# Neutron selection and software status

Ciro Riccio, Abraham Teklu, Guang Yang, Eric Chong

SuperFGD beam test analysis meeting

November, 19th 2020

Stony Brook University

# Overview

- Software development strategy
- Software development status
- Neutron selection overview, cut flow and first results
- Conclusions and next steps

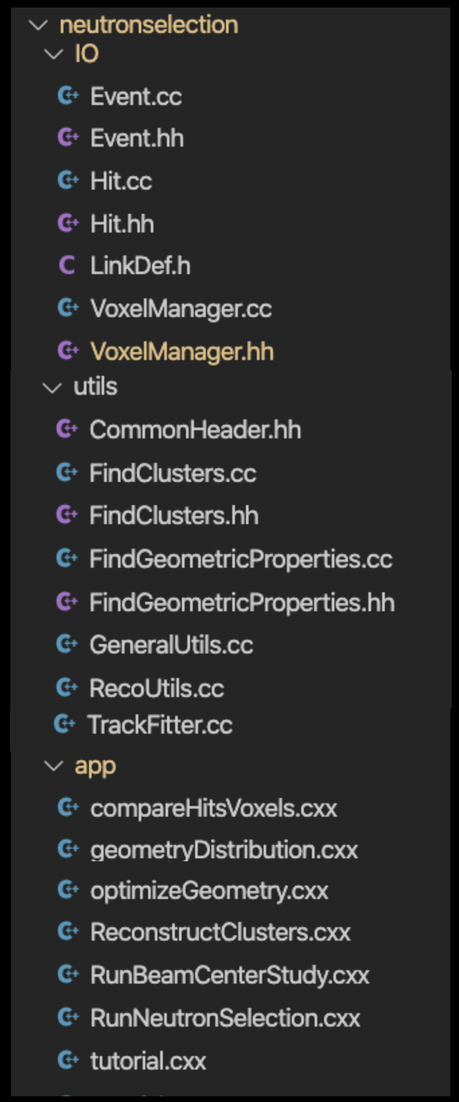
# Software development strategy

- Created a group on gitlab (for non-T2K people) called Neutron Test Beam Analysis
- There are three projects (all under MIT license):
  - neutronSimulation: package for the Monte Carlo simulation of experimental set-up used @LANL
  - neutronSelection: package to select neutrons from the data taken during the test beam largely based on sfgd_framework developed by Cesar, Dana and Wilf
  - neutronXsecFitter: Package to extract the neutron xsec from the data taken during the test beam

# Software development status

- neutronSimulation: ready
- neutronSelection:
  - Implemented Time clustering (Abe and Ciro) and voxelization (Eric)
  - Implemented (Eric) DBSCAN (spatial clustering - similar to what Cesar implemented in sfgd_framework) and Principal component analysis (cluster geometrical properties)
  - Implemented the track fitting the vertex finding (Ciro and Guang - similar to what Cesar implemented in sfgd_framework)
  - Improved documentation (Ciro) and solved some memory leak problems (Ciro and Eric)
  - Ongoing work: implementing new cuts (Ciro and Eric) and Hough transform (David)
- neutronXsecFitter: work in progress

# Neutron selection overview




Base classes:

- Read all the events (Event.*) in a file
- Handle hits (Hit.*) information
- Handle voxel information (VoxelManager.*)

Useful tools:

- List of useful common header
- Some general tool for the processing
- Time clustering and voxelization
- DBSCAN, PCA and track fitting

