DEEP UNDERGROUND
NEUTRINO EXPERIMENT

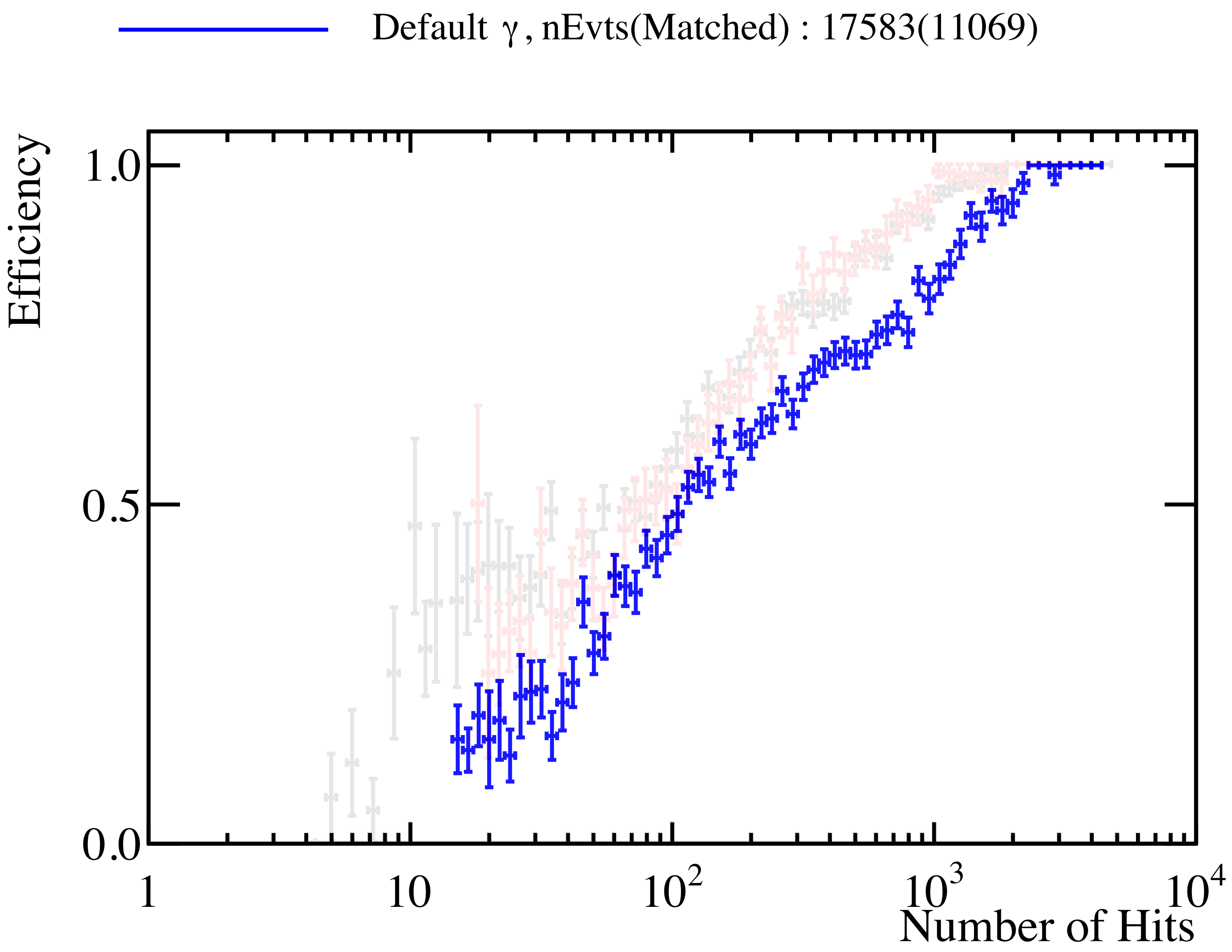
Running Pandora Reconstruction Metrics

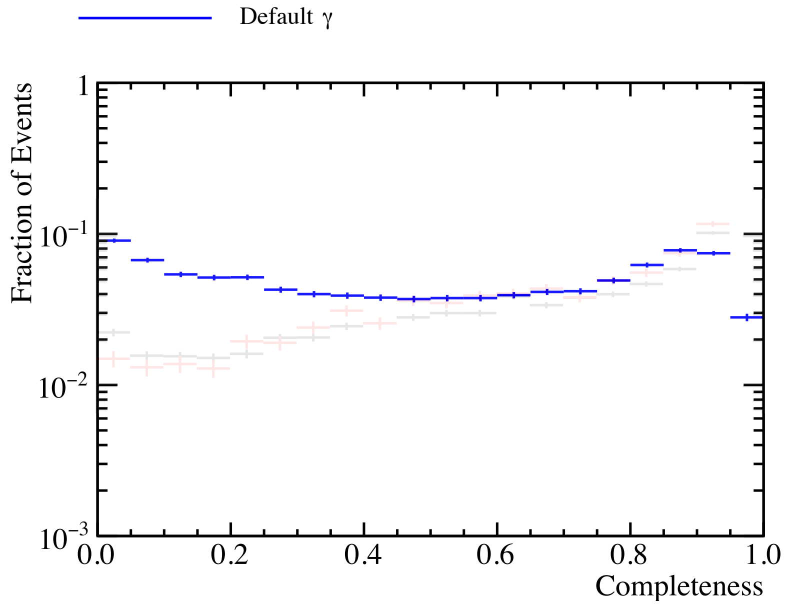
Steven Green

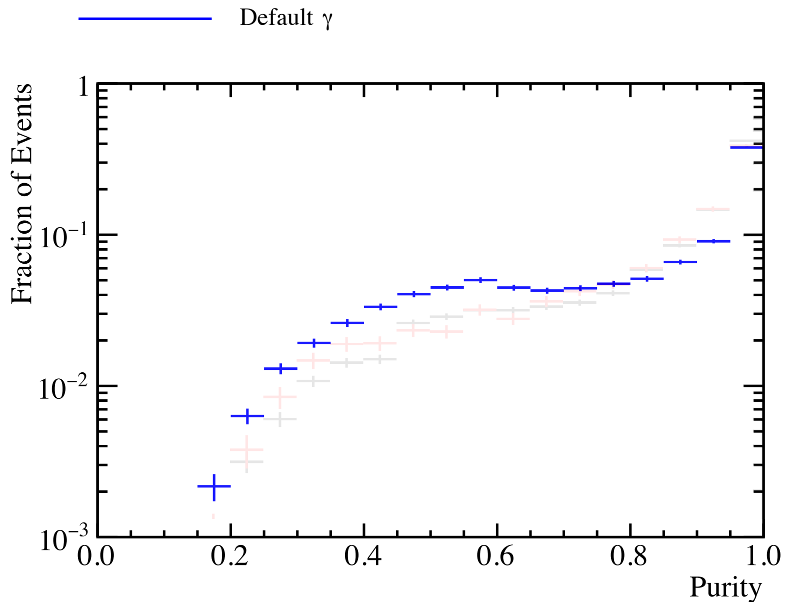
7th October 2019

Introduction

The aim of these slides it to demonstrate how to run the Pandora reconstruction and extract the metrics such as correct event fraction, reconstruction efficiency, purities, completenesses…







Pandora Settings Master ProtoDUNE

```
<pandora>
    <!-- GLOBAL SETTINGS -->
    <IsMonitoringEnabled>true</IsMonitoringEnabled>
    <ShouldDisplayAlgorithmInfo>false</ShouldDisplayAlgorithmInfo>
    <SingleHitTypeClusteringMode>true</SingleHitTypeClusteringMode>

    <!-- ALGORITHM SETTINGS -->
    <algorithm type = "LArEventReading"/>
    <algorithm type = "LArPreProcessing">
        …
    </algorithm>
    <algorithm type = "LArVisualMonitoring">
        …
    </algorithm>

    <algorithm type = "LArMaster">
        <CRSettingsFile>PandoraSettings_Cosmic_Standard.xml</CRSettingsFile>
        <NuSettingsFile>PandoraSettings_TestBeam_ProtoDUNE.xml</NuSettingsFile>
        <SlicingSettingsFile>PandoraSettings_Slicing_ProtoDUNE.xml</SlicingSettingsFile>
        …
    </algorithm>

    <algorithm type = "LArTestBeamEventValidation">
        <CaloHitListName>CaloHitList2D</CaloHitListName>
        <MCParticleListName>Input</MCParticleListName>
        <PfoListName>RecreatedPfos</PfoListName>
        <PrintAllToScreen>false</PrintAllToScreen>
        <PrintMatchingToScreen>true</PrintMatchingToScreen>
        <WriteToTree>false</WriteToTree>
        <OutputTree>Validation</OutputTree>
        <OutputFile>Validation.root</OutputFile>
    </algorithm>

    <algorithm type = "LArVisualMonitoring">
        …
    </algorithm>
</pandora>
```

