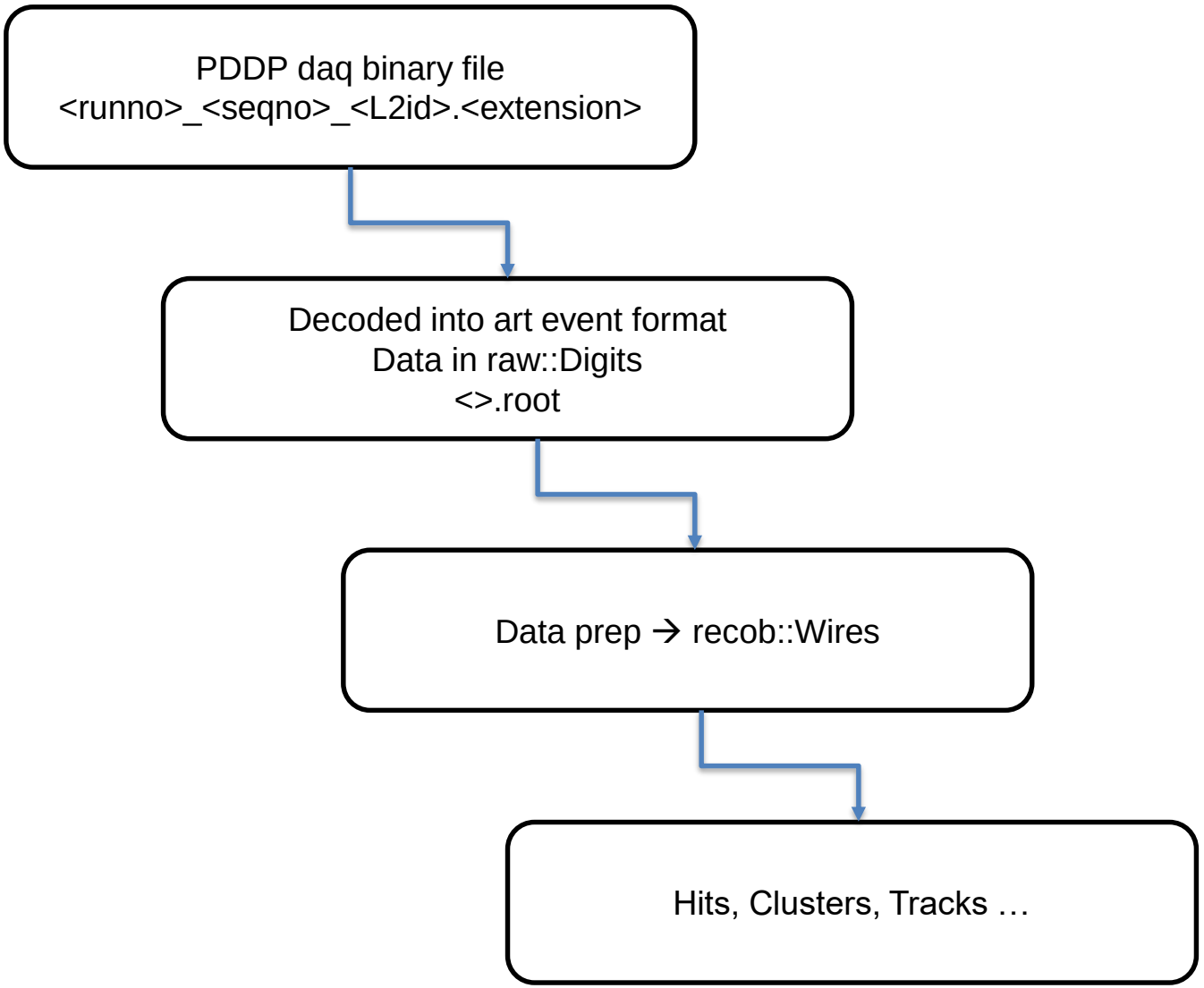
Status of ProtoDUNE-DP data interface to LArsoft reconstruction

Vyacheslav Galymov
IPNL Lyon

ProtoDUNE – DP data and software tutorial
31.10.2019

Steps


PDDP daq binary file
<runno>_<seqno>_<L2id>.<extension>
Decoded into art event format
Data in raw::Digits
<>.root
Data prep → recob::Wires
Hits, Clusters, Tracks …

DAQ raw data decoder

- The decoder to read in DAQ binary data is activated via custom art source input PDDPRawInput
  - E.g.,

```
source:
{
  module_type: PDDPRawInput
  maxEvents: -1
```

- Converter fcl file: dunetpc/dune/Protodune/dualphase/fcl/**pddp_daq_converter.fcl**
- Configuration parameters:
  - **OutputLabelRawDigits** (default if not given product name is **daq**)
  - **OutputLabelRDTime** (default if not given product name is **daq**)
  - **OutputLabelRDStatus** (default if not given product name is **daq**)
  - **InvertBaseline** (default is 0) if > 0, the signals will be inverted. This is a fix for the signal inversion in the early ProtoDUNE-DP data (see later)
  - **SelectCRPs** (default for [] is all) allows to store only data for selected CRPs (the rest are set to 0 samples per channel). This maybe be useful for some dedicated tests