Neutron selection and code for different optimization/studies

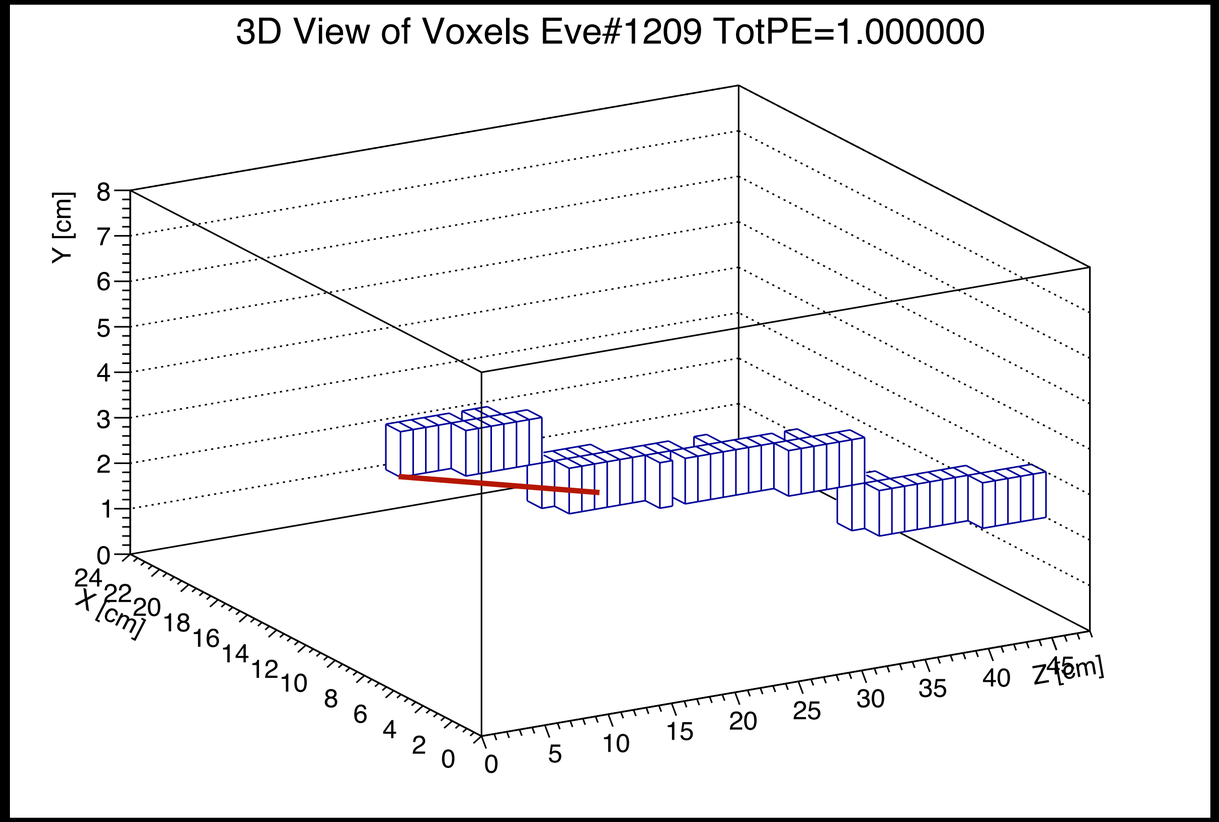
Tutorial: start from this to write an app

# Neutron selection cut flow

| Cut name | Cut descriptions and value |
|---|---|
| **Micropulse time window** | -326 ns < Hit time < 340 ns (90 m) -281 ns < Hit time < 418 ns (20 m) |
| **Number of hits** | #hits > 3 |
| **Time clustering** | If PE > 5 && $t_{hit}$ (i+1) - $t_{hit}$ (i) < 17.5 ns ⇒ same cluster |
| **Voxelization** | Build voxel based on the hits position and time difference between hits (time tick 7, but discussing to remove it) |
| **Voxel PE** | #PE>50 (not really effective we will remove it) |
| **Spatial clustering** | DBSCAN: minimum of 2 nodes and maximum distance of 1.8 cm (to be optimized) |
| **Number of clusters** | Select events with only one cluster |
| **Vertex in FV** | Vertex (first voxel in Z) must be in 2x2 FV (build around the beam center) |
| **3DLine-Point max distance** | Max distance between 3D line from PCA and voxels in a cluster must be < 1.8 cm |
| **Max cluster size** | If the max cluster size in one of the direction is below 5 cm the event is rejected |

