and **Daughters()** methods give the unique ID's of the parent and daughter PFParticles respectively - this is how we navigate the hierarchy.

See [recob::PFParticle documentation](https://example.com) for more details.
Each PFParticle is associated to up to one cluster per view, which allows us to access the hits associated to the particle.

See recob::Cluster documentation for more details.
The trajectories of the PFParticles are stored in the 3D space points, each of which is associated to a hit that was used to create the space point.

See recob::SpacePoint documentation for more details.
Each PFParticle belongs to a Slice as described previously. PFParticles in the same slice will be reconstructed using the same algorithm chain. We can also access the hits in each slice.

See recob::Slice documentation for more details.
Pandora’s outputs to LArSoft

The vertex of a PFParticle is the reconstructed 3D start position. Vertices associated to neutrino PFParticles, represent the neutrino interaction point.

See recob::Vertex documentation for more details.
Pandora’s outputs to LArSoft

Additional information about the PFParticles is stored in metadata, this includes scores calculated by Pandora’s algorithms that might be useful to downstream users.

The metadata available depends on the PFParticle and the configuration used. Examples may include:

- **TrackScore** - likelihood that a given PFParticle represents a track (vs. a shower)
- **NuScore** - likelihood that the slice selected as a neutrino represents a true neutrino

See larpandoraobj::PFParticleMetadata [documentation](#) for more details
Pandora’s outputs to LArSoft

Track and shower production is handled by separate producer modules - not considered part of the core pattern recognition output.

Usually, either a Track or a Shower is produced for each PFParticle depending on Pandora’s track-shower ID - but this decision can be changed downstream.

The tracks & showers form the input to downstream calorimetry and PID modules.

See `recob::Track` documentation or `recob::Shower` documentation for more details.
Pandora’s “event dump” in LArSoft
(see backup for instructions for how to run the event dump)
Dumping a Pandora event to the terminal

Each event starts with a summary block

```
- Event ---------------------------------
run: 1 subRun: 1674 event: 16740 pandora
```

- N PFParticles : 69
- N SpacePoints : 10285
- N Clusters     : 135
- N Vertices     : 67
- N Metadata     : 69
- N Tracks       : 20
- N Showers      : 46
- N PCAxes       : 46

All PFParticles are listed, and arranged according to the hierarchy

Can see associations of PFParticles to other collections (e.g. Tracks/Showers)

Metadata is available to learn more about the PFParticles
(This one was tagged as an unambiguous cosmic ray, so never was considered as the neutrino / beam particle)
Dumping a Pandora event to the terminal

---

**PFParticle**

- Key 1
- Id 1
- Primary
- PDG 12

---

- # SpacePoint 0
- # Cluster 0
- # Vertex 1
- # Track 0
- # Shower 0
- # PCAxis 0
- # Metadata 1
  -- Property IsNeutrino, value 1
  -- Property NuScore, value 0.934322
  -- Property SliceIndex, value 1

---

- # Daughters 1

---

**PFParticle**

- Key 0
- Id 0
- Parent 1
- PDG 11

---

- # SpacePoint 715
- # Cluster 3
- # Vertex 1
- # Track 0
- # Shower 1
- # PCAxis 1
- # Metadata 1
  -- Property TrackScore, value 0.00116577

---

- # Daughters 0

The reconstructed neutrino has IsNeutrino = 1, and an electron neutrino PDG code (12)
The NuScore is the output of the internal slice-ID for the slice, which in this case had index 1 (this is a MicroBooNE event, different experiments use different techniques)
The reconstructed final state particles are the daughter of the neutrino
In this case there is a single final state electron

Indent ⇒ daughter (print for each daughter)
How do I use Pandora’s consolidated output?
The **LArPandoraHelper** class has many useful functions to help you use Pandora’s outputs, E.g.

```cpp
/**
 * @brief Collect the reconstructed PFParticles from the ART event record
 * @param evt the ART event record
 * @param label the label for the PFParticle list in the event
 * @param particleVector the output vector of PFParticle objects
 */
static void CollectPFParticles(const art::Event &evt,
                               const std::string &label,
                               PFParticleVector &particleVector);

/**
 * @brief Determine whether a particle has been reconstructed as a neutrino
 * @param particle the input particle
 * @return true/false
 */
static bool IsNeutrino(const art::Ptr<recob::PFParticle> particle);
```
Typical task: Getting neutrino identified PFParticles

For more example code please see ConsolidatedPFParticleAnalysisTemplate_module.cc and the LArPandoraHelper class which has many useful functions!