Signal inversion parameter

- First ProtoDUNE DP commissioning data have inverted signal polarity
  - The signals have negative polarity instead of positive (a swap in the analog differential signals that was later fixed)
- For these data need to invert the signal polarity:
  - This is done via **InvertBaseline** option for the decoder
  - Need to specify positive number, should be > largest pedestal: 200 ADC should easily do this
  - The calculation: <Datum> = <InvertBaselineValue> - <Datum>
- Note that the final pedestals in this case would not correspond to the actual ones, but = InvertBaselineValue – ChannelPedestal
- **NOTE: the signal inversion parameter should only be used for runs < 1257 after which the inversion was fixed by using modified cables on CRP1 & 2 (but not SP-like CRP4)**

Channel mapping from DAQ to CRP

Need to remap data from from DAQ storage channels to CRP oriented order ("offline" channel map)
The convention has been established and followed during cabling of CRPs
The mapping is implemented as art service **PDDPChannelMap** in *dune/Protodune/dualphase/RawDecoding/PDDPChannelMap_service.cc*

Can retrieve mapping info from the channel map in different ways:

- By seq number of unpacked channel data
- By crate id / AMC card id / AMC channel no
- By CRP id / view id / view ch no

```
// from seq num in raw data file
boost::optional<ChannelId> find_by_seqn( unsigned seqn ) const;
std::vector<ChannelId> find_by_seqn( unsigned from, unsigned to ) const;

// from crate info
std::vector<ChannelId> find_by_crate( unsigned crate, bool ordered = true ) const;
std::vector<ChannelId> find_by_crate_card( unsigned crate, unsigned card,
                                           bool ordered = true ) const;
boost::optional<ChannelId> find_by_crate_card_chan( unsigned crate,
                                                    unsigned card, unsigned chan ) const;
// from CRP info
std::vector<ChannelId> find_by_crp( unsigned crp, bool ordered = true ) const;
std::vector<ChannelId> find_by_crp_view( unsigned crp, unsigned view, bool ordered = true ) const;
boost::optional<ChannelId> find_by_crp_view_chan( unsigned crp,
                                                  unsigned view, unsigned chan ) const;
```

Channel mapping

```
services:
{
    PDDPChannelMappings:
    {
      service_provider: PDDPChannelMap
      MapName: "pddp2crp"
    }
}
```

Defines mapping for 2CRP PDDP configuration

- Although getting the channel mapping right required a lot of work from many people (both during installation of CRPs and validating afterwards with CRP injection system and cosmics), a normal end user does not need to worry about it
- The decoded data are put in the order which is ready for the reconstruction

Order of raw::Digits in event store

- After the channel data are decoded, the channel map PDDPChannelMap service puts them in the CRP No / View channel order that is needed for reconstruction
- So the channels of raw::Digits written to data store are ordered as follows:
  - CRP 0: view Z ch 0 ... 959, view X ch 0 ... 959
    CRP 1: view Z ch 0 ... 959, view X ch 0 ... 959
    CRP 2: view Z ch 0 ... 959, view X ch 0 ... 959
    CRP 3: view Z ch 0 ... 959, view X ch 0 ... 959

Data products in Event store after decoder

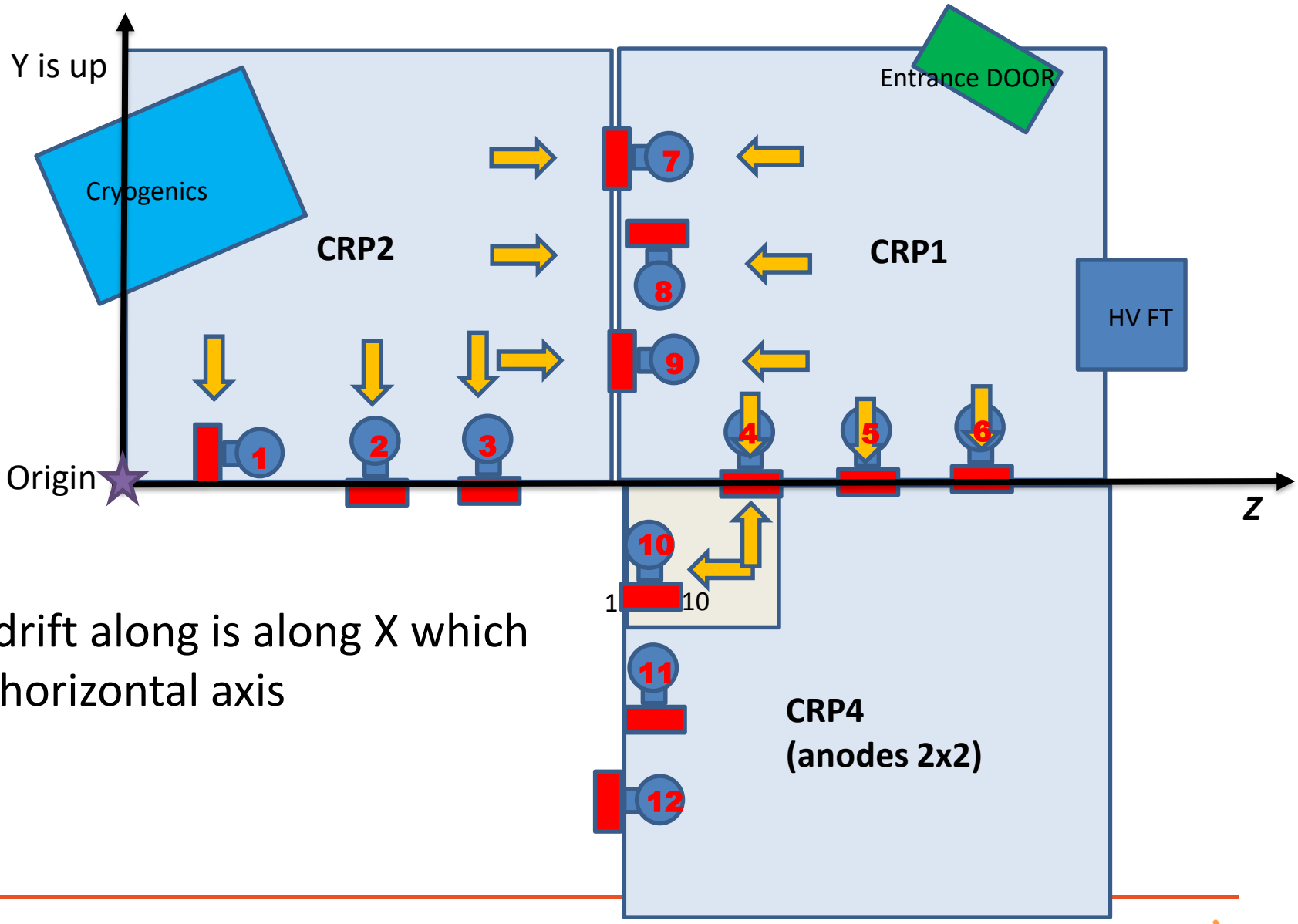
- Run info:
  - art::RunNumber_t ← decoded from the DAQ event header
  - art::SubRunNumber_t ← taken from the seq No in the DAQ file name (the naming convention <runno>_<seqno>_<L2id>.<extension>)

