DEEP UNDERGROUND
NEUTRINO EXPERIMENT

LArG4 Calorimeter

Steven Green

2nd October 2019

Introduction

These slides describe a toy model of a LArTPC detector implemented within Geant4.

The simulation of the LArTPC is broken down into two steps:

1) Geant4 : Traverse particles in the detector and record their 3D information in an output xml file. *https://github.com/StevenGreen1/Geant4Calorimeter*
2) LArTPC Simulation & Reconstruction : Read in the xml file and project the hits into the three Pandora views. *https://github.com/StevenGreen1/LArReco/tree/feature/LArG4Calo*

Geant4 : How to Get the Code

These instructions are designed to work on a machine with cvmfs that can access the setup_dune.sh script. The only dependencies are ROOT and Geant4.

This should also work with setup_uboone.sh (and similar). The code also works if you have ROOT and Geant4 installed on a local machine (i.e. no LArSoft), but you need to set some environment variables to ensure the CMake script can find ROOT and Geant4 (left as an exercise to the user…).

How to build the code:

cd /Where/I/Want/Code/To/Live

git clone https://github.com/StevenGreen1/Geant4Calorimeter ./

source **setup.sh**

mkdir build

cd build

cmake ..

make -j4 install

Done…

setup.sh

```
source /cvmfs/dune.opensciencegrid.org/products/dune/setup_dune.sh

setup boost v1_56_0 -q e6:prof

setup clhep v2_3_4_6 -q e17:prof

setup cmake v3_14_3 -f Linux64bit+2.6-2.12

setup gcc v7_3_0 -f Linux64bit+2.6-2.12

setup geant4 v4_10_3_p03c -f Linux64bit+2.6-2.12 -q e17:prof

setup root v6_16_00 -f Linux64bit+2.6-2.12 -q e17:prof
```

Geant4 : How to Run the Code

cd bin

./g4TPC scripts/Input.xml

Done….

Output.xml : Output xml file name. Remember to include the .xml part.

MaxNEventsToProcess : Number of events to simulate. Random seed set in generator based on CPU time, so unique sample every time if using particle gun. If using genie events, the number of events processed will either match this number or be less if the tracker file contains less events.

Particle Gun:

- Use : Use particle gun bool.
- Species : Particle gun species to simulate. Anything in this list : http://fismed.ciemat.es/GAMOS/GAMOS_doc/GAMOS.5.1.0/x11519.html. e.g. mu-. Direction of simulated particle is isotropic.
- Energy : Particle energy.
- ParticlesPerEvent

Genie:

- Use : Use genie events bool.
- Tracker file : File containing genie events.

```
<G4TPC>
    <Output3DXmlFileName>Output.xml</Output3DXmlFileName>
    <MaxNEventsToProcess>3</MaxNEventsToProcess>

    <KeepMCEmShowerDaughters>true</KeepMCEmShowerDaughters>
    <HitThresholdEnergy>0</HitThresholdEnergy>
    <CenterX>0</CenterX>
    <CenterY>0</CenterY>
    <CenterZ>0</CenterZ>
    <WidthX>1000</WidthX>
    <WidthY>1000</WidthY>
    <WidthZ>1000</WidthZ>
    <NLayers>1000</NLayers>

    <ParticleGun>
        <Use>true</Use>
        <Energy>2</Energy>
        <Species>mu-</Species>
        <ParticlePerEvent>1</ParticlePerEvent>
    </ParticleGun>

    <GenieInput>
        <Use>false</Use>
        <TrackerFile>GenieTrakcerFile.txt</TrackerFile>
    </GenieInput>
</G4TPC>
```

Configurable Detector Size, Center and NLayers to used in 3D binning

Geant4 : What you Get

OUTPUT.XML Contents:

A list of Cells, which are blocks of energy deposits (the real energy deposited by the particle):

Id : A unique counter.

MCID : ID of the MCParticle making the largest energy deposit in the cell.

(X,Y,Z) : Position

Energy : Energy.

```
<Run>
    <Event>
        <Cell Id="20697" MCId="1" X="-15" Y="15" Z="-500" Energy="0.847252" />
        <Cell Id="100697" MCId="1" X="-10" Y="15" Z="-490" Energy="0.760835" />
        <Cell Id="100698" MCId="1" X="-10" Y="15" Z="-485" Energy="0.360251" />
        <Cell Id="140697" MCId="1" X="-10" Y="15" Z="-485" Energy="0.955727" />
        <Cell Id="140698" MCId="1" X="-10" Y="15" Z="-485" Energy="0.786975" />
        <Cell Id="180697" MCId="1" X="-10" Y="15" Z="-480" Energy="1.06231" />
```

Important note: At this stage, the energy deposits are binned in 1mm cube bins. Later on they're protected into LArTPC views that spoof a ~5mm pitch and ~5mm hit width (in the drift direction). If the final binning you're interested in playing with approaches 1mm you'll need to increase the number of layers in the Geant4 step to make this binning finer (https://github.com/StevenGreen1/Geant4Calorimeter/blob/master/src/G4TPCDetectorConstruction.cc#L71)

Geant4 : What you Get

OUTPUT.XML Contents:

Followed by a list of MCParticles in the event:

Id : A unique counter.

