Update on Michel light/track reconstruction

Will, Pawel, Flavio

(With lots of LArSoft help from Johnny and Jason!)

LArIAT Meeting

Oct 6, 2015

Overview

- Now have working alg/module to process data and reconstruct Michel electron properties
  - Combines information from **light** and **tracks**

Branch: wforeman_OpDetHits → on develop as of 10/5

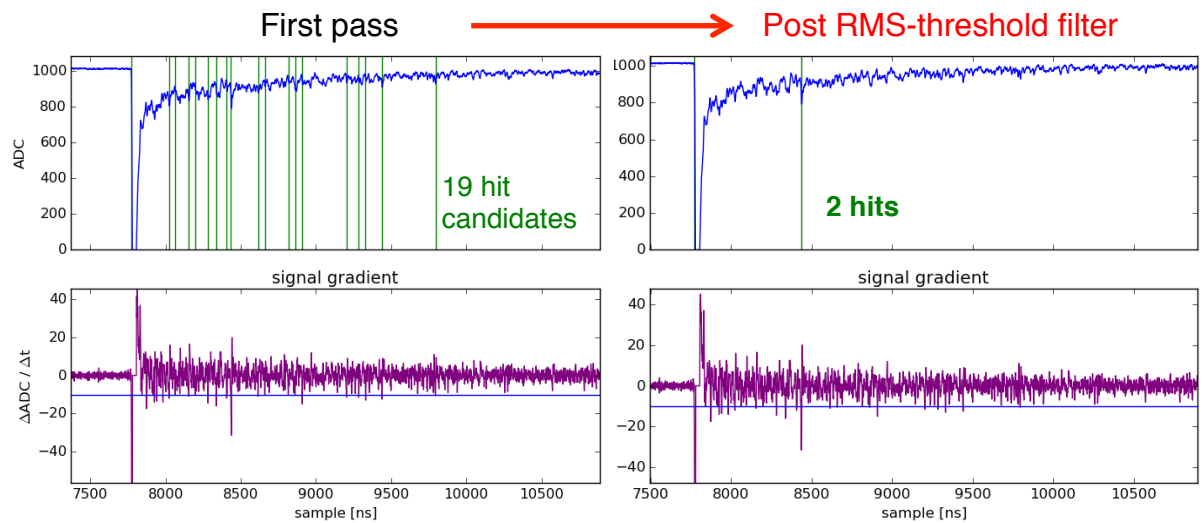
`srcs/lariatsoft/LArIATRecoModule/MichelWfmReco_module.cc`

`srcs/lariatsoft/LArIATRecoAlg/OpHitBuilderAlg.cxx`

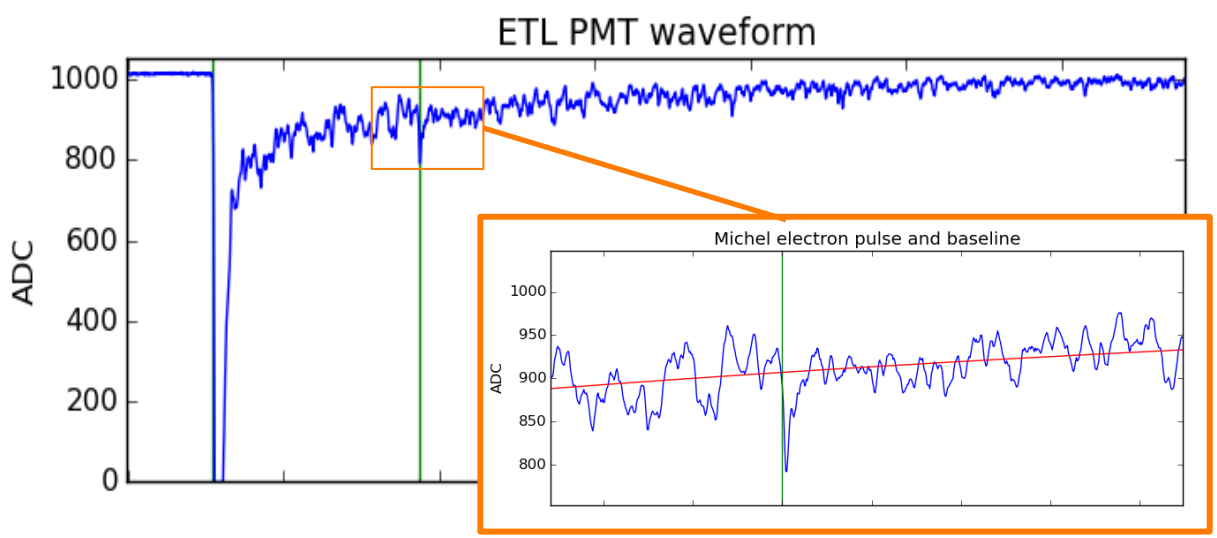
- **Goals**:
  1. Decay spectrum to measure $\tau_{capture}$
  2. Photon spectrum for light yield
  3. Testing ground for algorithms to tag stopping beam μ's
  4. Michel energy spectrum using charge + light
  5. Electron dE/dx

Hit finding and integration

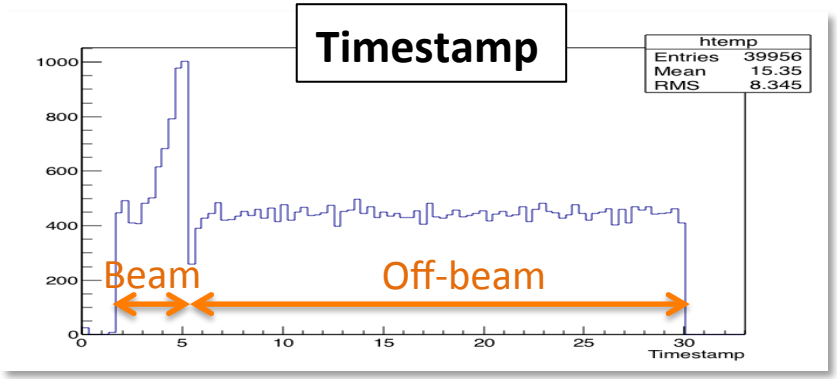
1. Enhanced hit-finding
   - Gradient based finder, with variable threshold tied to local RMS

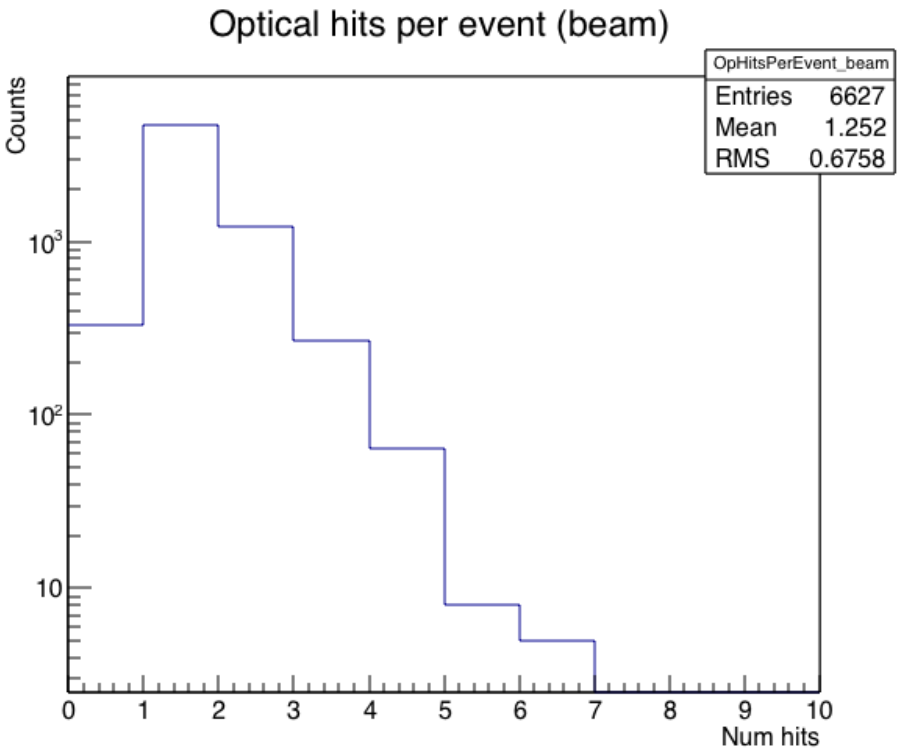


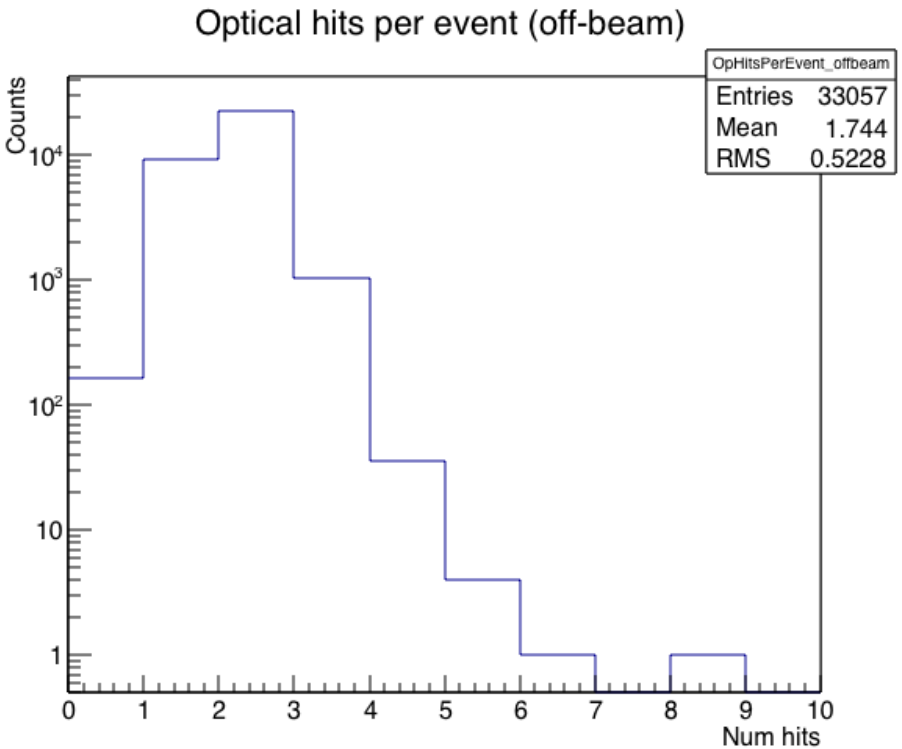
2. Prompt pulse integration using fitted exponential baseline



Number of optical hits per event







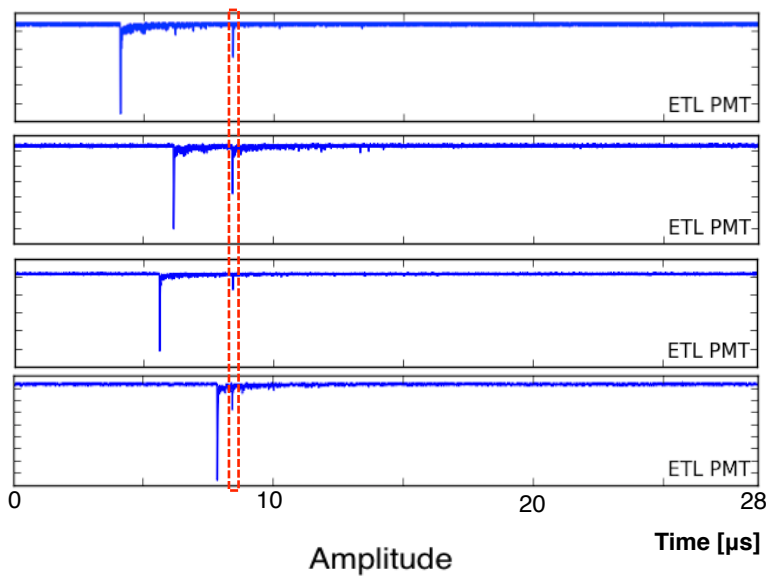
Finding Michel trigger candidates (using light)

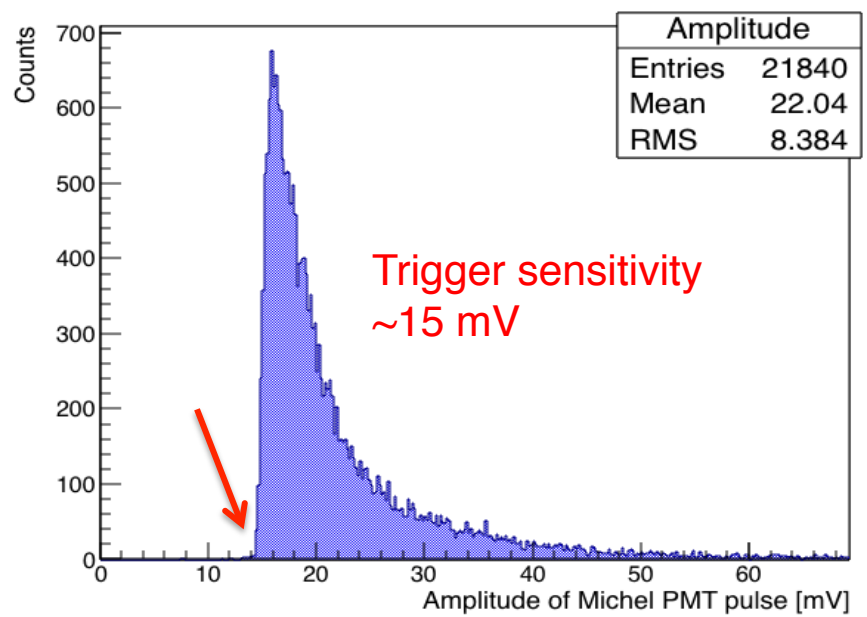
Require:

- Non beam event
- 2 optical hits

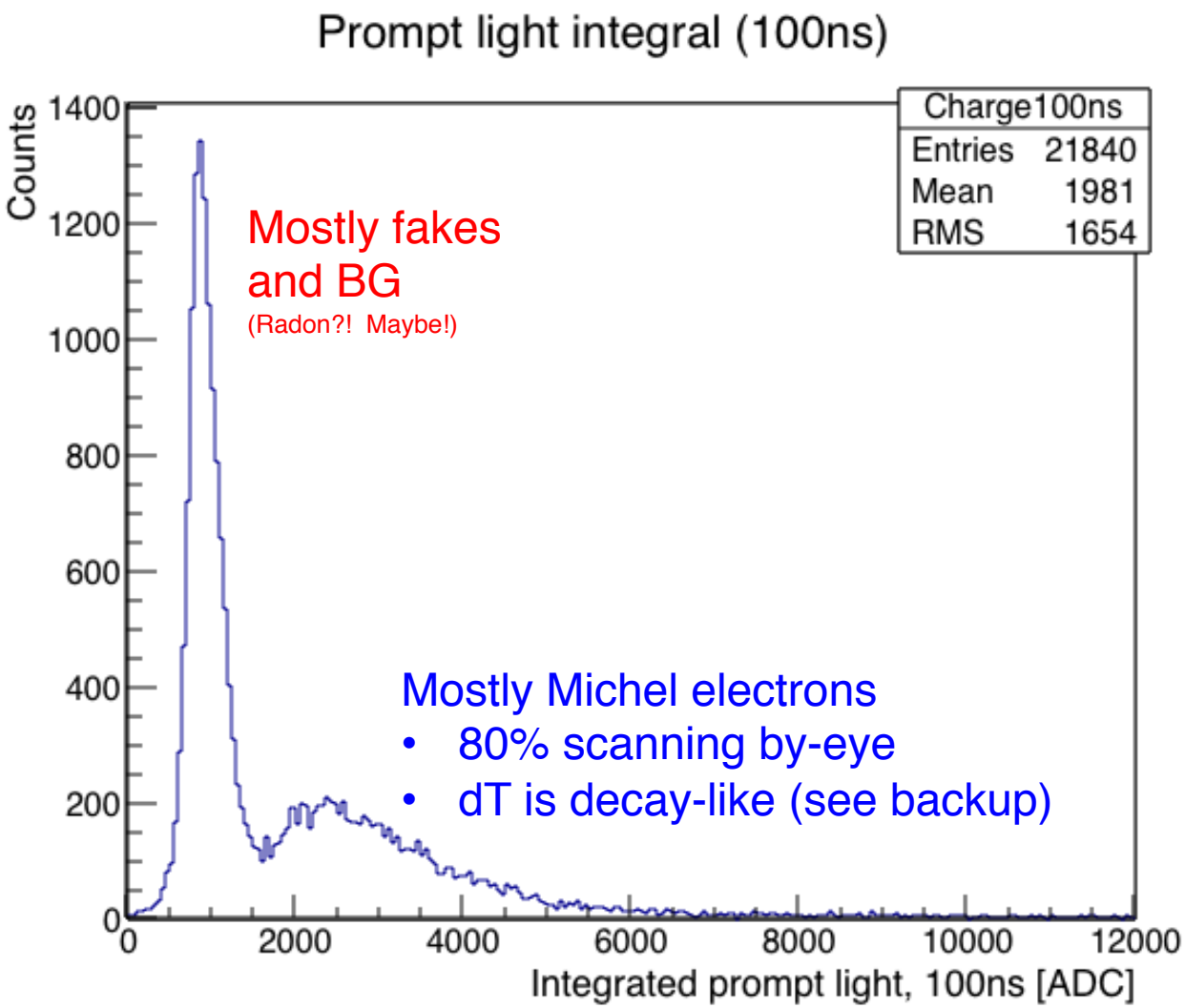
Using ETL (2” PMT) for analysis of Run1 data

Optical hit finding and integration of 2nd (e-candidate) pulses in waveforms





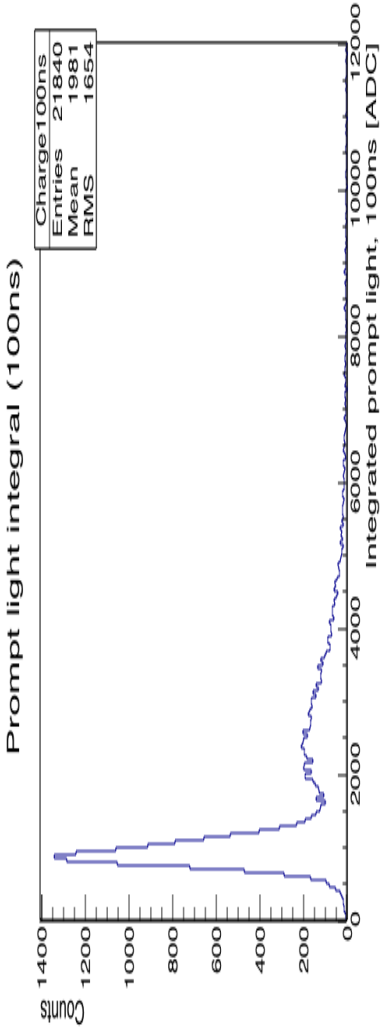
Prompt (100ns) light integral of Michels

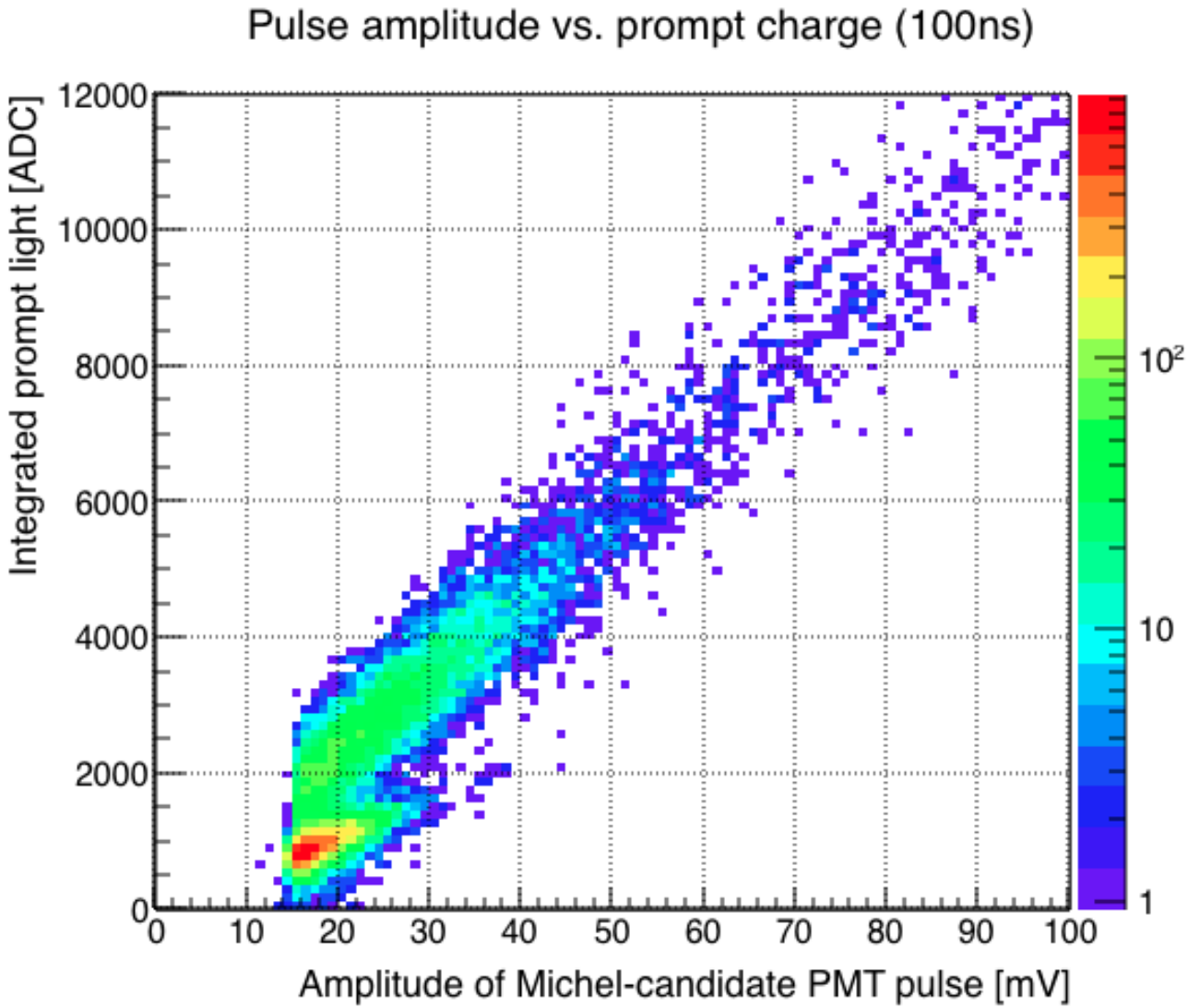


Prior to any further cuts, 2 components to photon spectrum

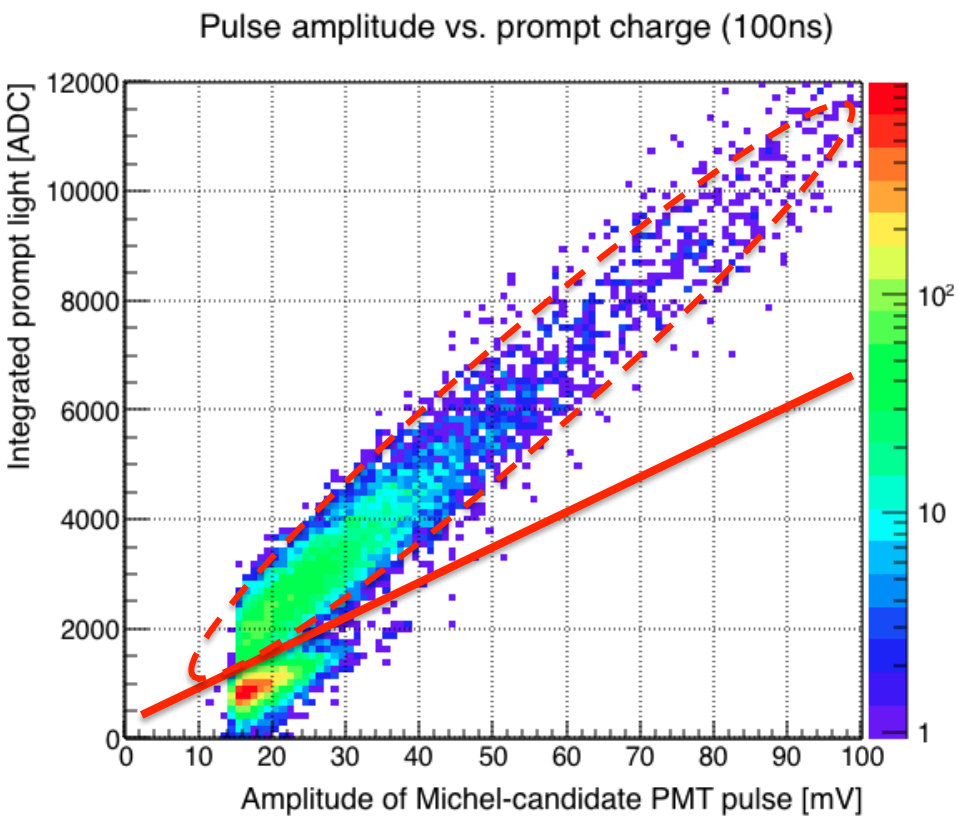
Can we use more information to improve this?...