- Event:
  - Event number
  - art::Timestamp event timestamp (timestamp from the WR system)
  - raw::RDStatus status flags
  - raw::RDTimeStamp, same as event timestamp + trigger flags (see dunetpc data prep for more info, but for now these set to CRT trigger bits)
  - raw::RawDigit vector containing ADC data from each channel ordered according to CRP No, View 0, View 1
    - CRP 0, view 0 ch 0 to 959, view 1 ch 0 to 959, CRP1, view 0 ch 0 to 959, …

Detector geometry

- The normal DP detector geometry should have the drift coordinate along the vertical axis (Y-axis)
- However, the 3D reconstruction algorithms are not able to handle this (see for example Protodune DRA meetings of 16/10/19 and 23/10/19)
  - Since X axis, horizontal coordinate, is used by SP detectors, this assumption on the drift appears to be still implicit within the framework
- The rotated (and normal) geometry file has been regenerated (Geometry/gdml/generate_protodunedphase.pl) to follow the correct CRP numbering convention
  - services.Geometry: @local::protodunedphase_geo

Rotated geometry



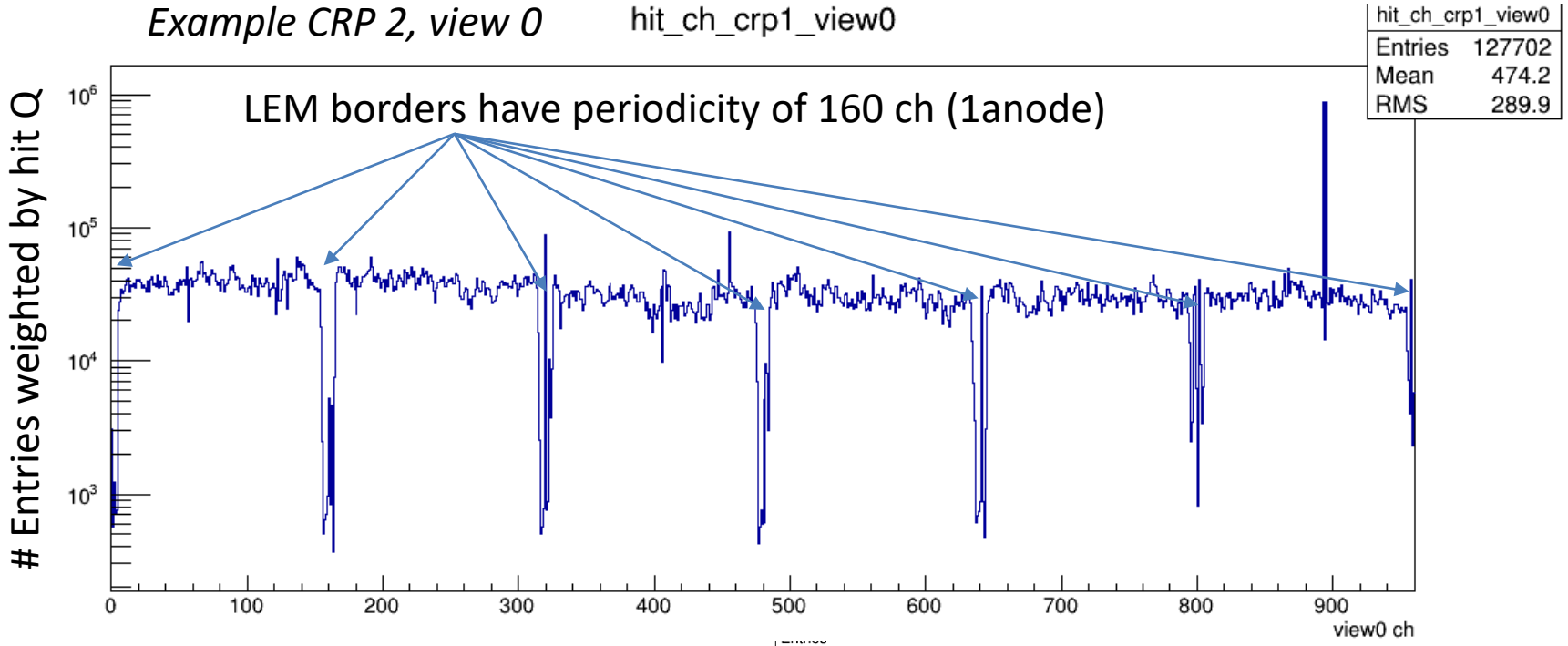
e drift along is along X which is horizontal axis