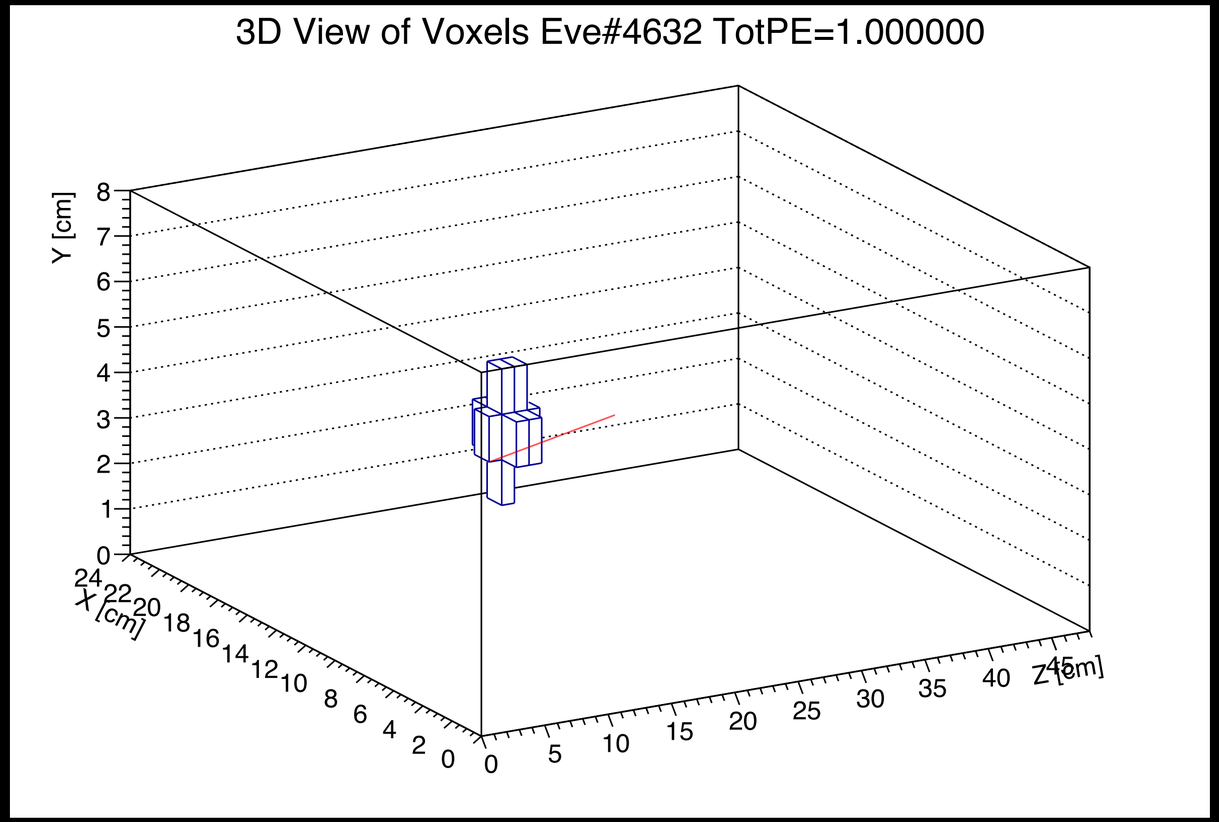
# 3DLine-Point max distance

- Select only event with one cluster
- Perform a PCA
- The main vector represents the direction of 3D line with origin the vertex of the track (red line in figure)
- Compute the distance between the voxel and this line
- If the maximum distance is greater than 1.8 cm the event is rejected