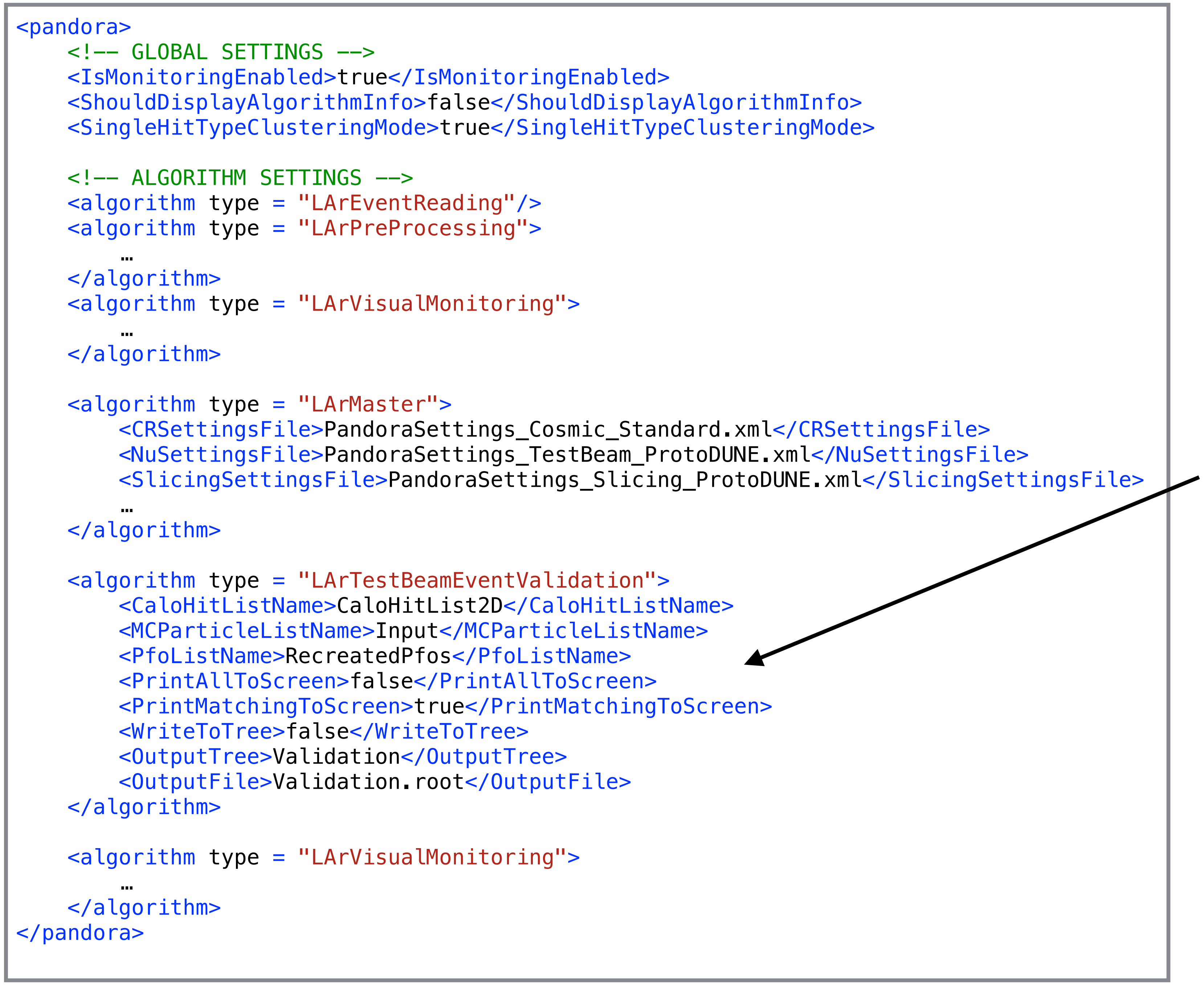
Event reading algorithm is needed when running outside LArSoft, input is pndr/xml.

Lots of visual monitoring.

Note: event displays will only load if IsMonitoringEnabled in global settings is true.

Master algorithm. This creates the multiple Pandora instances required for the consolidated reconstruction. Algorithm chains PandoraCosmic, PandoraTestBeam and Slicing settings files are specified here.