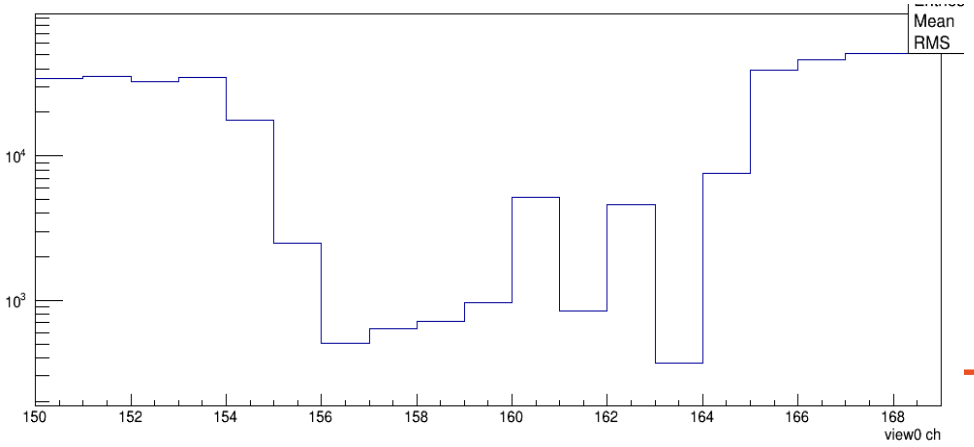
PDDP ChannelStatus service

- The list of bad / noisy channels is specified here:
  - /dunetpc/dune/Protodune/dualphase/fcl/channelstatus_pddp.fcl
  - To be declared in the standard way:
    - services.ChannelStatusService: @local::pddp_channel_status

- Currently the list of bad channels contains:
  - The 3200 channels that are not read in the 2 CRP configuration of ProtoDUNE-DP, but still written by the L1 event builder
  - The list of 280 channels which do not see any signal due to LEMs inactive borders. This is 5 channels around each LEM (so periodicity of 160 channels) → 280 channels in total
- To be added: a few of channels that did show any signals during tests with CRP charge injection system