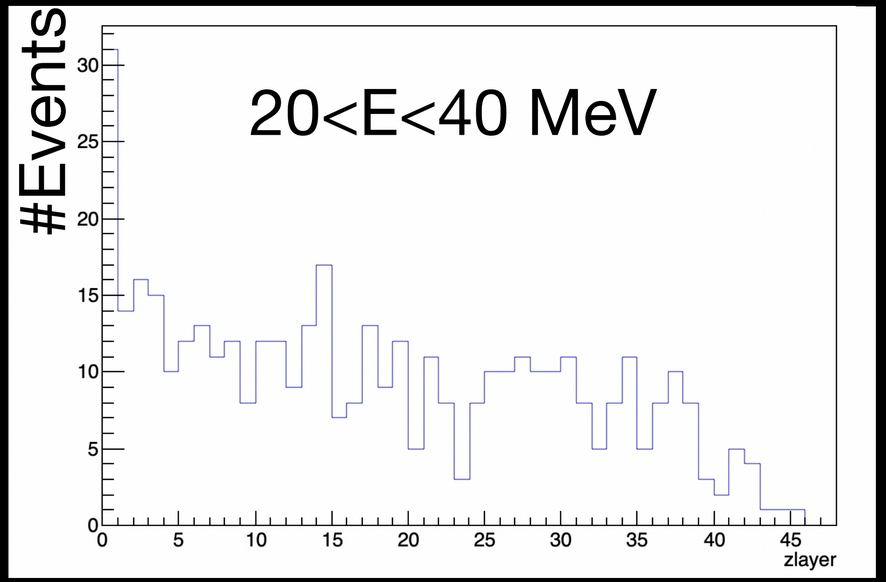
# Max cluster size

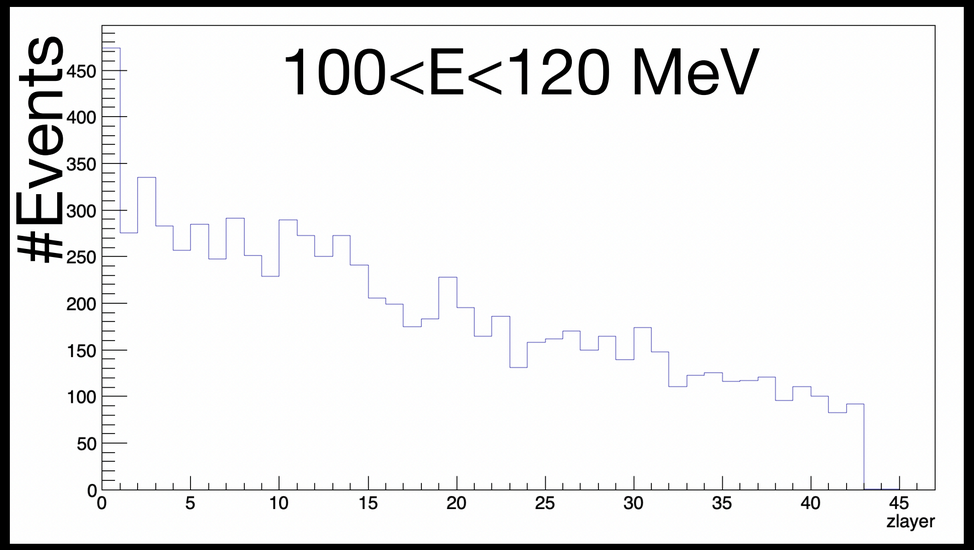
- Can have very small clusters (blobs)
- To reject them we compute the cluster length in X, Y and Z
- This length is compute as the difference between the last and first voxel in a certain direction
- If the maximum distance is lower than 5 then we reject the event