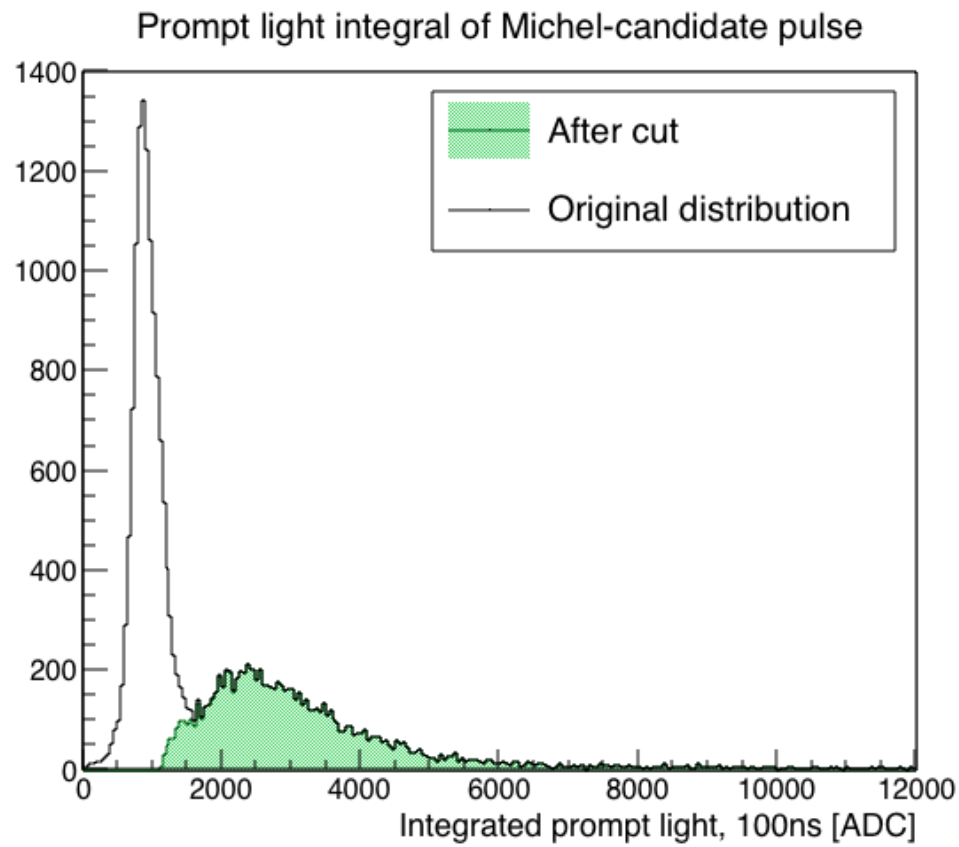
Amplitude vs. prompt light





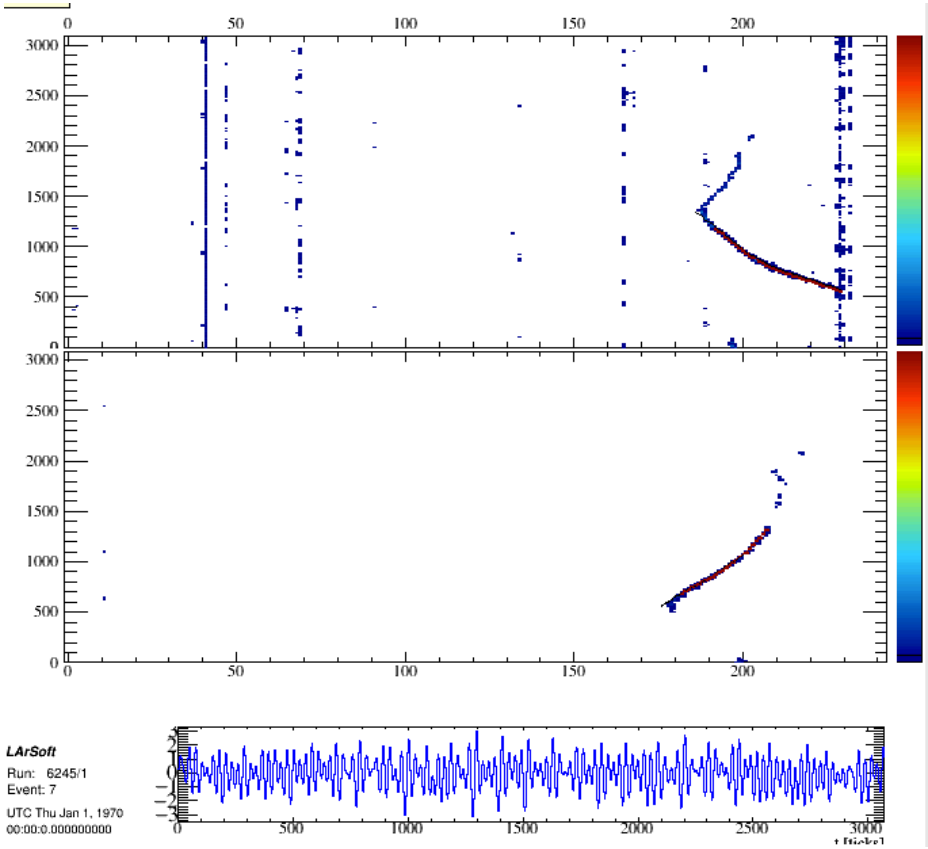
Cutting out the population



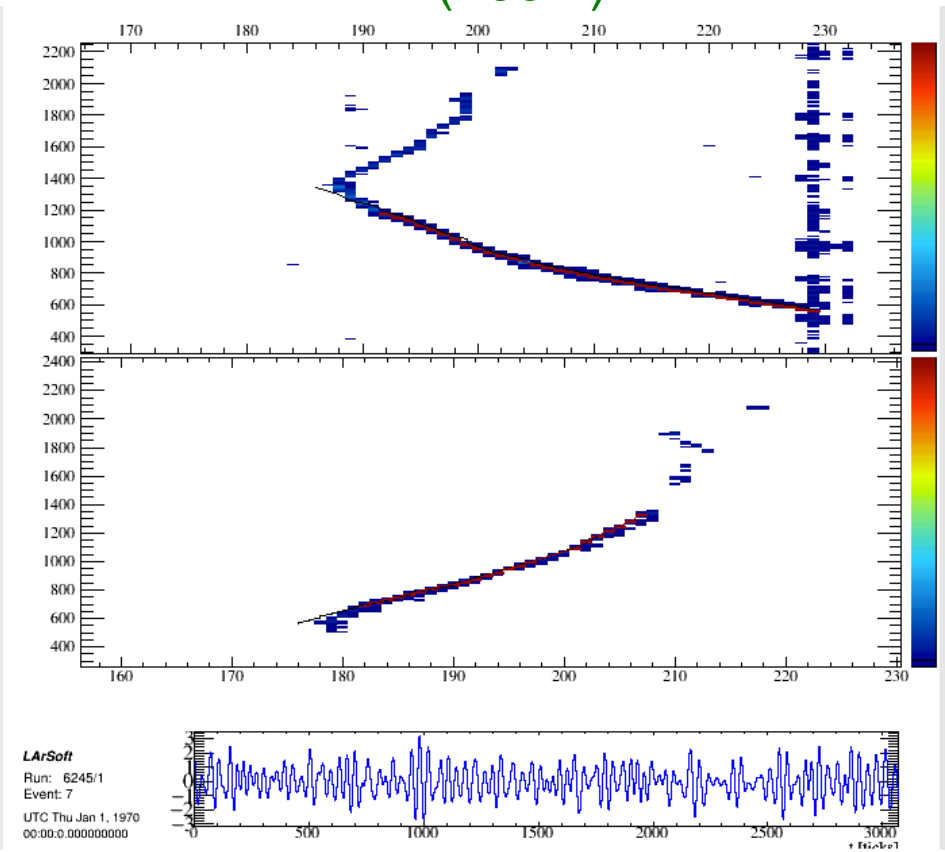


Track information

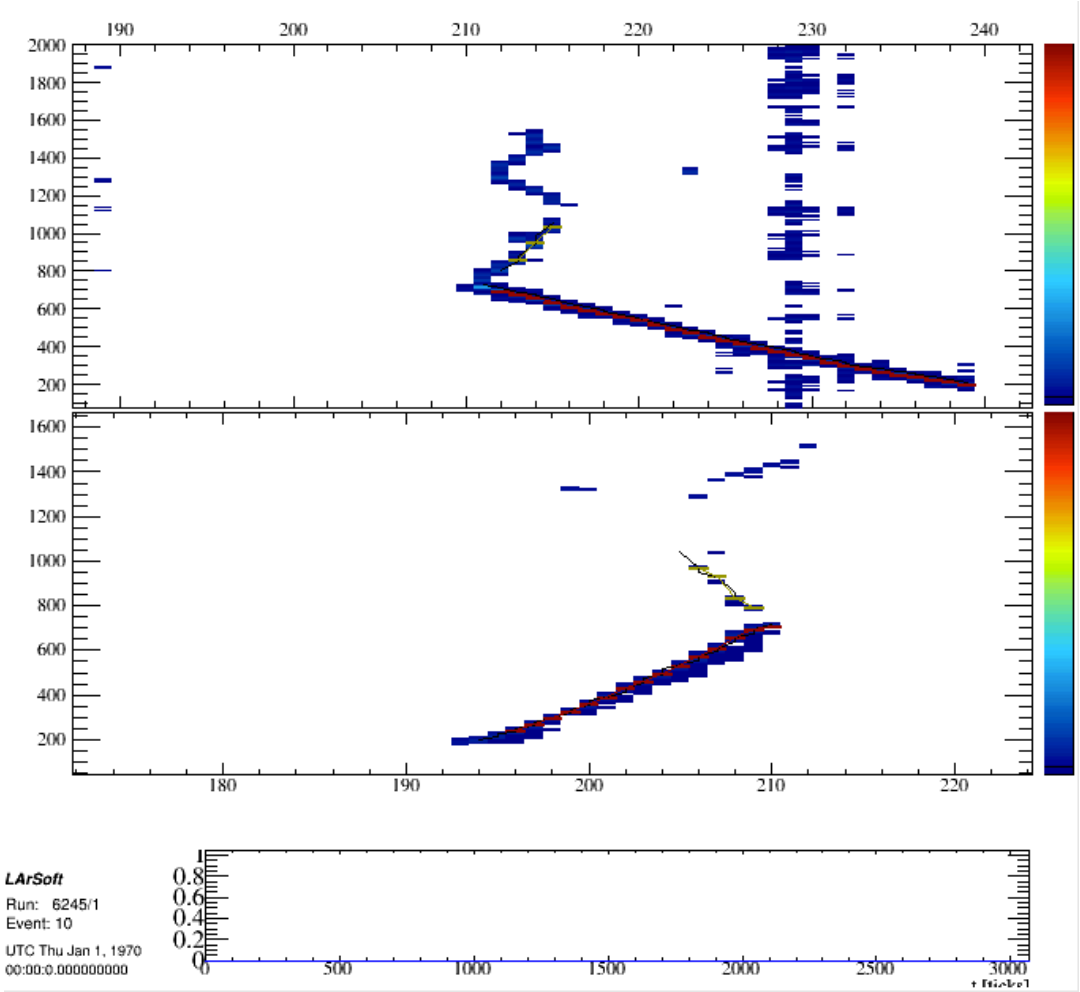
- Using “pmtrack” (in Reco.fcl) as is



(Zoom)



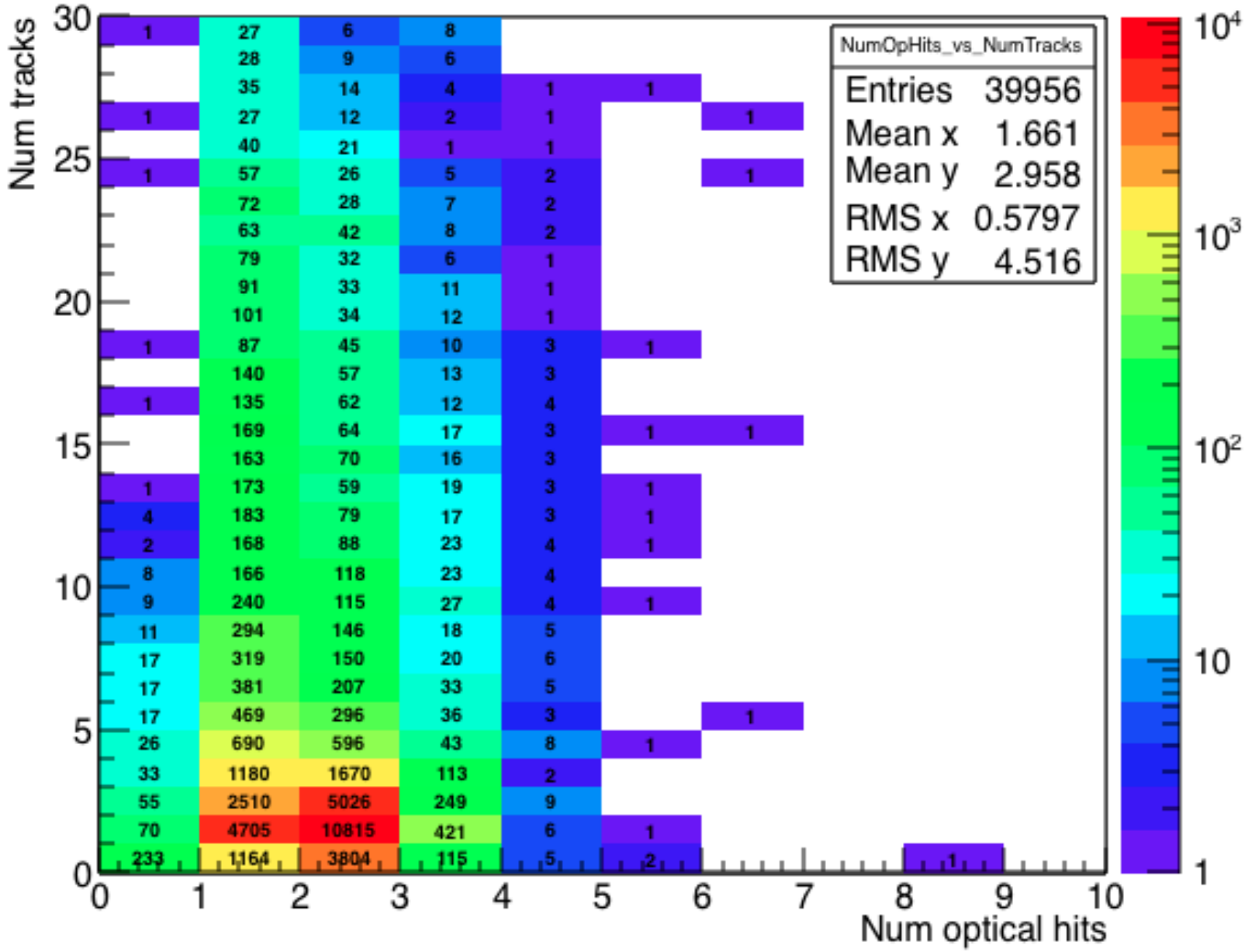
Track information



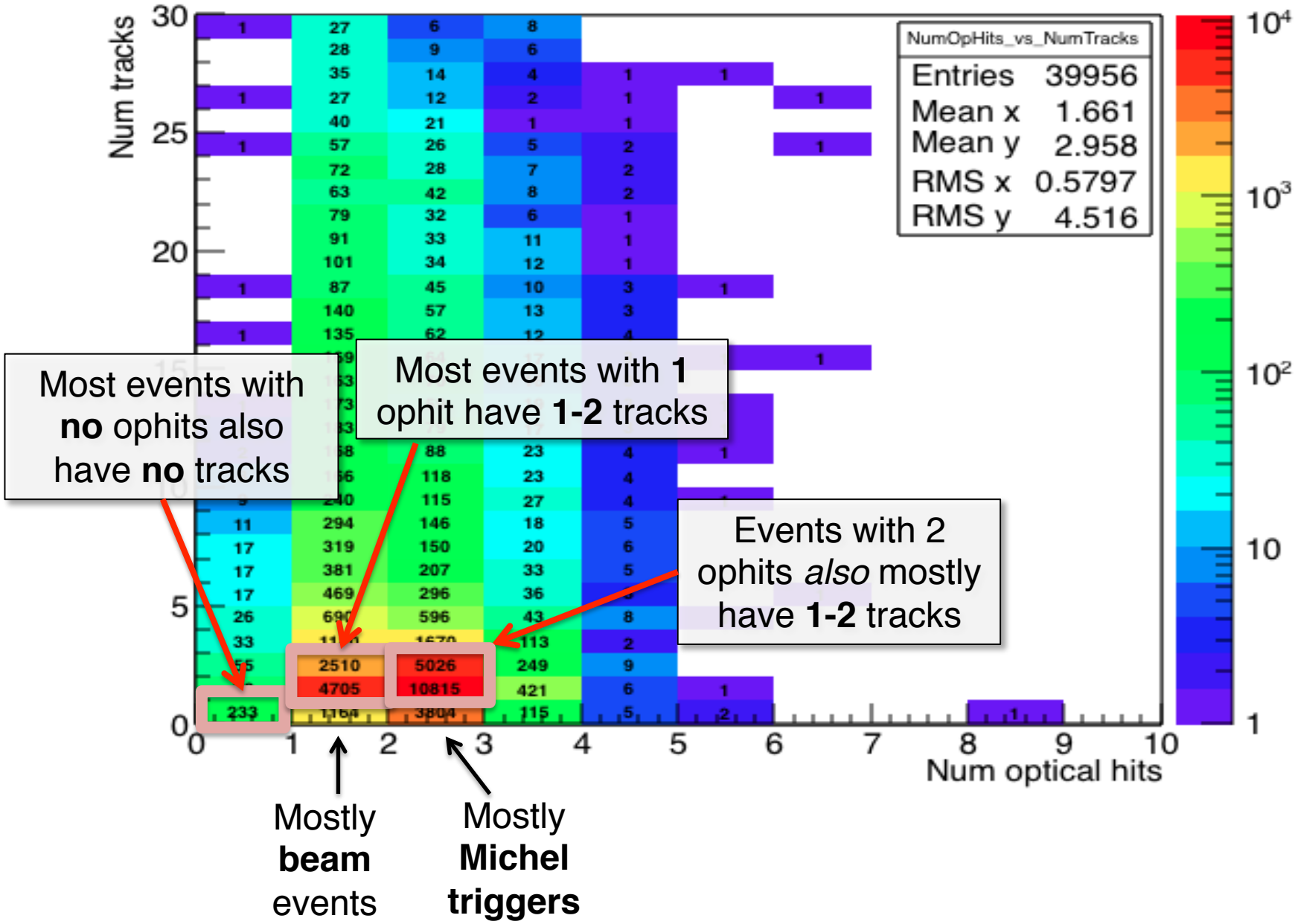
Occasionally, decay electron also gets picked up by tracking…

but not well.