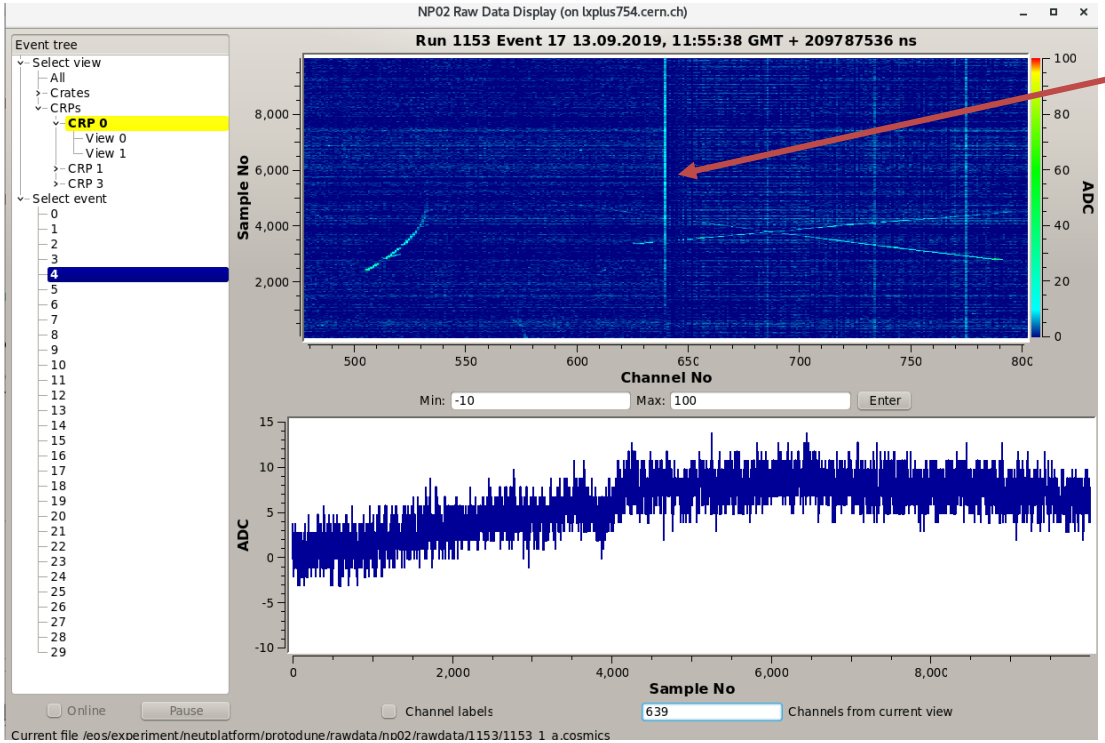
Blind channels due to LEM border



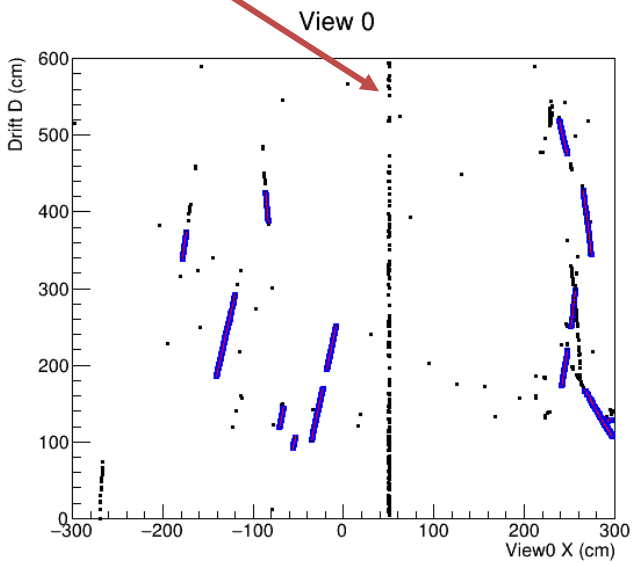


- The border around each LEM masks 5 anode channels
- 5 x 3.125 ~ 16 mm, which is what is expected from LEM design
  - 10 mm FR4 border + 5 mm Cu rim

LEM borders



Slow distortion of baseline generates fake hits



- Have also seen very slow signals in the gaps of 3x1x1 CRP (charge build at borders)
- This can also generate “hot” channels in the middle of dead region

Drift field

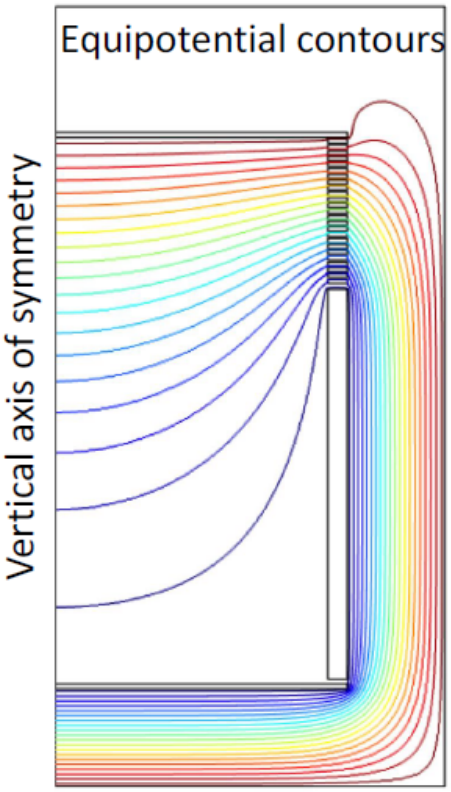
- The drift field in PDDP is non-uniform after about a meter from CRP
  - Problem with VHV feedthrough shorting on one of the field cage profiles via the first degrader link

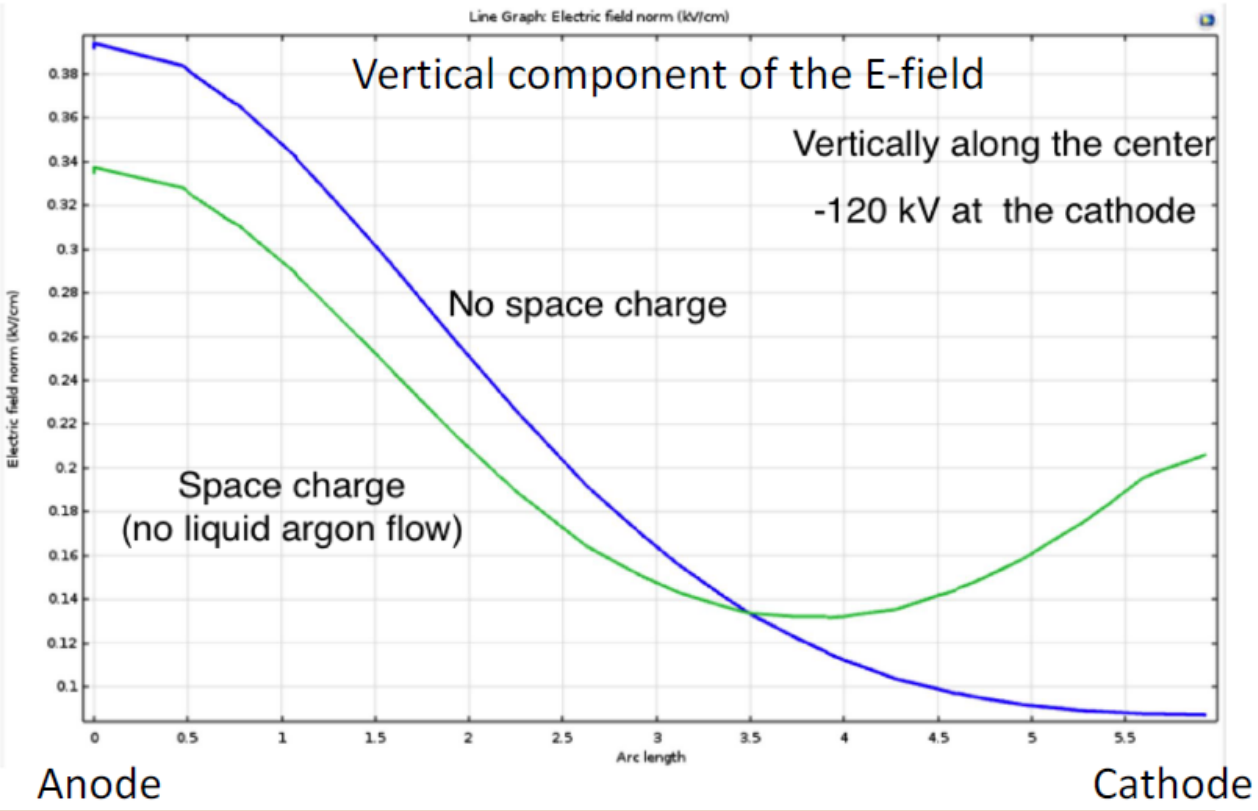
- With the reduced field due to VHV feedthrough issue, the drift field is not 0.5 kV/cm:
  - So far have been running at 50 kV (the drift velocity is ~0.8 mm/us or ½ from what it should have been)
  - Then according to simulations:
    services.DetectorPropertiesService.Efield: [**0.166**, 4.0, 0.0]
    - The other two number should not be touched normally. This is an artifact of the drift field configuration that includes fields from U → V and V → Z in APA planes