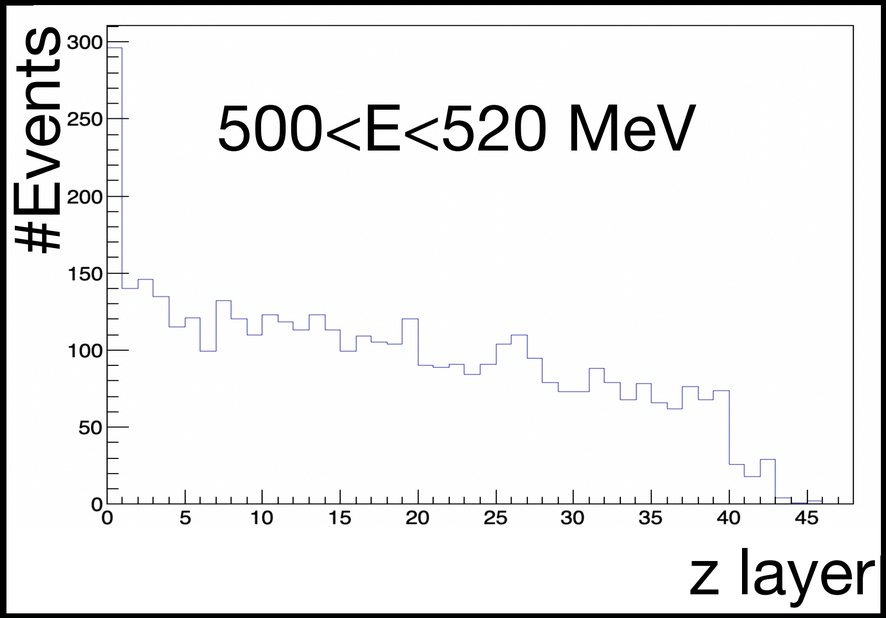


# Selection: first look

- Analyzed run8 take on Dec. 7, 2019 (16 min)
- Histogram every 20 MeV of the #events vs layers (just 3 shown here)
- The cuts reject many low energy events
- For high energies the behavior seems not exponential: need further investigation



# Other possible cuts

- Select only event with one cluster
- Perform a PCA
- Compute the linearity as (principal value - second principal value)/principal value and require it to be greater than 0.9 to remove blobs
- Project the voxels on the second principal vector and compute the distance between the max and min point. This is a measure of the cluster spread
- If this different is greater than 2.5 cm the event is rejected. This should remove multiple track events

# Conclusions

- neutronSimulation: ready
- neutronSelection next steps:
  - Optimize the cuts value
  - Test the Hough Transform implemented by David
  - Speed up the code
  - Improve the handling of the file IO (minor)
- neutronXsecFitter: WIP

# Backup

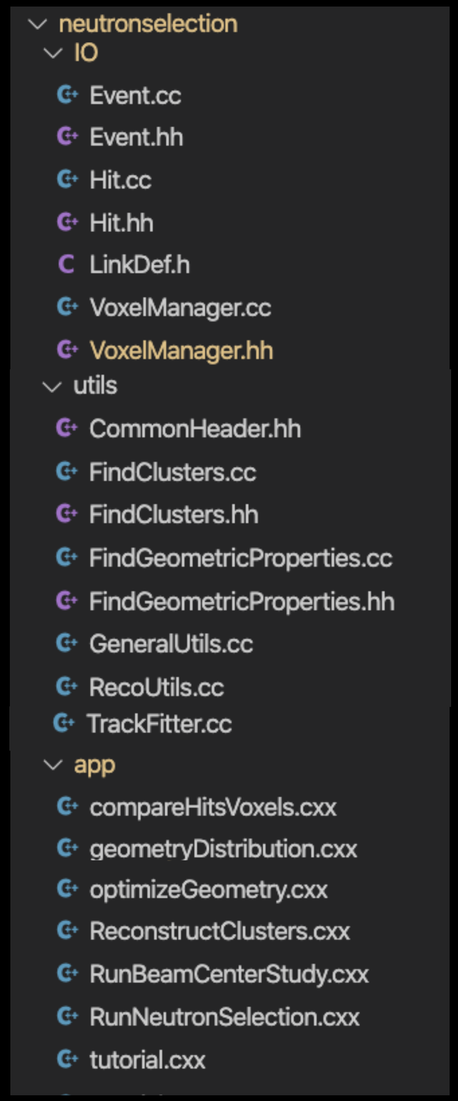
# Neutron energy

Distance (90 or 20 m)