// Get the PFParticle collection from the event record
PFParticleVector pfParticles;
LArPandoraHelper::CollectPFParticles(event, pfParticleLabel, pfParticles);

// Find the PFParticles that have been identified as neutrinos by the slice ID
PFParticleVector neutrinos;
for (const auto &particle : pfParticles)
{
    // Query the PFParticle’s PDG code using a helper function to see if it’s been identified
    // as a neutrino by the slice ID - if so, then add it to our vector of neutrinos
    if (LArPandoraHelper::IsNeutrino(particle))
        neutrinos.push_back(particle);
}
Typical task: Getting neutrino final state PFParticles

For more example code please see ConsolidatedPFParticleAnalysisTemplate_module.cc and the LArPandoraHelper class which has many useful functions!

```cpp
// Make a map from PFParticle.Self() -> PFParticle object for navigation of the hierarchy
PFParticleMap pfParticleMap;
LArPandoraHelper::BuildPFParticleMap(pfParticles, pfParticleMap);

// Find the daughter PFParticles of each primary neutrino PFParticle
for (const auto &neutrino : neutrinos)
{
  for (const auto &daughterId : neutrino->Daughters())
  {
    const auto daughter = pfParticleMap.at(daughterId);

    // Do something with the daughter particle! E.g. find associated tracks / showers
  }
}
Where can I find more information?
Papers and documentation

- The **Pandora SDK paper**
  - Details the design of the software development kit and how algorithms interface with the application that is running Pandora (e.g. larpandora)

- The **Pandora MicroBooNE paper**
  - Gives details of Pandora’s algorithms in MicroBooNE at the time of publication, but generally applicable to other LArTPC experiments too

- All Pandora code is self-documented using doxygen and is available on github
  - [https://github.com/PandoraPFA](https://github.com/PandoraPFA)
Recent workshops & hands-on exercises

- Multi-day Pandora workshop in Cambridge, UK - 2016
  - Talks about how the algorithms work and step-by-step exercises about how you might develop a new algorithm using Pandora!

- LArSoft workshop in Manchester, UK - 2018
- Workshop on advanced computing & machine learning, Paraguay - 2018
  - Talks and exercises about running and using Pandora within LArSoft, including tutorials on using Pandora’s custom event display

- Experiment specific resources:
  - ProtoDUNE analysis workshop, CERN - 2019
  - MicroBooNE Pandora workshop, Fermilab - 2018
Pattern recognition is an important step in the reconstruction of LArTPC events.

Pandora is a solution to the patrec problem that’s widely used by LArTPC experiments.

Pandora uses a multi-algorithm approach to gradually build up our understanding of the event.

Pandora’s consolidated algorithm flow allows us to deal with neutrino / test-beam interactions in dense cosmic-ray environments.

Pandora can be run as part of LArSoft so its outputs are available for use in your own code.

The core outputs are PFParticles and their hierarchical relationships.

There are a number of good resources if you want to learn more about Pandora or get started with some hands on exercises, but don’t be afraid to get in touch with a member of the team!
Pandora team for LArTPC reconstruction

Pandora is an open project and new contributors would be extremely welcome. We’d love to hear from you and we will always try to answer your questions.

<table>
<thead>
<tr>
<th>Pandora SDK development</th>
<th>John Marshall</th>
<th><a href="mailto:john.marshall@warwick.ac.uk">john.marshall@warwick.ac.uk</a></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mark Thomson</td>
<td><a href="mailto:thomson@hep.phy.cam.ac.uk">thomson@hep.phy.cam.ac.uk</a></td>
</tr>
<tr>
<td>LArTPC algorithm development</td>
<td>John Marshall</td>
<td><a href="mailto:a.blake@lancaster.ac.uk">a.blake@lancaster.ac.uk</a></td>
</tr>
<tr>
<td></td>
<td>Andy Blake</td>
<td></td>
</tr>
<tr>
<td>MicroBooNE integration</td>
<td>Andy Smith</td>
<td><a href="mailto:asmith@hep.phy.cam.ac.uk">asmith@hep.phy.cam.ac.uk</a></td>
</tr>
<tr>
<td>ProtoDUNE integration</td>
<td>Steven Green</td>
<td><a href="mailto:sg568@hep.phy.cam.ac.uk">sg568@hep.phy.cam.ac.uk</a></td>
</tr>
<tr>
<td>DUNE FD integration</td>
<td>Lorena Escudero</td>
<td><a href="mailto:escudero@hep.phy.cam.ac.uk">escudero@hep.phy.cam.ac.uk</a></td>
</tr>
<tr>
<td>Graduate students</td>
<td>Joris Jan de Vries, Jack Anthony, Andy Smith</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Stefano Vergani</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Jhanzeb Ahmed, Mousam Rai, Ryan Cross</td>
<td></td>
</tr>
</tbody>
</table>

github.com/PandoraPFA

PandoraPFA.slack.com
Backup
BEGIN_PROLOG
pandora_event_dump:
{
    module_type: "LArPandoraEventDump"
}
dump: @local::pandora_event_dump
dump.PandoraLabel: "pandora"
dump.TrackLabel: "pandoraTrack"
dump.ShowerLabel: "pandoraShower"
dump.VerbosityLevel: "detailed"
END_PROLOG

#include "services_dune.fcl"
services:
{
    scheduler: { defaultExceptions: false }
    RandomNumberGenerator: {} #ART native random number generator
    FileCatalogMetadata: @local::art_file_catalog_mc
}
process_name: LArPandoraEventDump
source:
{
    module_type: RootInput
}
physics:
{
    analyzers:
    {
        dump: @local::dump
    }
    stream1: [ dump ]
    end_paths: [ stream1 ]
}
outputs: {}

Module itself lives here
These labels will depend on which experiment you are working on
Choose between “brief”, “summary”, “detailed” or “extreme”
Choose the services for your experiment

Run using standard lar command:
>> lar -c <my_fhicl_file_name> -s <my_root_file> -n <n_events>