Electric field simulation

Simulation of F. Resnati from the report on HV problem

- Given our present understanding, the electric field in the drift volume has been calculated with and without space-charge (only the contribution from primary ionization, no from the ion back flow)





Reco stage

- Example fcl file: dunetpc/dune/Protodune/dualphase/fcl/pddp_reco.fcl
- Tool-based data prep
- Hit finding with DPRawHitFinder developed by C. Alt
  - Fits raw waveform with asymmetric function
- Track reco:
  - trajcluster / pmtrack

Basic DataPrep list of tools for DPDP reconstruction

pddp_datapreptools: [

“digitReader”, ← reads in RawDigits (mandatory)

“pddp_RemoveBadChannels”, ← masks bad channels (optional)

“pddp_adcPedestalFit”, ← runs pedestal fit (mandatory)

“adcSampleFiller”, ← ADC – pedestal (mandatory)

“pddp_adcMultiThreshSignalFinder” ← builds ROI recob::Wire for hit reco using multi threshold ROI search algorithm

]

ROI search tool

```
tools.pddp_adcMultiThreshSignalFinder: {
  tool_type: AdcMultiThreshSignalFinder
  LogLevel: 1
  UseStandardDeviation: true
  Threshold1: 2
  SamplesAboveThreshold1: 10
  Threshold2: 2
  SamplesAboveThreshold2: 10
  ThresholdMax: 3
  ThresholdMin: 0
  BinsBefore: 10
  BinsAfter: 50
}
```

Absolute or in units of PedRMS

Min threshold 2 x PedRMS ~ 3 ADC

Min nb of consecutive samples above

Optionally can set another threshold

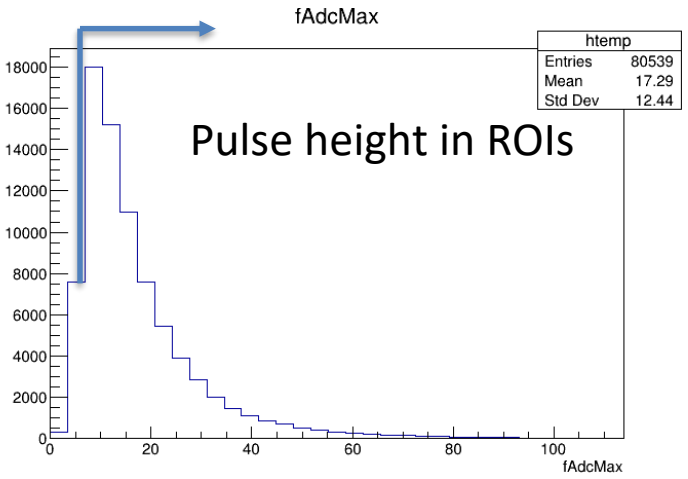
Require max value within ROI region to be above 3 x PedRMS

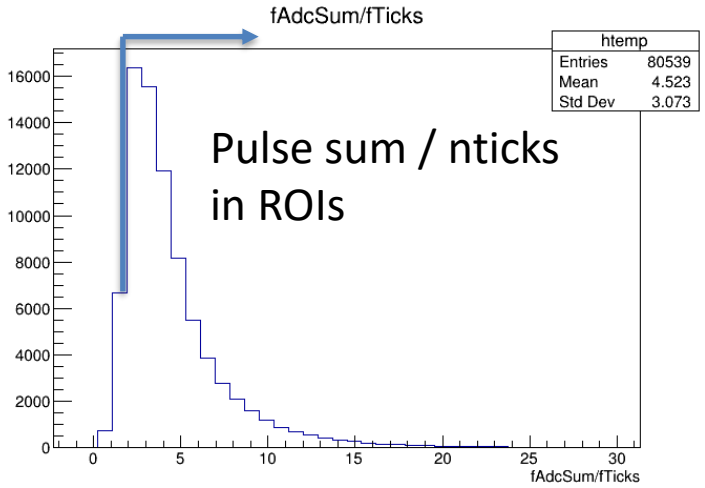
Decides where to stop ROI expansion

Number of ticks to pad on each side

Parameters for hit reco

From run 1265 (dVlem = 3.1 kV)