Pandora Settings Master ProtoDUNE

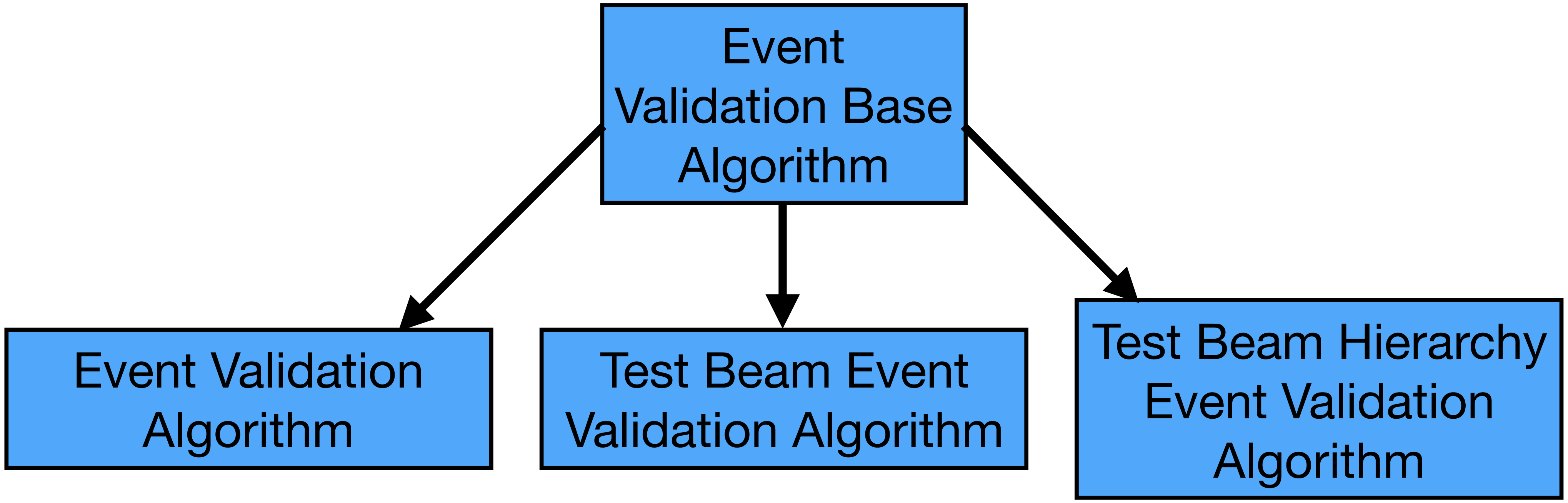


```
<pandora>
    <!-- GLOBAL SETTINGS -->
    <IsMonitoringEnabled>true</IsMonitoringEnabled>
    <ShouldDisplayAlgorithmInfo>false</ShouldDisplayAlgorithmInfo>
    <SingleHitTypeClusteringMode>true</SingleHitTypeClusteringMode>

    <!-- ALGORITHM SETTINGS -->
    <algorithm type = "LArEventReading"/>
    <algorithm type = "LArPreProcessing">
        …
    </algorithm>
    <algorithm type = "LArVisualMonitoring">
        …
    </algorithm>

    <algorithm type = "LArMaster">
        <CRSettingsFile>PandoraSettings_Cosmic_Standard.xml</CRSettingsFile>
        <NuSettingsFile>PandoraSettings_TestBeam_ProtoDUNE.xml</NuSettingsFile>
        <SlicingSettingsFile>PandoraSettings_Slicing_ProtoDUNE.xml</SlicingSettingsFile>
        …
    </algorithm>

    <algorithm type = "LArTestBeamEventValidation">
        <CaloHitListName>CaloHitList2D</CaloHitListName>
        <MCParticleListName>Input</MCParticleListName>
        <PfoListName>RecreatedPfos</PfoListName>
        <PrintAllToScreen>false</PrintAllToScreen>
        <PrintMatchingToScreen>true</PrintMatchingToScreen>
        <WriteToTree>false</WriteToTree>
        <OutputTree>Validation</OutputTree>
        <OutputFile>Validation.root</OutputFile>
    </algorithm>

    <algorithm type = "LArVisualMonitoring">
        …
    </algorithm>
</pandora>
```

Event Validation Algorithm. This algorithm creates the output root files, Validation.root, used for creating the metrics. As input it takes the reconstructed particles, (*RecreatedPfos*), MCParticles, (*Input*) and the 2D input hits (*CaloHitList2D*). There are also options regarding tree writing (*WriteToTree*) and printing to screen (*PrintAllToScreen*, *PrintMatchingToScreen*).