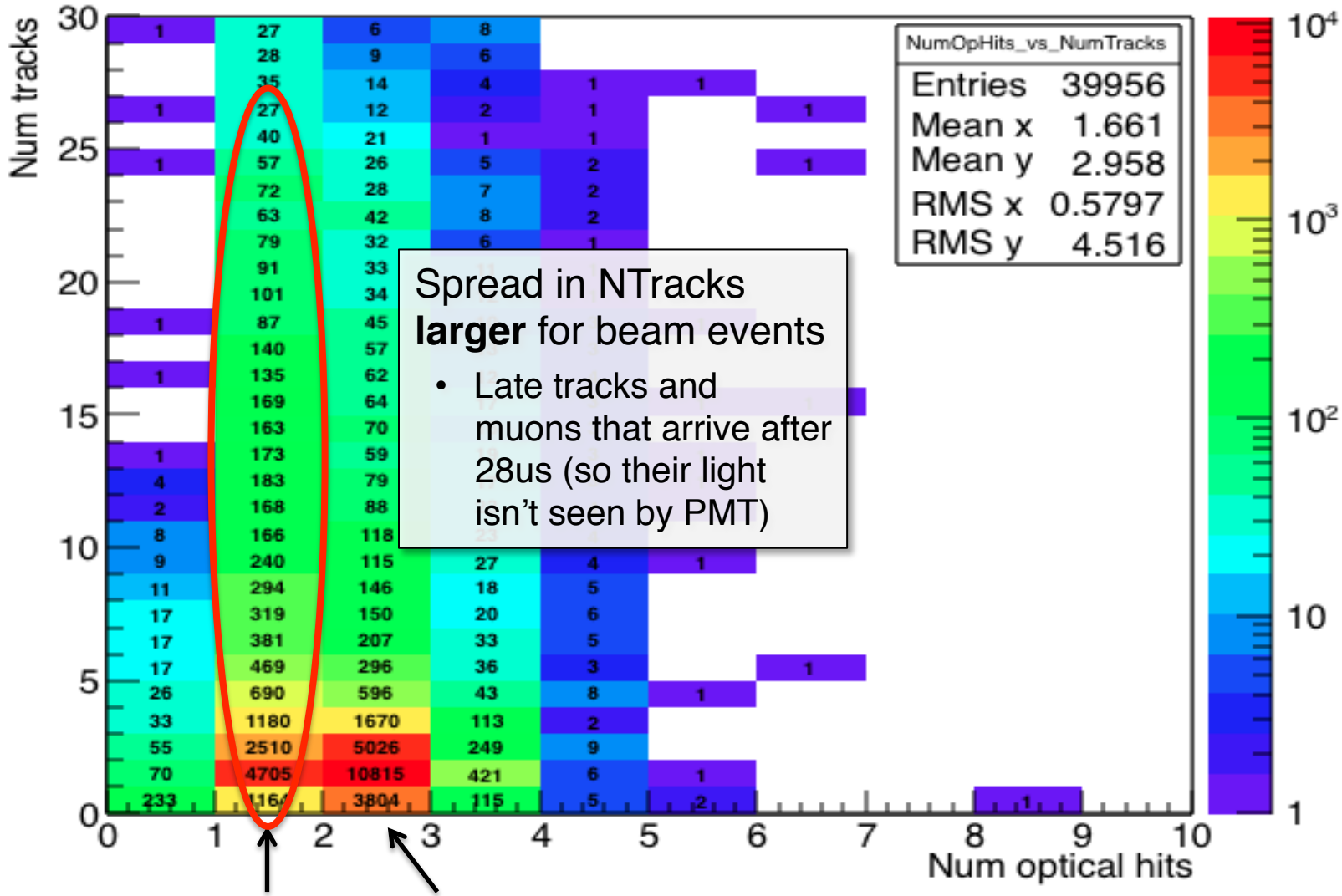
Optical hits vs. number of tracks in event



Optical hits vs. number of tracks in event



Optical hits vs. number of tracks in event

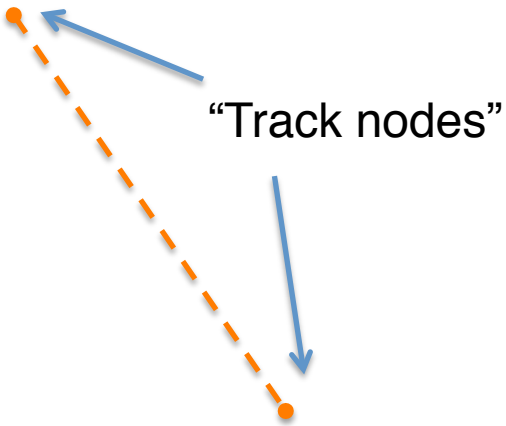


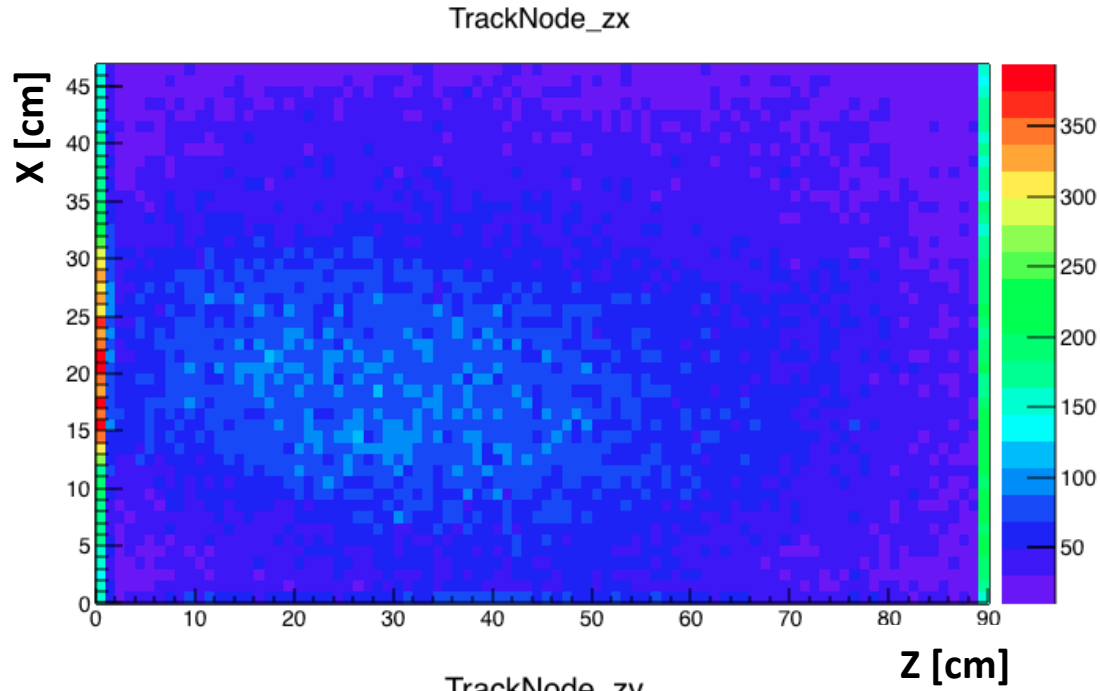
Spread in NTracks **larger** for beam events

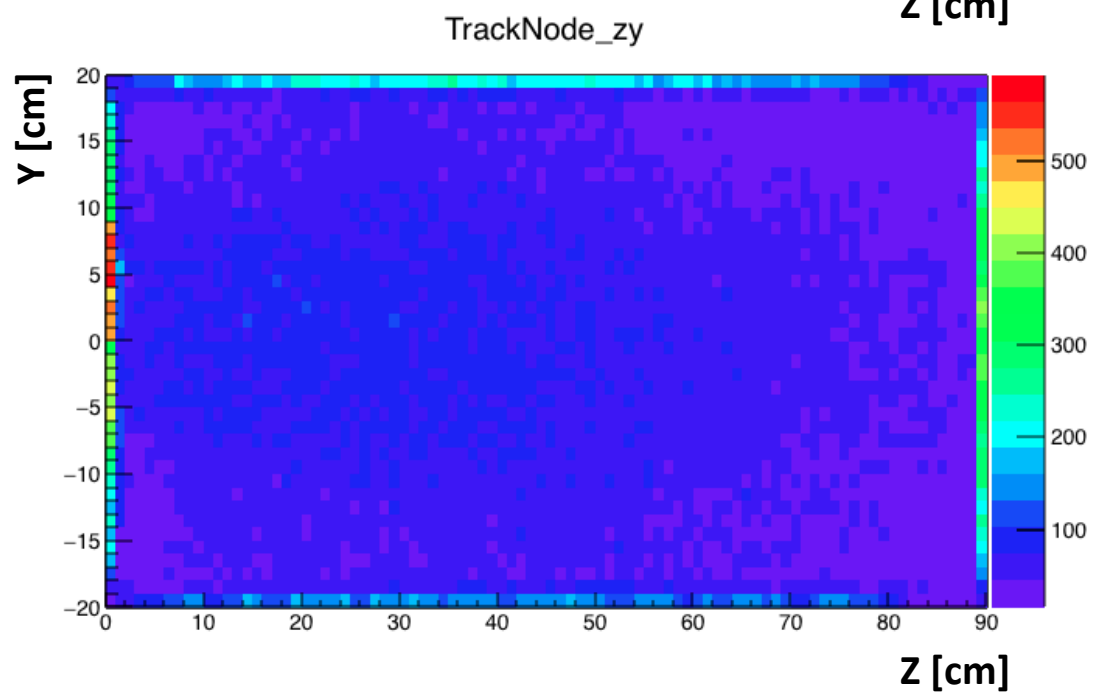
- Late tracks and muons that arrive after 28us (so their light isn't seen by PMT)

Mostly **beam** events

Mostly **Michel triggers**







Buildup of nodes near edges (especially up-stream wall!)

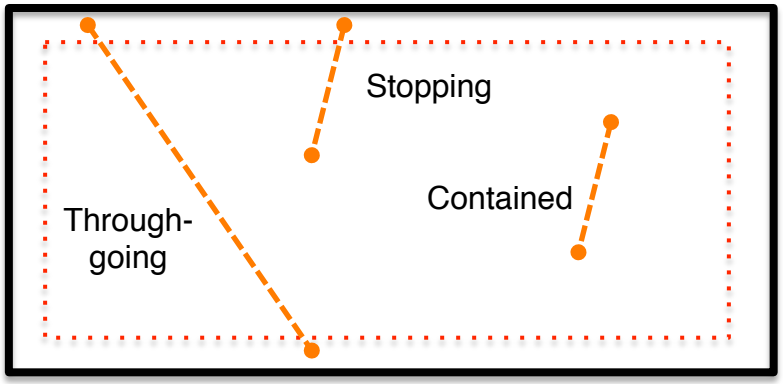
Use these plots to decide on a "fiducial margin"

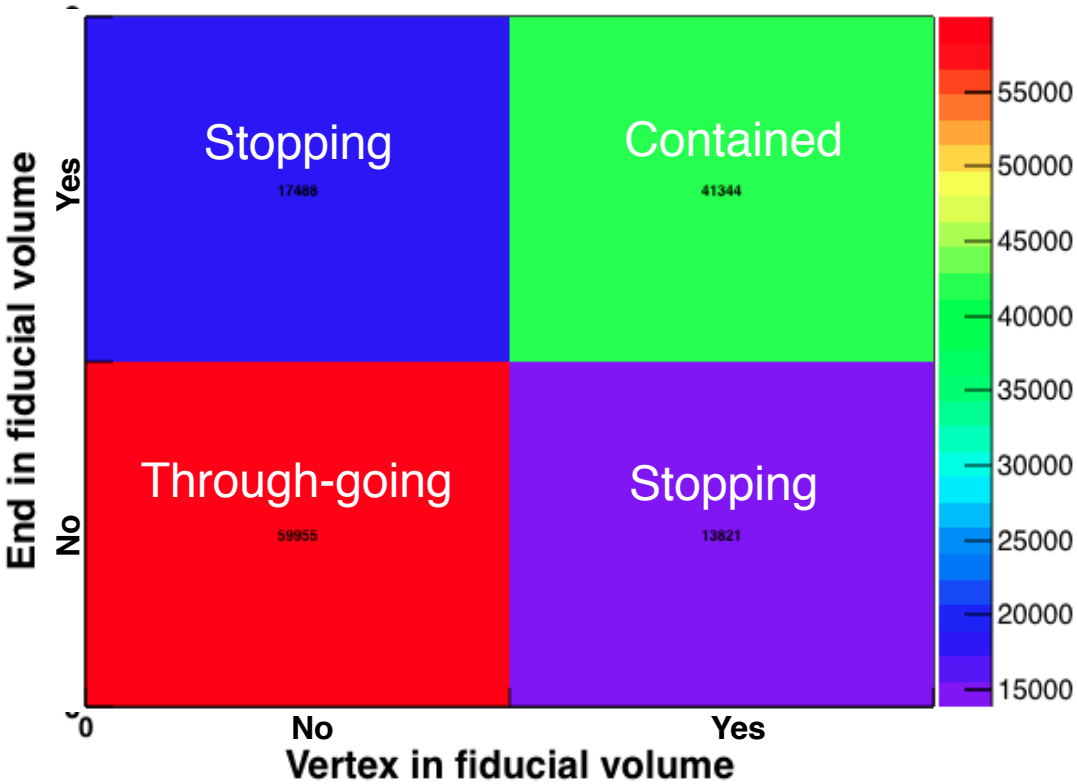
Through-going vs. stopping tracks

A track is called <u>stopping</u> if:

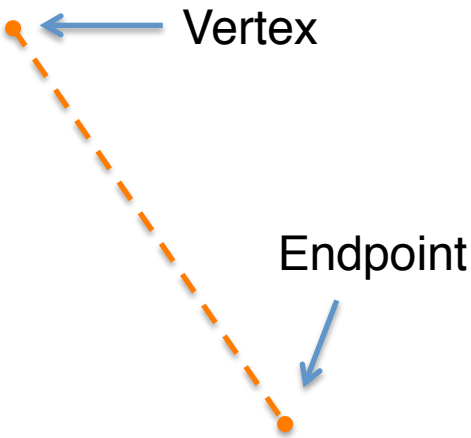
- 1 node is outside the fiducial volume (ie, along edge)
- other node is inside fiducial volume

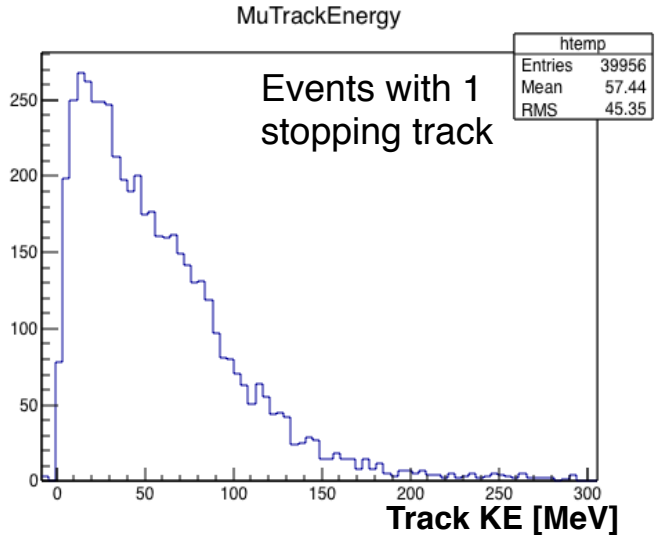
~24% of tracks meet this definition of stopping*

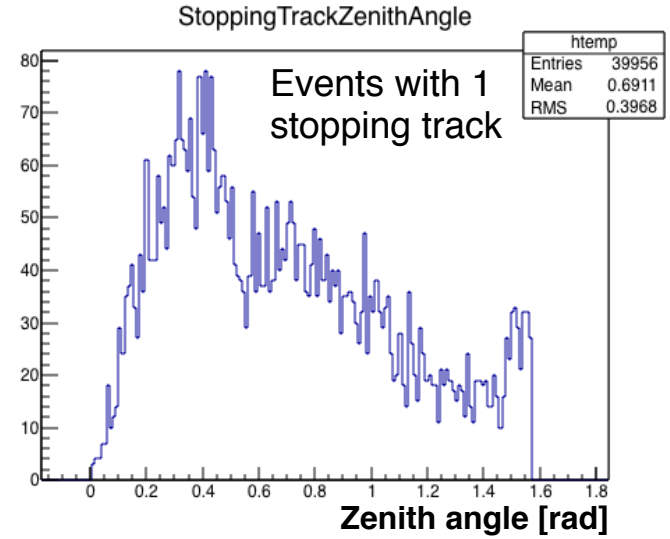




*Using initial x,y,z margins of 5cm,4cm,5cm (can probably reduce these…)

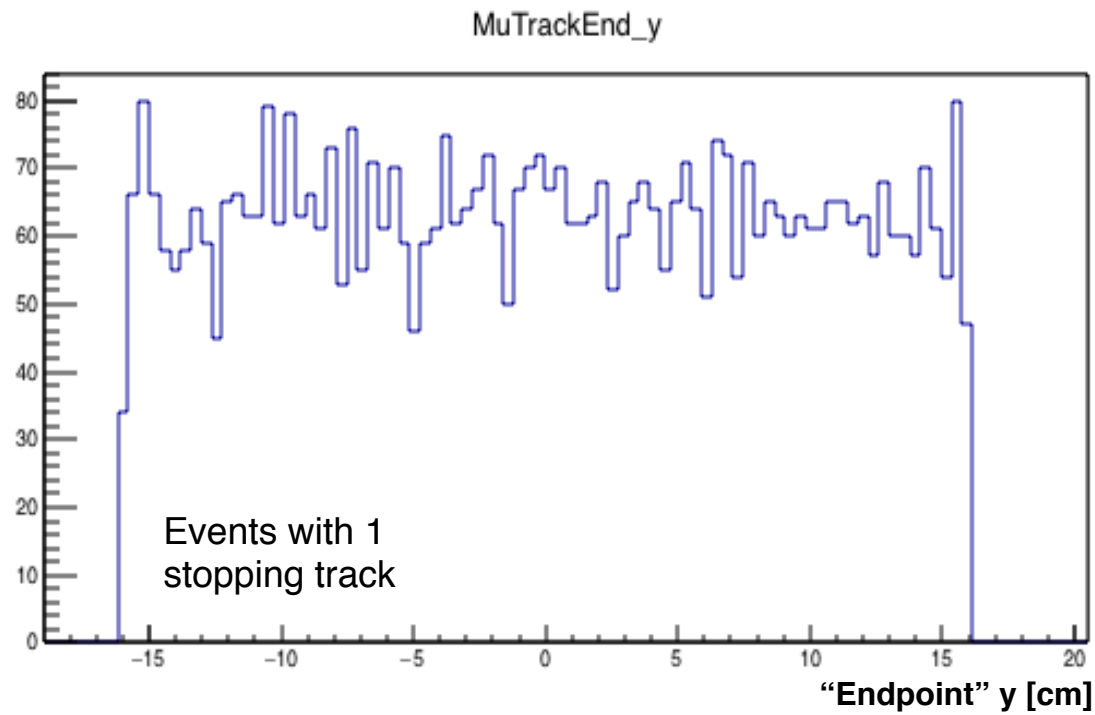




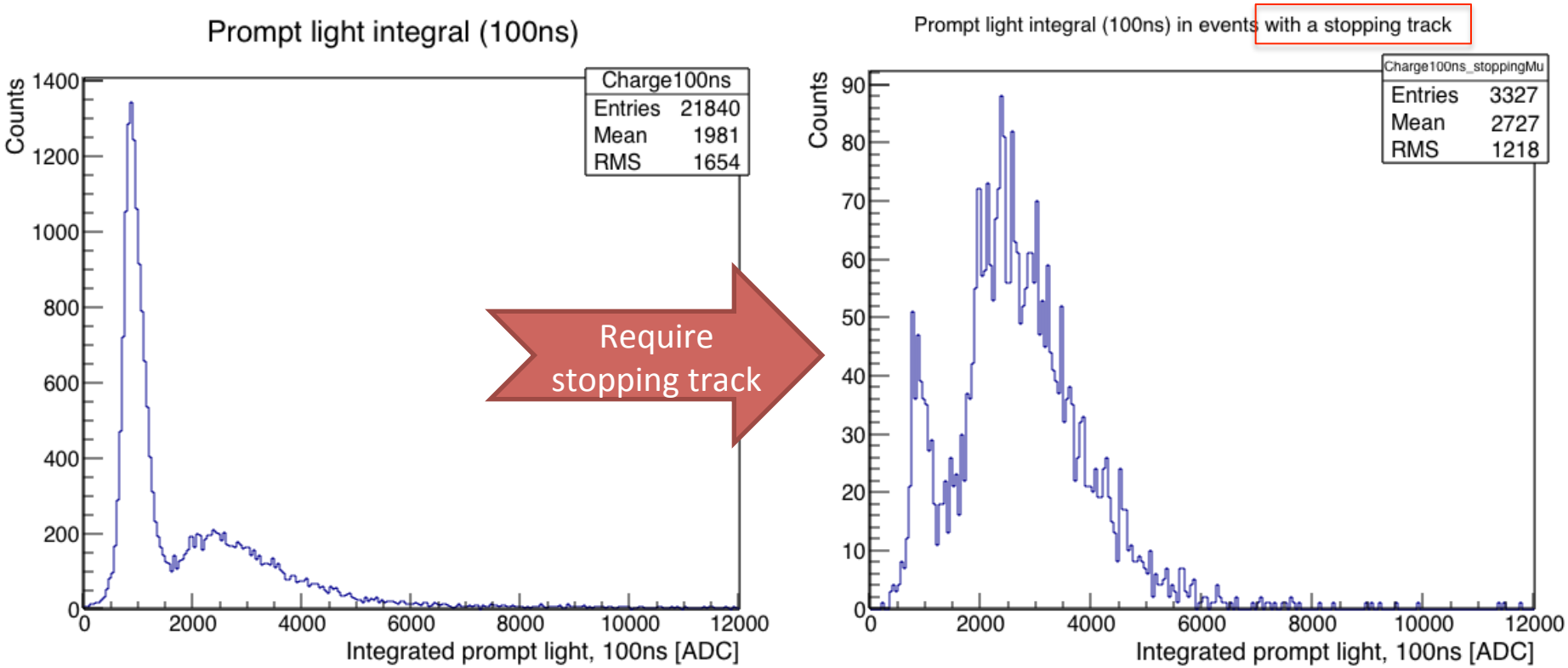


For this study, calling node with larger y the "vertex" (OK assumption for cosmic μ)

IF an event contains exactly <u>one</u> stopping track, assume this is a stopping μ and save its endpoint.

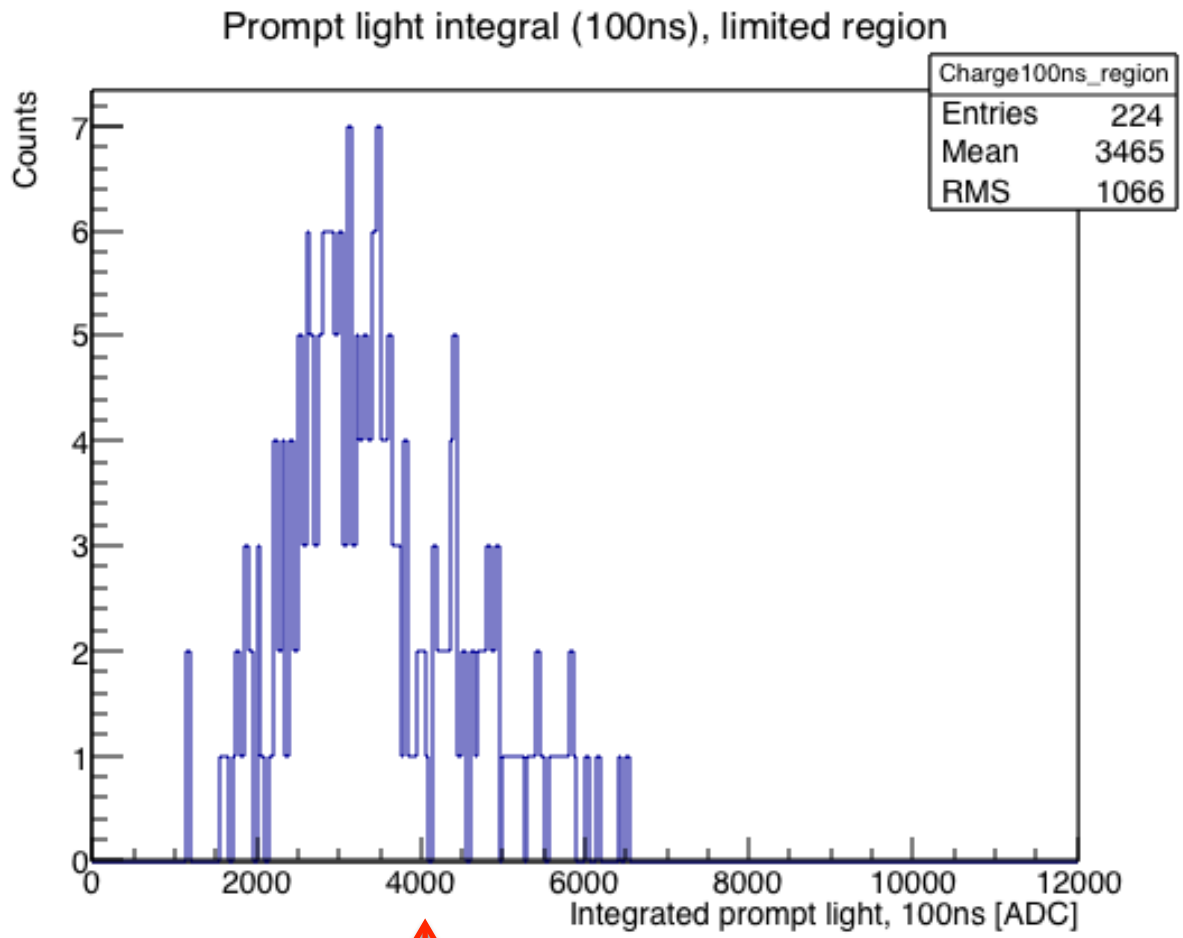


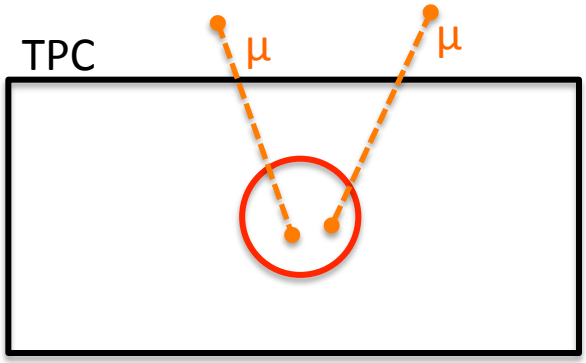
Photon spectrum (requiring 1 stopping track)



Background peak reduced (but still lots of room for improvement)

Selecting only a small region…





Select events where stopping track endpoint within a 20-cm diameter sphere in center of TPC.