Time of the voxel associated to the vertex

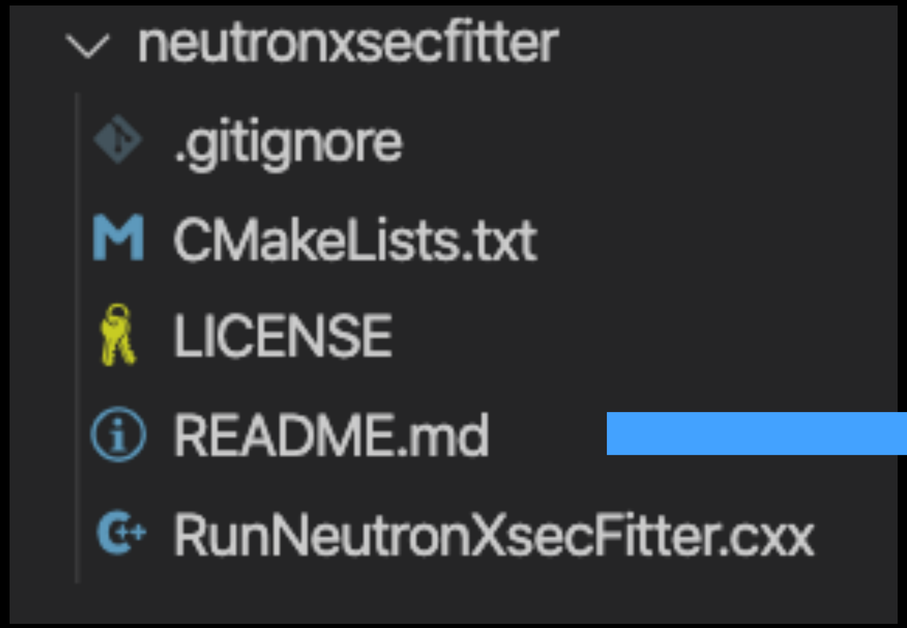
```
double ComputeNeutronEnergy(double time, double d)
{
  double energy=0.;
  Double_t m = 939.5654133;//neutron mass in MeV
  Double_t c = 299792458;  //speed of light in m/s
  Double_t offset = 352;   //time between FEB12 signal and the center of the gamma flash peak in units of 2.5 ns
  Double_t gammaToF = d/c; //photon travel time in secs

  Double_t ToF = (time+offset)*2.5*1e-9+gammaToF; //neutron time of flight in seconds

  Double_t v = d/ToF; //neutron velocity using time of flight and

  energy = m/sqrt(1-pow(v/c,2)) - m;

  return energy;
}
```

# Neutron selection: track fitting




- Fit the linear cluster with a 3D line
- Compute the angle w.r.t. z and select forward going tracks
- Validating with CERN TB proton data we are observing good results (more quantitive results soon) and MC samples
- Find the vertex (preliminary): obtained by finding the intersection of the beam center and the track line on XZ and YZ planes.

# Neutron Xsec Fitter overview




## Introduction

This package is intended to provide all the necessary tools to extract the neutrons cross section from the data taken at LANL.

## Download

To dowloand it:

```
$ git clone git@gitlab.com:neutron-lanltb-analysis/neutronxsecfitter.git
$ git checkout -b <choose a branch name>
```

## Installation

There are several requirements for building the fitter:

- GCC 4.8.5+ or Clang 3.3+ (a C++11 enabled compiler)
- CMake 3.10+
- ROOT 6

Set up the ROOT environment before attempting to build by:

```
$ source /path/to/ROOT/bin/thisroot.sh
```

To build:

```
$ mkdir build; cd build
$ cmake ../
$ make
```

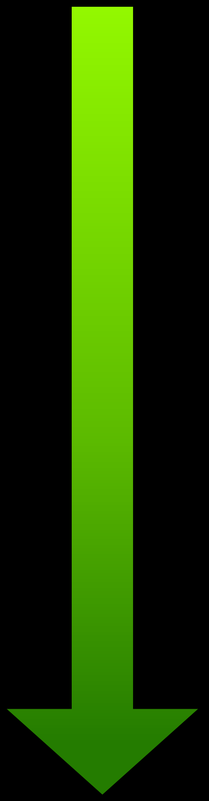
## Running the Code

If the installation succed the application to te used for the selection is saved under build:

```
$ ./RunNeutronXsecFitter.exe
```

# Neutron Xsec Fitter workflow

- Fill an histogram for every energy bin (20 MeV)
- Fit this histogram with an exponential
- Inclusion of systematics:
  - Repeat the fit for every variation of a systematic uncertainty
  - Fill an histogram with every value of the cross-section extracted
  - The systematic uncertainty will be the RMS of this distribution
- Statistical uncertainty: similar to systematics but varying the number of events using a Poisson distribution



# Git workflow

Neutron selection