PDG : PDG code of the MCParticle.

ParentID : Parent MCParticle Id.

Mass : Mass.

Energy : Energy.

Start/End (X,Y,Z) : Start/End Position

Momentum (X,Y,Z) : Momentum

```
        <Cell Id="7939142" MCId="1" X="210" Y="-20" Z="490" Energy="0.766972" />
        <Cell Id="7979142" MCId="1" X="215" Y="-20" Z="495" Energy="0.860859" />
        <MCParticle Id="1" PDG="13" ParentId="0" Mass="0.105658" Energy="1.10566" StartX="-18.9592"
            StartY="16.7836" StartZ="-525" EndX="220.129" EndY="-22.6778" EndZ="525"
            MomentumX="0.246394" MomentumY="-0.0112282" MomentumZ="1.0726" />
    </Event>
</Run>
```

A list of Cells and MCParticles makes a single event. Multiple events can appear in each run.

LArTPC Simulation & Reconstruction : How to Get the Code

The projecting of the 3D hits made in the previous step into the 2D LArTPC views happens inside a feature branch of LArReco. However, to use it you also need to build Pandora (PandoraPFA), so here are the instructions to do both:

cd /Where/I/Want/Code/To/Live

export MY_TEST_AREA=`pwd`

git clone https://github.com/PandoraPFA/PandoraPFA.git

git clone https://github.com/StevenGreen1/LArReco.git

git clone https://github.com/PandoraPFA/LArMachineLearningData.git

cd PandoraPFA

Edit line 128 of CMakeLists.txt to include -DBoost_NO_BOOST_CMAKE=ON.

> CMAKE_ARGS -DCMAKE_INSTALL_PREFIX=${CMAKE_CURRENT_SOURCE_DIR}/Eigen3-${Eigen3_version} -DBoost_NO_BOOST_CMAKE=ON

mkdir build

cd build

cmake -DCMAKE_MODULE_PATH=$ROOTSYS/cmake -DPANDORA_MONITORING=ON -DPANDORA_EXAMPLE_CONTENT=OFF -DPANDORA_LAR_CONTENT=ON -DPANDORA_LC_CONTENT=OFF -DCMAKE_CXX_FLAGS='-std=c++17 -Wno-implicit-fallthrough' ..

make -j4 install

Continued…

Note: The $ROOTSYS/cmake folder should contain ROOTConfig.cmake

LArTPC Simulation & Reconstruction : How to Get the Code

The projecting of the 3D hits made in the previous step into the 2D LArTPC views happens inside a feature branch of LArReco. However, to use it you also need to build Pandora (PandoraPFA), so here are the instructions to do both:

cd $MY_TEST_AREA

cd LArReco

git checkout origin/feature/LArG4Calo_WritingAlg -b feature/LArG4Calo_WritingAlg

mkdir build

cd build

cmake -DCMAKE_MODULE_PATH="$ROOTSYS/cmake;$MY_TEST_AREA/PandoraPFA/cmakemodules" -DPANDORA_MONITORING=ON -DPandoraSDK_DIR=$MY_TEST_AREA/PandoraPFA/ -DPandoraMonitoring_DIR=$MY_TEST_AREA/PandoraPFA/ -DLArContent_DIR=$MY_TEST_AREA/PandoraPFA/ -DCMAKE_CXX_FLAGS=-std=c++17 ..

make -j4 install

cd $MY_TEST_AREA

cd LArMachineLearningData

export **FW_SEARCH_PATH=**$FW_SEARCH_PATH:`pwd`

cd ../LArReco/settings

export **FW_SEARCH_PATH=**$FW_SEARCH_PATH:`pwd`

cd $MY_TEST_AREA

Done…

LArTPC Simulation & Reconstruction : How to Run the Code

cd bin

./PandoraInterface -r full -i ../settings/PandoraSettings_Master_LArG4.xml -g ../geometry/PandoraGeometry_LArG4.xml -e Output.xml (from Geant4 part)

Done….

In this case the geometry for the projection (the wire angles and pitch) are set inside the PandoraGeometry_LArG4.xml file. You can vary the wire angles and see how this affects the input 2D hits.

LArTPC Simulation & Reconstruction : How to Get the Images

Inside the feature branch of LArReco, I've added an algorithm that will write out the calo hit information to a text file. The settings file PandoraSettings_Master_LArG4.xml loads the event display to show you the calo hits in the three different views and then runs the hit writing algorithm only (no further reconstruction is attempted).

Three text files will be made per event (U, V and W views). Each line in these text file contains the hit location for all calo hits in an event.