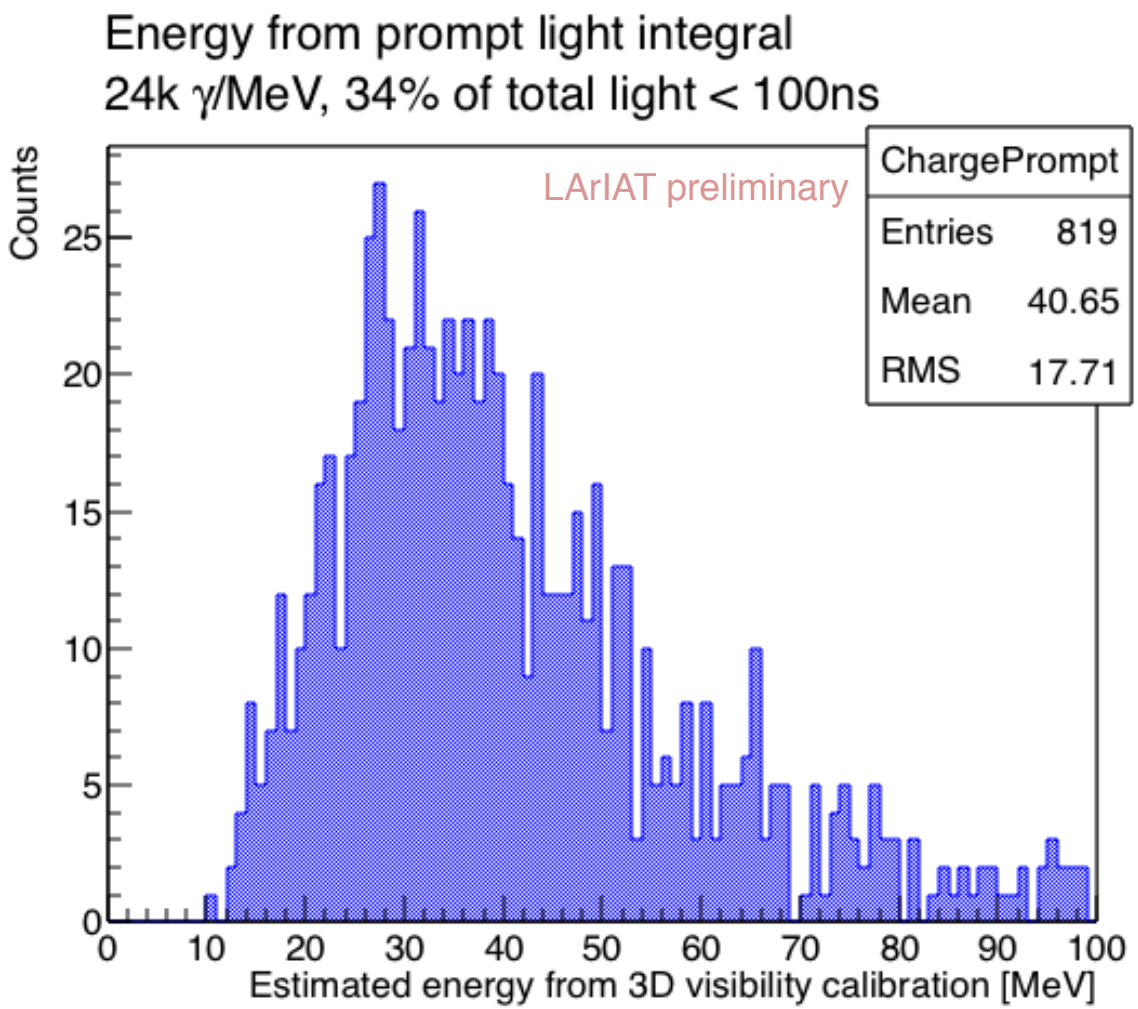
Work in progress!
(Need more statistics…)

LY estimate: 4000 ADC ÷ 51 ADC/pe ÷ 34% total photons within 100ns = 231 pe
231 pe / 53 MeV = **4.4 pe/MeV**

Light-to-energy calibration

Can use "μ" endpoint to get an **energy calibration:**

- Scale each integral using photon visibility at stopping μ endpoint
- Other assumptions:
  - 60ADC = 1 p.e.
  - 24γ/MeV in LAr at E-field
  - 34% of light arrives within 100ns (from integral of average waveform)
  - Michel electron position ~ μ endpoint position

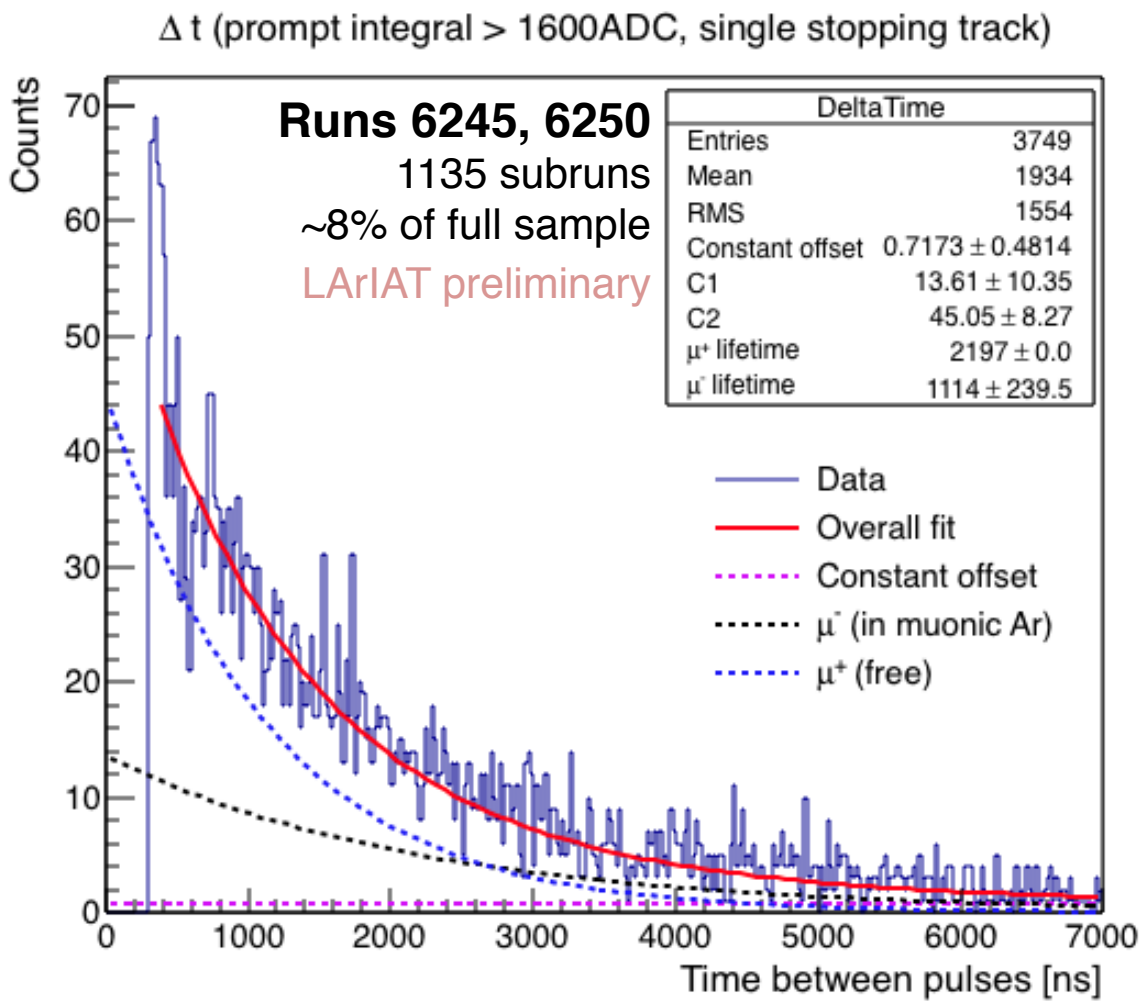


Stopping muon decay spectrum (requiring one stopping track)

$$\frac{1}{\tau_{\mu^-}} = \frac{1}{\tau_{free}} + \frac{1}{\tau_{capture}}$$

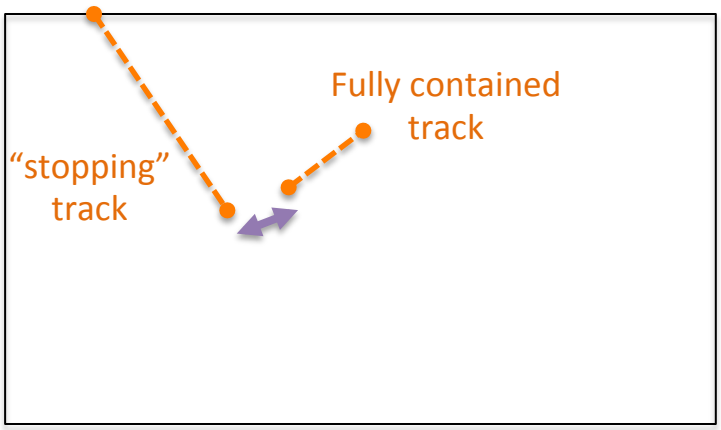
Fit shown here gives
τ_{μ^-} = 1114(240) ns

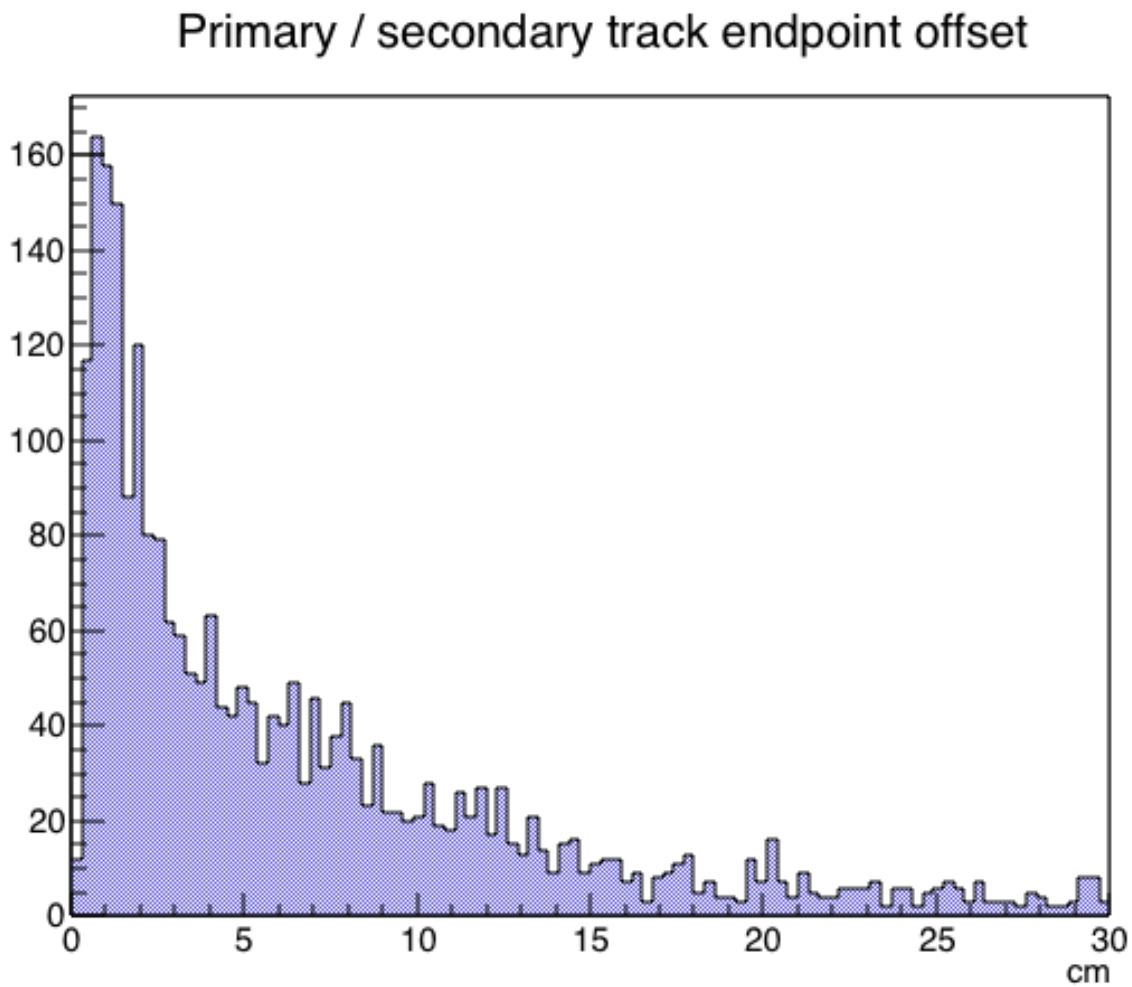
- → $\tau_{capture}$ = 739 ± 106 ns
- Prev. measured values:
  833±55 ns (Bertin et al, 1973)
  709±56 ns (Carboni et al, 1980)



More fun with tracks…

IF an event contains two tracks, and only one is "stopping", how close is the closest node of the second (contained) track?