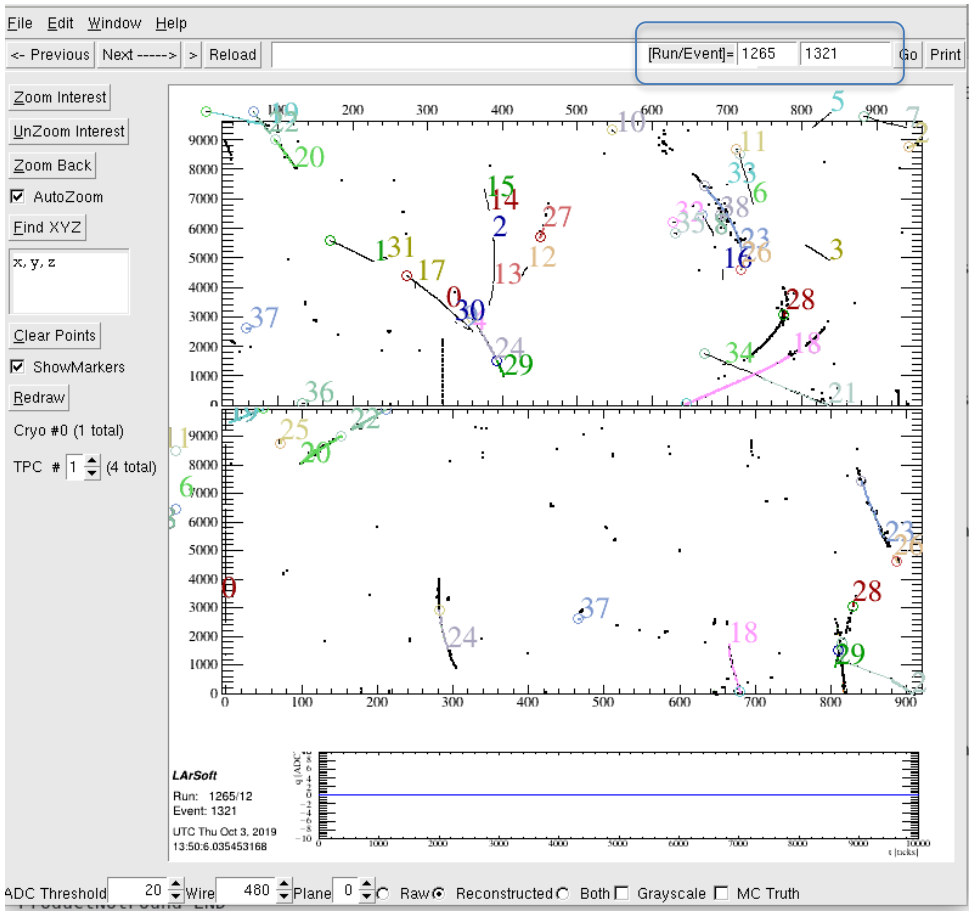
- Hits:
  - Min pulse height 2 ADC
  - Avg pulse sum in 2 ADC / tick
  - Min pulse width 5 ticks

Parameters for track reco

- trajcluster: parameters of standard_trajcluster / standard_trajclusteralg

- pmtrack/pmtrajfit: standard parameters dunefd_pmalgtrackmaker / dunefd_pmalgtrajfitter

Event example



Many trajectories are broken up: missing hits due to LEM dead borders

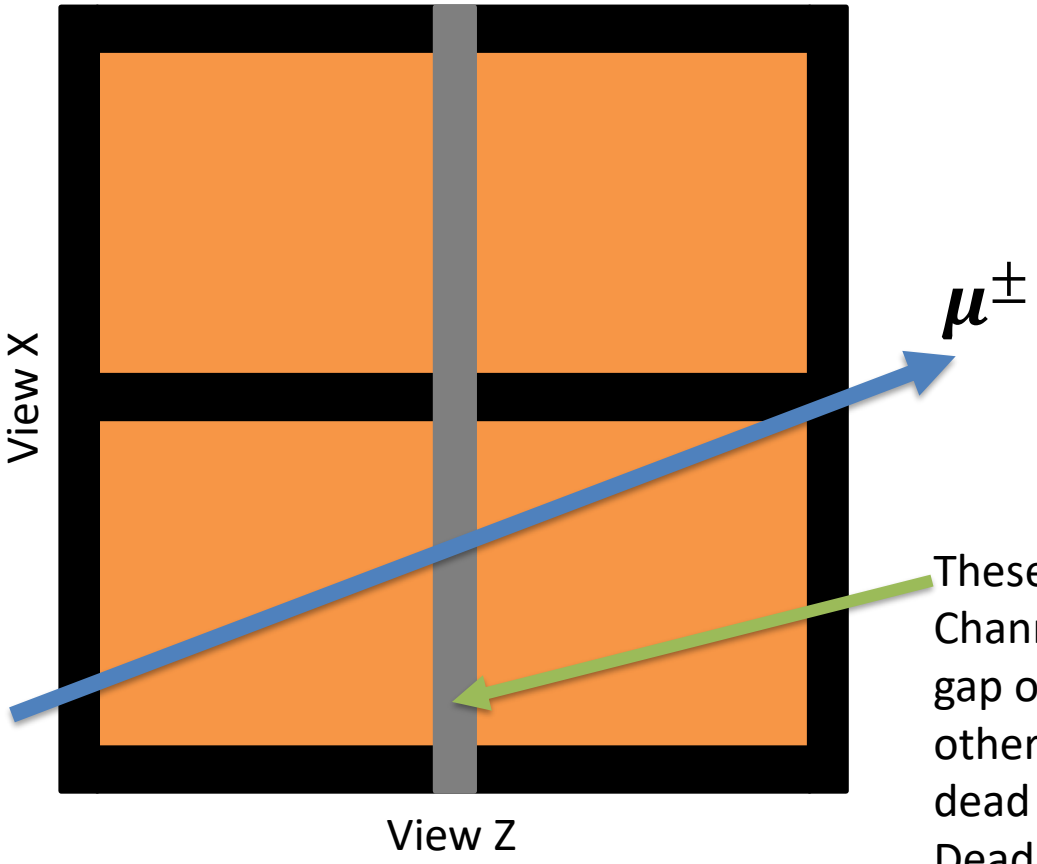
Visually could identify a number of 3D mismatches (a segment of one track in one view is assigned to the wrong track in the other)