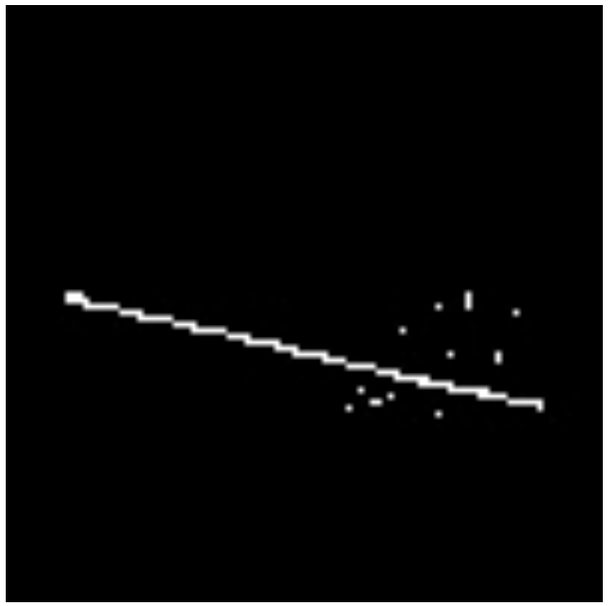
The first item on the line is a time and data stamp, following that is the first number on the line is the total number of calo hits in the event, then numbers appear in batches of 5 describing the calo hit properties (X, Y, Z, Energy, PDG Code, NuanceCode), finally a 1 appears at the end of the line, which is redundant.

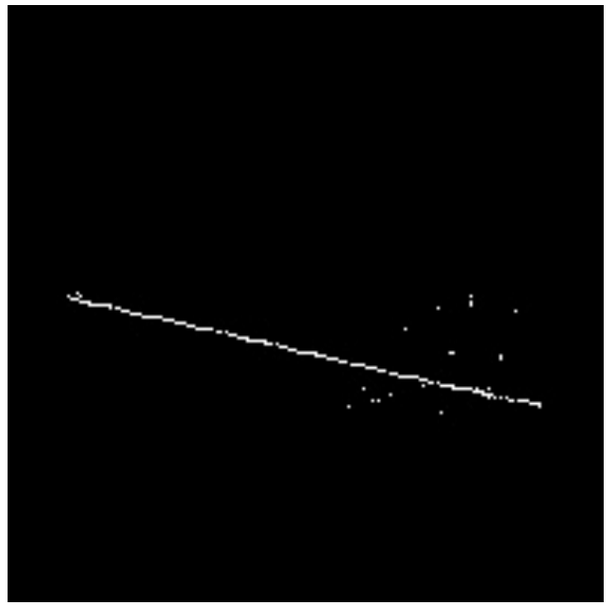
10/03/19_15:16:19,213,-1.5,0,-49.8368,0.107036,13,2001,-1,0,-48.8784,
0.0207015,13,2001…….etc……,-1,0,-48.3992,13,2001,1

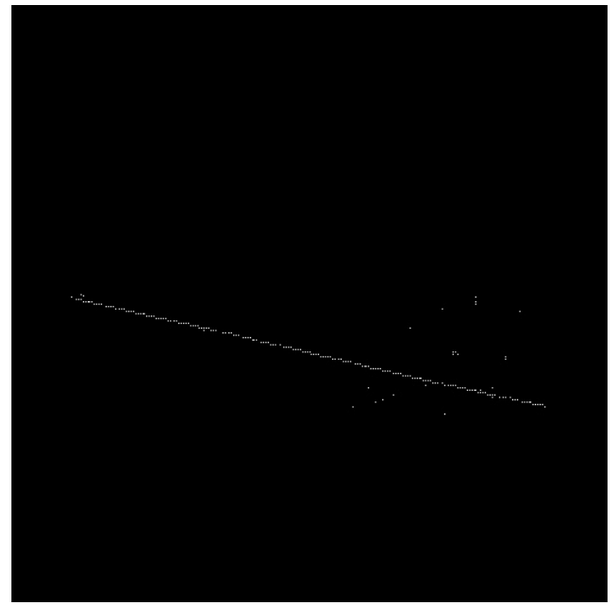
LArTPC Simulation & Reconstruction : How to Get the Images

Here's a link to a python script that will convert these text files to jpegs : https://www.hep.phy.cam.ac.uk/~sg568/Image.py

In this script the variable imageSize controls the resolution of the image. In these example I used the 1m cube detector and 100, 200 and 500 pixels to make this image for the W view :







It's clear that using 500 pixels for this detector because adding more pixels isn't improving the resolution because of the ~5mm intrinsic detector resolution (so 200 is optimal, but remember the U and V views are distorted due to the non-zero angle of the wires).

Once you've got a detector size you're happy with, you can just reprocess the same text files with this script (or your own) to give you the different pixel counts you're interested in.