Event Validation Algorithm Structure

All algorithms related to event validation are found in the LArMonitoring directory of LArContent: https://github.com/PandoraPFA/LArContent/tree/master/larpandoracontent/LArMonitoring


Event Validation Base Algorithm
Event Validation Algorithm
Test Beam Event Validation Algorithm
Test Beam Hierarchy Event Validation Algorithm

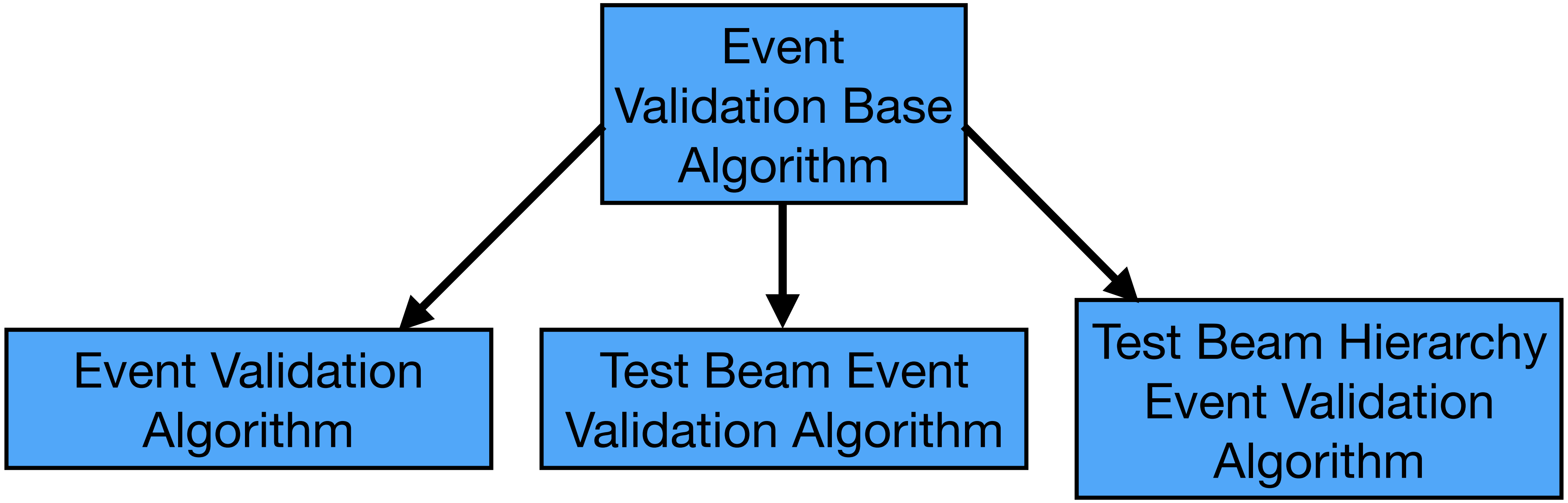
Generically speaking, each of these algorithms works by defining target MC Particles in an event and creating maps of MC <-> Reconstructed Particles.

Interpretation of those matches is then applied such that particles have unique matches only.

The base algorithm contains all common functions for making the best matches, while the derived classes handle the definition of the target MCParticles, the loading of the reconstructed particles, the interpretation of the matches and the filling output root tree.

Event Validation Algorithm Structure

All algorithms related to event validation are found in the LArMonitoring directory of LArContent: https://github.com/PandoraPFA/LArContent/tree/master/larpandoracontent/LArMonitoring



Event Validation Base Algorithm

Event Validation Algorithm

Test Beam Event Validation Algorithm

Test Beam Hierarchy Event Validation Algorithm

Target MC Particles:

- Interaction products from a neutrino interaction.

Target MC Particles:

- Parent test beam particles.