Need to look at more events to understand how tracker is dealing with Michels…

Next steps

- Mak dedicated optical hit algs for public use in event filtering, reco, etc
- Michel photon spectrum at different regions for course visibility/LY map
- Use light+tracks to find stopping beam muons
  - More complete Michel charge spectrum! (not limited by sensitivity of the light-based trigger sample)
- Calorimetry of Michel cluster/shower
  - Scan region around muon endpoint, add up energy?
  - Combine with light information?
- dE/dx studies (for stopping μ and electrons)

Playing around with optical hits – brief tutorial

1. Source the alg at the top of your module (ie, "MyModule_module.cc") along with all the other standard files, and then create an instance of it inside the class definition.

```
#include "LArIATRecoAlg/OpHitBuilderAlg.h"
...
class MyModule : public art::EDProducer {
  ...
  private:
    OpHitBuilderAlg fOpHitBuilderAlg;
  ...
}
```

2. Append it to your module's constructor function:

```
MyModule::MyModule(fhicl::ParameterSet const & pset)
: fOpHitBuilderAlg(pset)
{
  ...
}
```

Playing around with optical hits – brief tutorial

3. Now to access the OpDetPulses inside your "produce" function of MyModule, you create and fill an art::Handle by looping over the OpDetPulse objects saved in the event, and saving the one with "OpChannel = 1" (the ETL).

```
// Create empty raw::OpDetPulse object to store the ETL waveform in
raw::OpDetPulse ETL_opdetpulse;

// Get the OpDetPulses (WARNING: some events will not
// contain any OpDetPulses...)
art::Handle< std::vector< raw::OpDetPulse >> WaveformHandle;
evt.getByLabel("daq","",WaveformHandle);

// Loop through the available waveforms in this
// event and grab ETL (opchan = 1)
for( int ipulse = 0; ipulse < (int)WaveformHandle->size(); ipulse++){

    art::Ptr< raw::OpDetPulse > ThePulsePtr(WaveformHandle,ipulse);
    raw::OpDetPulse ThePulse = *ThePulsePtr;

    if( ThePulse.OpChannel() == 1) {
        ETL_opdetpulse = ThePulse;
        ETL_waveform = ThePulse.Waveform();
        NSamples = ETL_waveform.size();
        Timestamp = (double(ThePulse.PMTFrame())*8.)/1.0e09;
        PostPercentMark = short(ThePulse.FirstSample());
    }
}
```

Playing around with optical hits – brief tutorial

4. You can now pass this OpDetPulse object "ETL_opdetpulse" directly into a function in OpHitBuilderAlg.cxx which will return "true" if there is exactly 1 hit in the waveform near the trigger time, and false otherwise.

```
bool IsCleanBeamWaveform = fOpHitBuilderAlg.IsCleanBeamWaveform(ETL_opdetpulse);
```

Otherwise, if you want to do more with the waveform (look for 2- or 3-hit events, etc…), you can use more fundamental functions like GetHits. This returns a vector<short> of found hit times in the waveform:

```
// Perform hit-finding/filtering
std::vector<short> hit_times = fOpHitBuilderAlg.GetHits(ETL_waveform);
NumHits = hit_times.size();
std::cout << "Performing optical hitfinding...\n";
std::cout << "We found "<<NumHits<<" hits\n";
for (int i=0; i<NumHits; i++) std::cout<<"   "<<i<<"    t =
"<<hit_times[i]<<"\n";
```

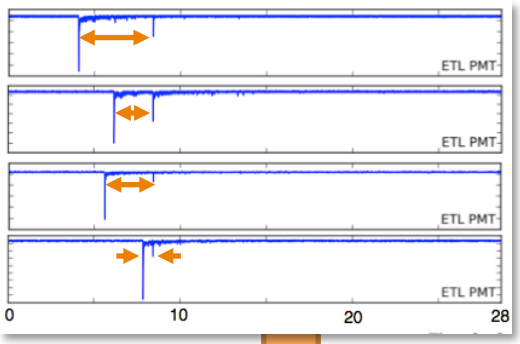
And to get amplitude (in mV) of hit "j":

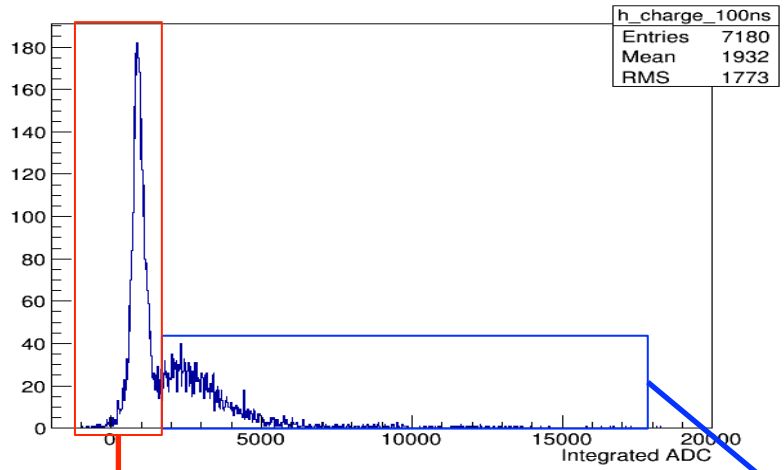
```
double amplitude = fOpHitBuilderAlg.GetHitAmplitude(ETL_waveform,hit_times[j])
```

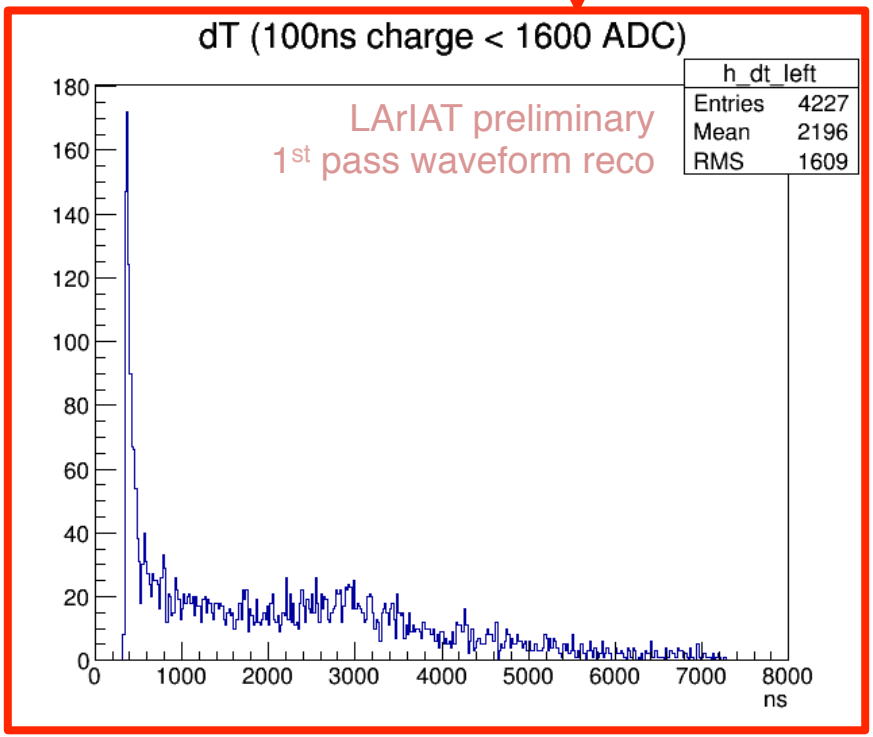
Backup

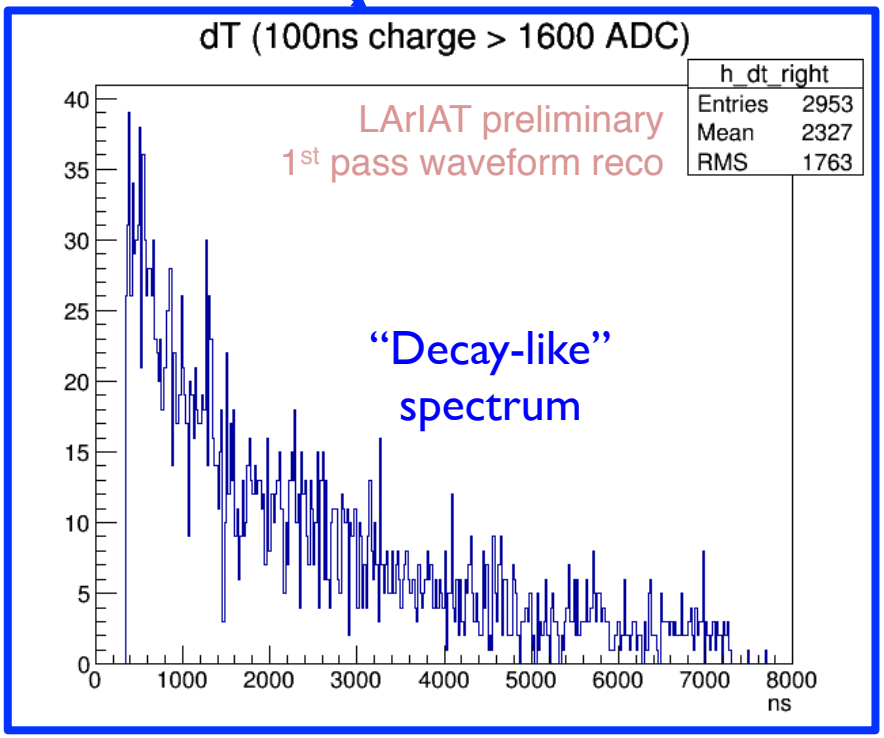
Let's examine the populations'… decay times

Getting decay time spectrum

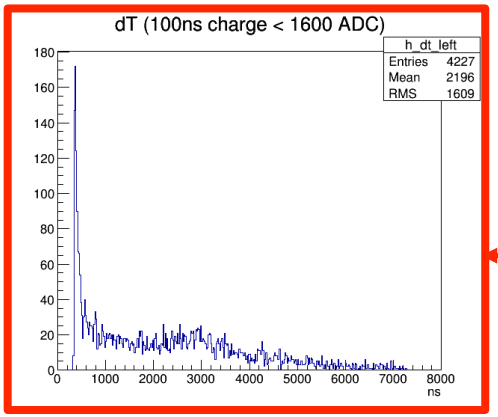


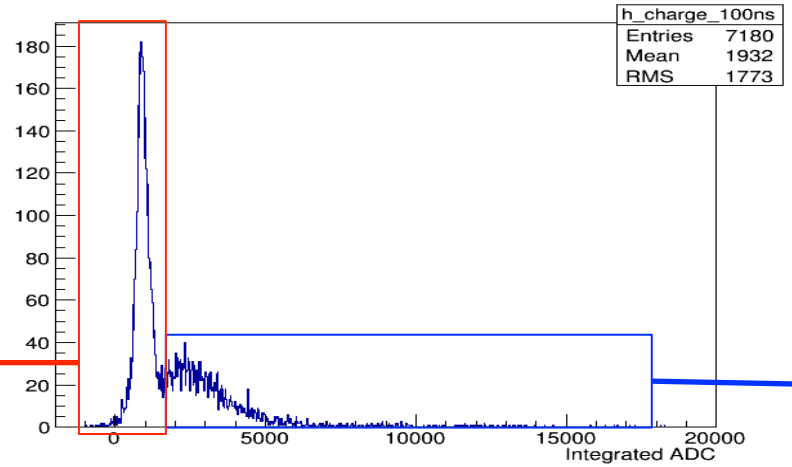


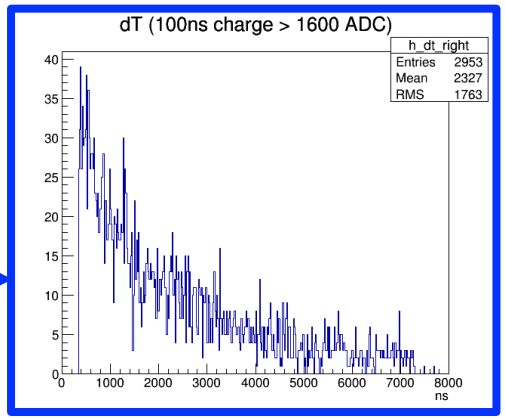




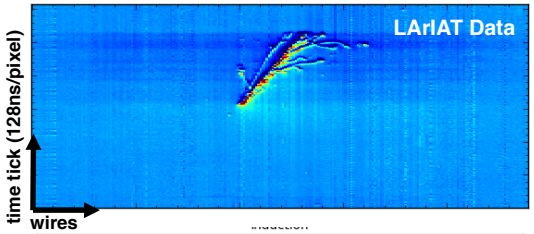
Let's examine the populations'… event displays