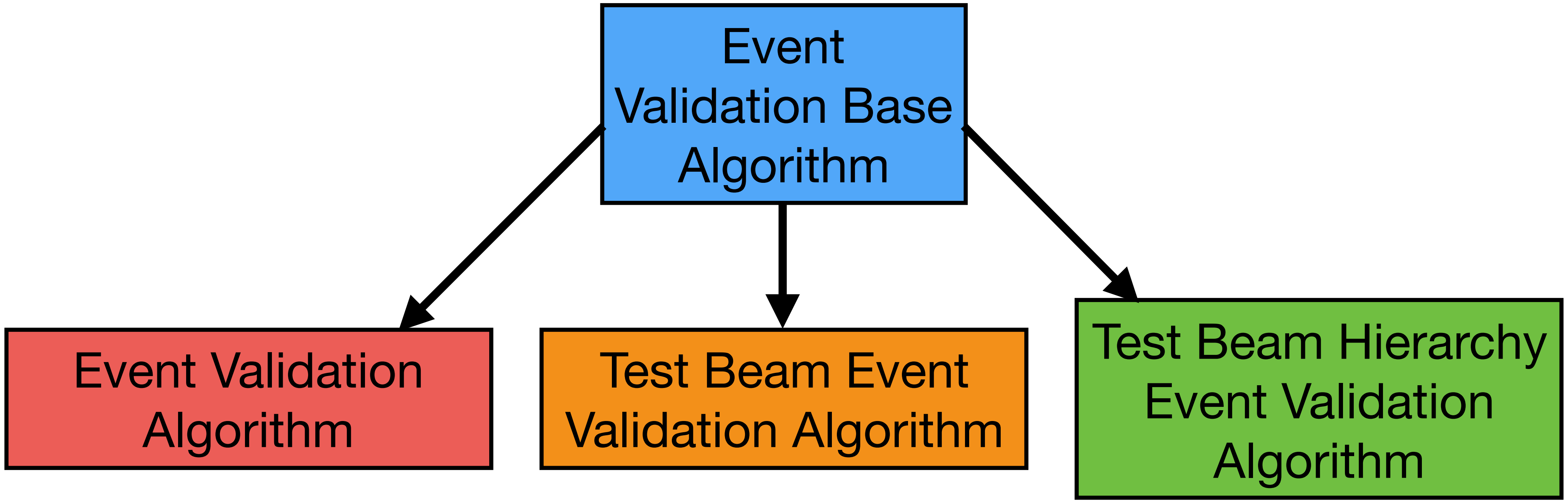
Target MC Particles:

- Parent test beam particles and any daughters from the primary interaction.

Anything lower in the MC hierarchy than the target particle is folded back to the relevant target. So using the Test Beam Event Validation Algorithm a $\pi^+ \rightarrow p\pi^+$ would only be treated as a π^+ because the daughters are folded back, but in the Test Beam Hierarchy Event Validation Algorithm all three particles would be considered.

Event Validation Algorithm Structure

All algorithms related to event validation are found in the LArMonitoring directory of LArContent: https://github.com/PandoraPFA/LArContent/tree/master/larpandoracontent/LArMonitoring



Event Validation Base Algorithm

Event Validation Algorithm

Test Beam Event Validation Algorithm

Test Beam Hierarchy Event Validation Algorithm

Similarly, only the relevant reconstructed particles are considered by these algorithms:

- Daughters of the reconstructed neutrino.
- Test beam particles (with daughters and beyond folder back).
- Test beam particles parents and daughters (with granddaughters and beyond folded back to the relevant daughter particle).

Event Validation Algorithm Root Trees

The output root trees are filled on a particle hierarchy basis for neutrino interactions, test beam interactions and cosmic rays.

The output root trees contain a mixture of single values (ints, floats) as well as vector quantities (vectors of ints and floats).

The single values relate to properties of a full reconstructed hierarchy e.g. vertex (primary) position, while the vector values describe the individual matches made within the hierarchy.

For example the mcPrimaryPdg vector for $\pi^+ \rightarrow p\pi^+$ would contain 211 for the Test Beam Event Validation Algorithm (parent only), but 211, 2112 and 211 for the Test Beam Hierarchy Event Validation Algorithm (parent and daughters).

The index is shared between all vector quantities so mcPrimaryPdg.at(**1**) would be the proton and it would have this energy mcPrimaryE.at(**1**).

Event Validation Algorithm Root Trees

Test Beam Event Validation Algorithm:

fileIdentifier = 0	File Identifier
eventNumber = 0	File Identifier
mcNuanceCode = 2001	MC Particle is triggered test beam (2001), other beam particle (2000) or cosmic ray (3000)
isBeamParticle = 1	
isCosmicRay = 0	
nTargetPrimaries = 1	
targetVertexX = -30.7579	MC Primary Interaction Vertex
targetVertexY = 419.584	
targetVertexZ = -0.49375	
recoVertexX = -29.3559	Reconstructed Vertex
recoVertexY = 419.106	
recoVertexZ = 29.7876	
mcPrimaryId = (vector<int>*)0x7f90117b8be0	Unique MC Target ID
mcPrimaryPdg = (vector<int>*)0x7f90117bbb50	MC Target PDG Code
mcPrimaryE = (vector<float>*)0x7f9011755540	MC Target Energy
mcPrimaryPX = (vector<float>*)0x7f900ec56d20	MC Target Momentum
mcPrimaryPY = (vector<float>*)0x7f9011789750	
mcPrimaryPZ = (vector<float>*)0x7f90116de890	
mcPrimaryVtxX = (vector<float>*)0x7f901178cc90	MC Target Vertex (Start)
mcPrimaryVtxY = (vector<float>*)0x7f9011754f40	
mcPrimaryVtxZ = (vector<float>*)0x7f9011755d10	
mcPrimaryEndX = (vector<float>*)0x7f90117617a0	MC Target Vertex (End)
mcPrimaryEndY = (vector<float>*)0x7f90117cb4f0	
mcPrimaryEndZ = (vector<float>*)0x7f90117cb910	
mcPrimaryNHitsTotal = (vector<int>*)0x7f90117cbdb0	MC Target Hit Numbers
mcPrimaryNHitsU = (vector<int>*)0x7f90117cc220	
mcPrimaryNHitsV = (vector<int>*)0x7f900ee4d660	
mcPrimaryNHitsW = (vector<int>*)0x7f900ee52800	

Event Validation Algorithm Root Trees

Test Beam Event Validation Algorithm:

nPrimaryMatchedPfos = (vector<int>*)0x7f900ee52f00	Number of good matches for this MC particle to Reco particles.
nPrimaryMatchedTBPfos = (vector<int>*)0x7f900ee533a0	Number of good matches for this MC particle to Reco test beam particles.
nPrimaryMatchedCRPfos = (vector<int>*)0x7f900ee53860	Number of good matches for this MC particle to Reco cosmic ray particles.
bestMatchPfoId = (vector<int>*)0x7f900ee53d20	Reconstructed Particle ID
bestMatchPfoPdg = (vector<int>*)0x7f900ee541c0	Reconstructed Particle PDG Code
bestMatchPfoNHitsTotal = (vector<int>*)0x7f900ee54610	Reconstructed Particle Number of Hits (Total Number of Hits)
bestMatchPfoNHitsU = (vector<int>*)0x7f900ee54ae0	
bestMatchPfoNHitsV = (vector<int>*)0x7f900ee54fa0	
bestMatchPfoNHitsW = (vector<int>*)0x7f900ee55460	
bestMatchPfoNSharedHitsTotal = (vector<int>*)0x7f900ee55920	Reconstructed Particle Number of Hits (Shared Number of Hits with the Target MC Particle)
bestMatchPfoNSharedHitsU = (vector<int>*)0x7f900ee55de0	
bestMatchPfoNSharedHitsV = (vector<int>*)0x7f900ee562a0	
bestMatchPfoNSharedHitsW = (vector<int>*)0x7f900ee56760	
nTargetMatches = 1	Number of good matches made between MC and Reco Particles
nTargetTBMatches = 1	Number of good matches made between MC and Reco Test Beam Particles
nTargetCRMatches = 0	Number of good matches made between MC and Reco Cosmic Ray Particles
bestMatchPfoIsTB = (vector<int>*)0x7f900ee575f0	Is the best match Reco particle a test beam particle
interactionType = 158	Enum, primarily used for neutrino interactions.
isCorrectTB = 1	Has test beam MC particle been reconstructed successfully
isCorrectCR = 0	Has test beam cosmic ray particle been reconstructed successfully
isFakeTB = 0	Has this particle been labelled as a test beam when it's a cosmic ray
isFakeCR = 0	Has this particle been labelled as a cosmic ray when it's a test beam
isSplitTB = 0	Has the test beam particle been split into multiple hierarchies
isSplitCR = 0	Has the cosmic ray particle been split into multiple hierarchies
isLost = 0	Did we not reconstruct anything…