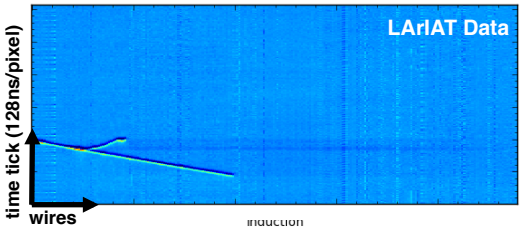


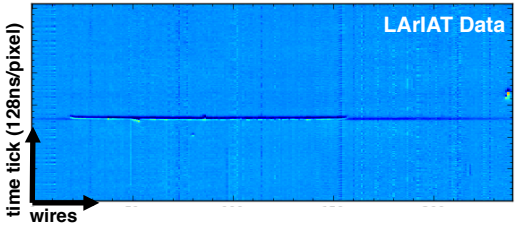




Scanning by eye…

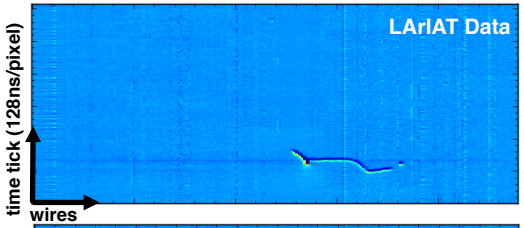


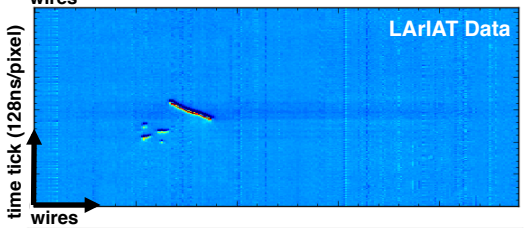


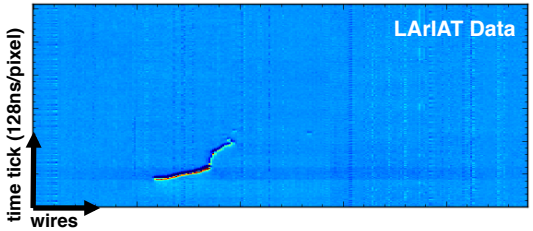


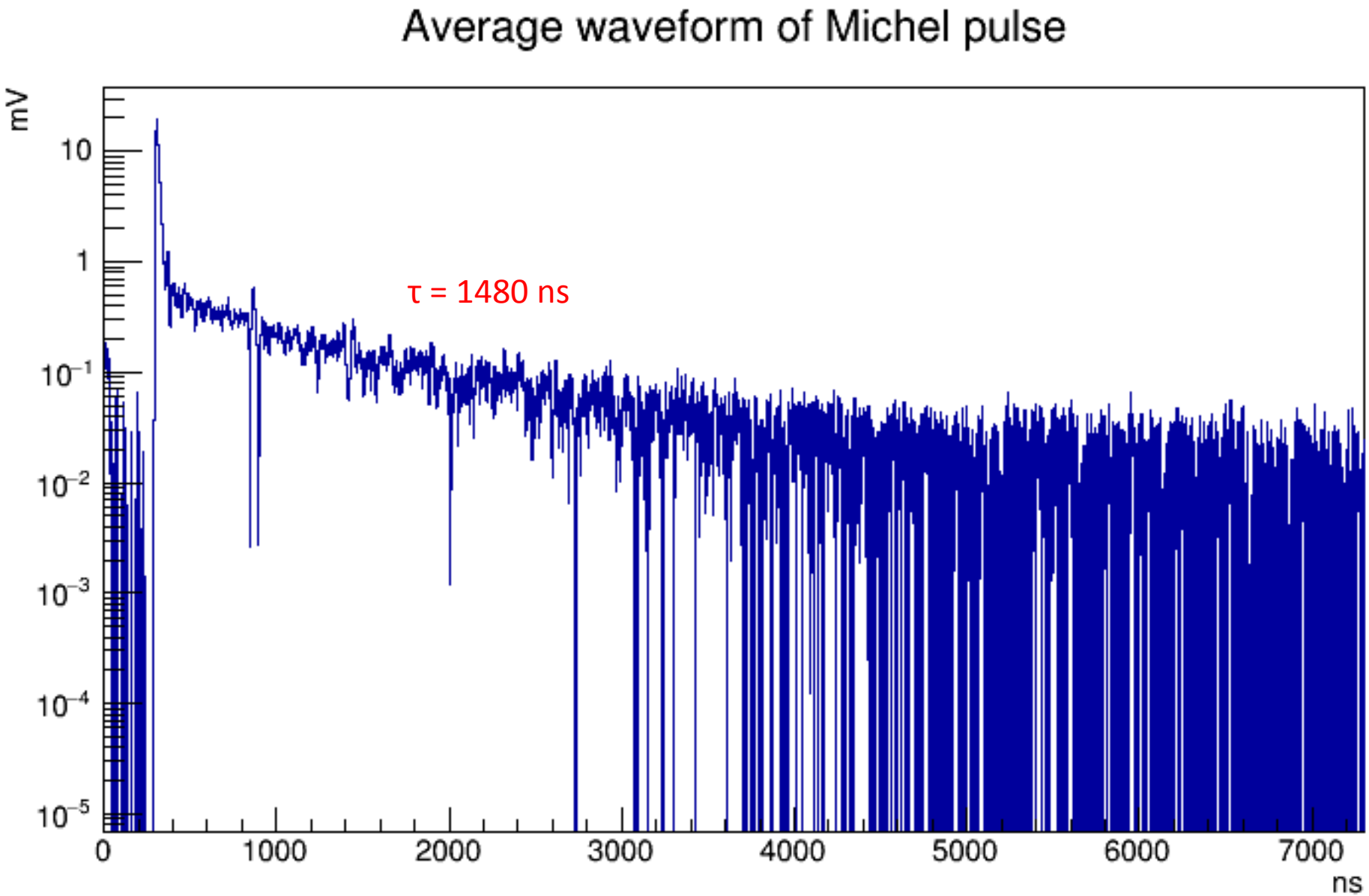
"Clean" sample:
80% Michel-like

BG sample:
13% Michel-like

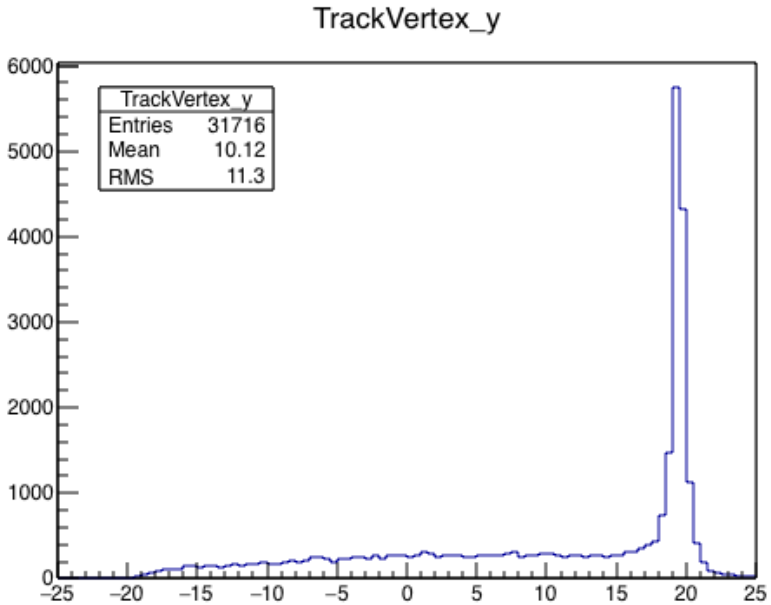


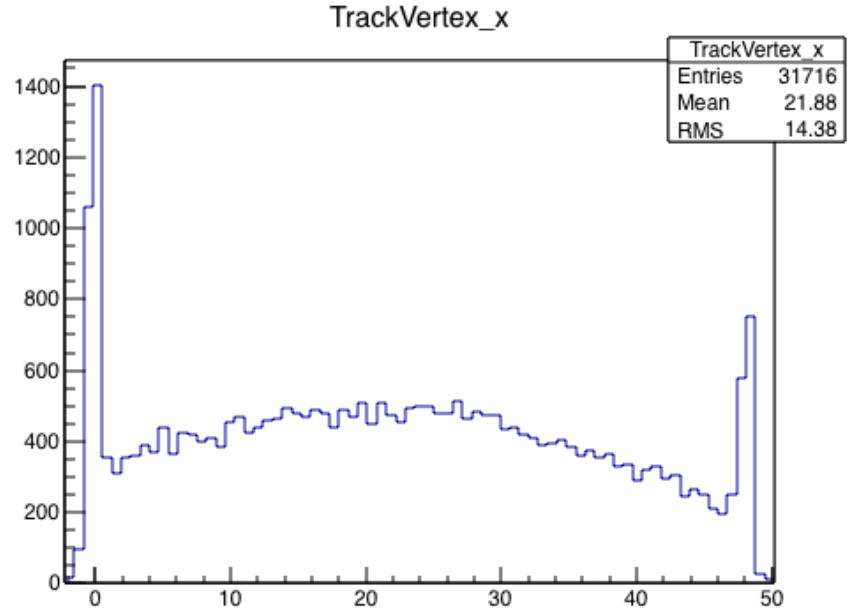


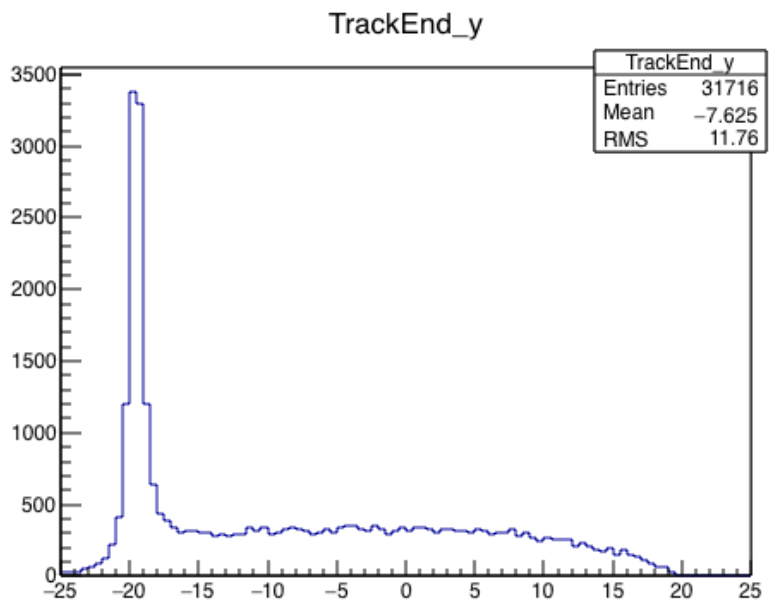


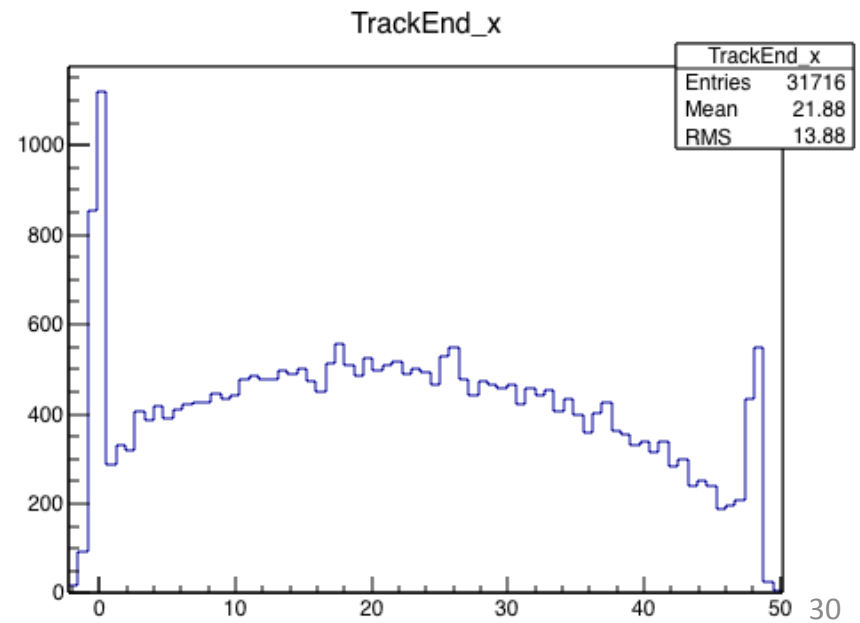
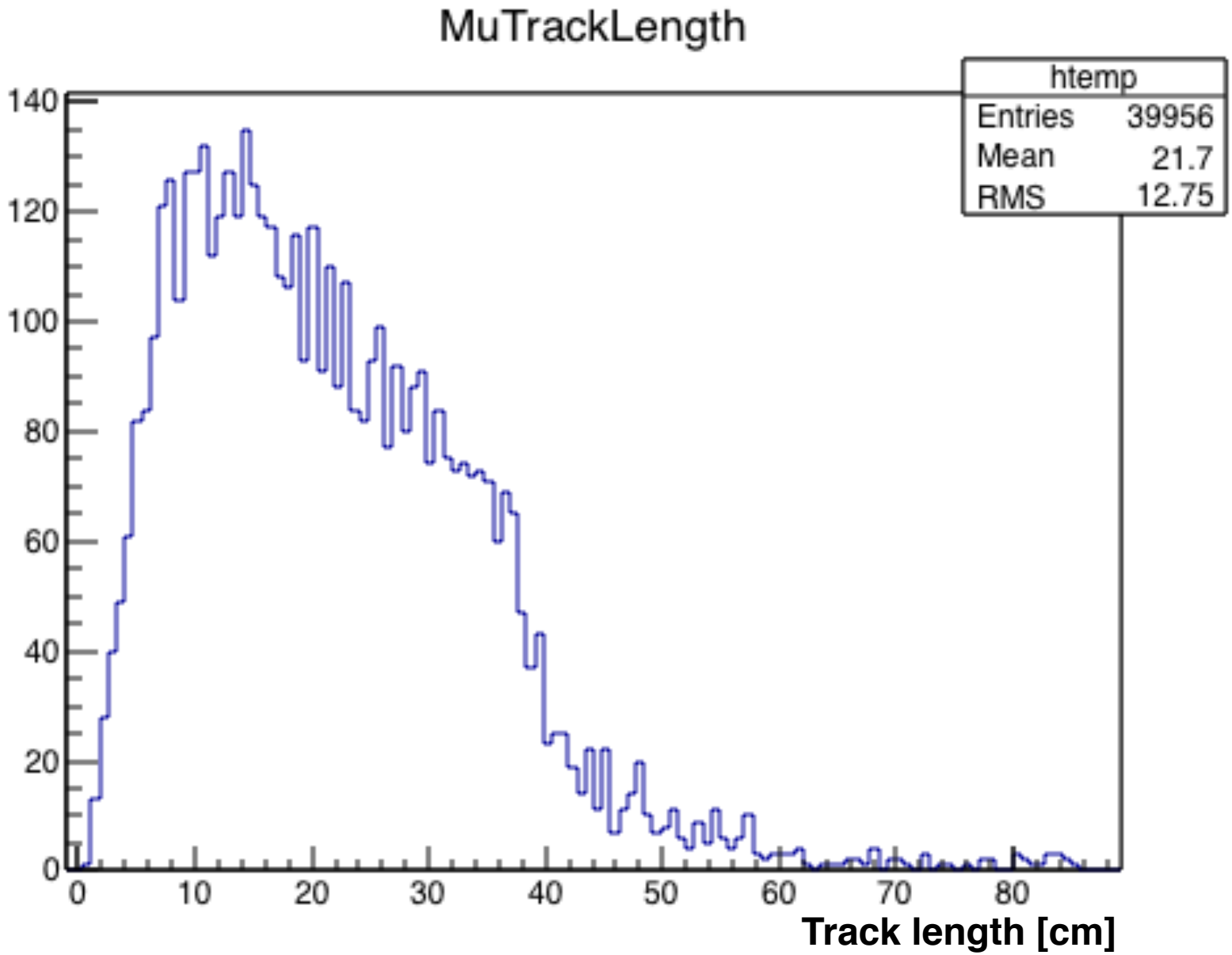

Average waveform of Michel pulse
mV
10
1
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
τ = 1480 ns
0
1000
2000
3000
4000
5000
6000
7000
ns

Track “vertex” and “end” distributions









Track lengths for events with 1 stopper