Event Validation Algorithm Root Trees

Test Beam Hierarchy Event Validation Algorithm (additional variables):

mcPrimaryTier = (vector<int>*)0x7f900ee52f00	The MC hierarchy tier (i.e. 0 = parent, 1 = daughter..;)
nPrimaryMatchedTBHierarchyPfos = (vector<int>*)0x7fe0a7225a3	How many matches involved particles in the reconstructed test beam hierarchy
bestMatchPfoTier = (vector<int>*)0x7fe0a7225a30	The Reco hierarchy tier (i.e. 0 = parent, 1 = daughter..;)
nTargetTBHierarchyMatches	Total number of matches involving particles in the MC test beam hierarchy
estMatchPfoIsTestBeamHierarchy = (vector<int>*)0x7fe0a7225a3	Is best match pfo in test beam hierarchy
bestMatchPfoRecoTBId = (vector<int>*)0x7fe0a7225a30	The ID of the reconstructed test beam particle parent (ideally just one ID per hierarhcy, multiple IDs = split)
isCorrectTBHierarhcy	Each target has a single unique match all together in the same hierarchy
isSplitTBHierarhcy	Have we split up the hierarchy into multiple reconstructed hierarchies.
isFakeTBHierarhcy	Have we labelled some parts of the hierarchy as cosmic rays.

Running The Jobs

I've put together some scripts that help with the running of the validation metric jobs on the Cambridge system: https://github.com/StevenGreen1/ValidationMetricsScripts

These scripts rely on a certain folder structure to work, which is as follows:

- The parent directory to run them is : /usera/USERNAME/LAr/Jobs/protoDUNE/YEAR/MONTH/JOBNAME/Condor (e.g. /usera/sg568/LAr/Jobs/protoDUNE/2019/October/ProtoDUNE_RecoMetrics/Condor).
- The pndr files are found in : /r06/dune/protoDUNE/SAMPLE/DETECTOR/LArSoft_Version_VERSION/Beam_Cosmics/MOMENTUMGeV/SPACECHARGEOPTION (e.g. /r06/dune/protoDUNE/mcc12_Pndr/ProtoDUNE-SP/LArSoft_Version_v08_30_02/Beam_Cosmics/6GeV/SpaceCharge).
- The pndr files have the following naming structure : SAMPLE_Pndr_DETECTOR_LArSoftVersion_VERSION_Beam_Cosmics_Momentum_MOMENTUMGeV_JobNumber.pndr (e.g. mcc12_Pndr_ProtoDUNE-SP_LArSoft_Version_v08_30_02_Beam_Cosmics_Momentum_6GeV_9.pndr

Running The Jobs

Run them as follows:

1) Source setup.sh, then inside the Condor folder run `pandora_setup`. This will build the standalone Pandora reconstruction packages.

2) Edit LArReco.sh. There are 3 bash variables in the LArReco script to set the JOBNAME, MONTH and YEAR. This script also points to the geometry xml file and if this requires changing should be done so here.

3) Create a `Settings` folder in the Condor folder and put in a PandoraSettings_Master file. See example : https://github.com/StevenGreen1/ValidationMetricsScripts/blob/master/Settings/PandoraSettings_Master_ProtoDUNE.xml. This runs both the TestBeamValidation and TestBeamHierarchy Event Validation Algorithms.

4) Run `python GenerateSettings.py` having edited the dictionary eventsToRun to select the samples you want to process. This will create a one job, one unique input xml file, per available pndr file. These jobs will be detailed in the CondorRunFile.txt file generated.

5) Run `CondorSupervisor.py LArReco.sh CondorRunFile.txt 1000`, where the 1000 is the maximum number of jobs to submit (vary as you see fit).

The output root files can be found in the folder declared on this line of the GenerateSettings.py script : https://github.com/StevenGreen1/ValidationMetricsScripts/blob/master/GenerateSettings.py#L69

Analysing The Jobs

I've also made some hopefully helpful scripts for quickly analysing the jobs they can be found here : https://github.com/StevenGreen1/Analysis_ProtoDUNE_RecoMetrics

The master branch of this script will run on the root files produced from the TestBeamEventValidationAlgorithm (and not the TestBeamHierarchyEventValidationAlgorithm)

Simple checkout the code into your local directory then:

- `mkdir build`
- `cd build`
- `cmake ..`
- edit line 54 of src/Analysis.cxx to load your root files
- `make -j4 install`
- `cd ../bin`
- `./Analysis`

And you'll get a variety of plots including reconstruction efficiencies, purities, completenesses for test beam particles and cosmic rays.

Analysing The Jobs

There is also a feature branch of this code (feature/HierarchyMetrics) that works on the root files produced form the TestBeamHierarchyEventValidationAlgorithm. This is slightly more experimental and I will try and clean it up, but if run directly will give you similar plots, but for the parent and daughter particles in the hierarchy rather than folding it all back togheter.

Fin.