Most of the tracks are stubs: τ_e ~ 500 us for this run

➔ the drift volume transparency is ~40 cm and muon tracks fade away making them harder to reconstruct towards the end

However, cosmics that come sufficiently late after trigger should still give a good sample to work with (they are cut by the length of the readout window)

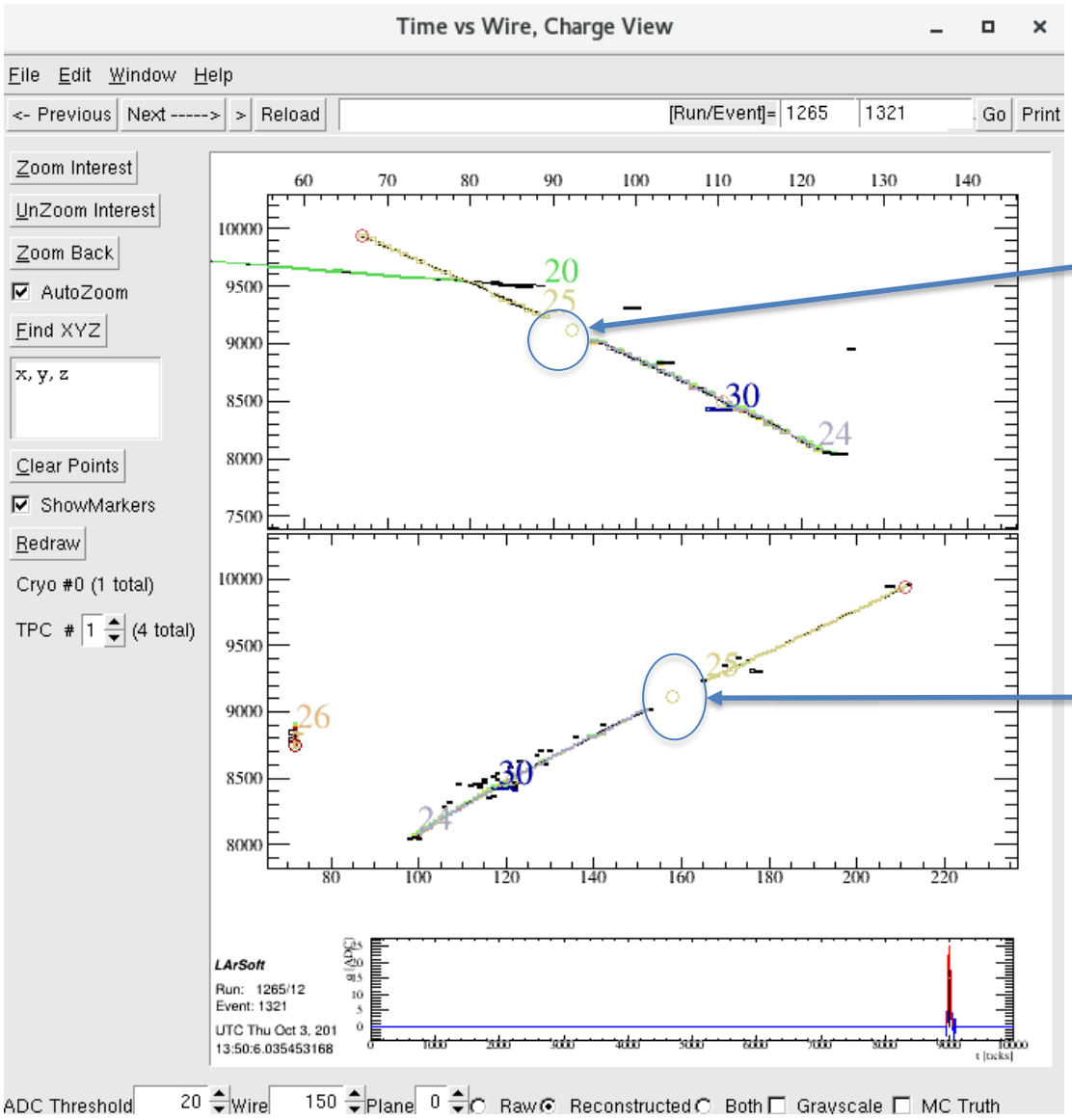
LEM border effects



These channels would be masked by ChannelStatus, but there would also be a gap on otherwise active channels in the other view due to the track crossing the dead space

Dead area is 2D problem and cannot be simply dealt at the level of single electronic channel mapping (1D)

Track breaks at LEM border



In this view, the gap is too large and the track is broken

These channels are masked by channel status service

Detector simulation

Validation reconstruction (tune parameters) requires the detector simulation that incorporates at least major features

- Effect of LEM dead areas
  - track break-up due to gaps / stitching
  - impact on EM shower resolution (reconstruction of ν_e energy spectrum)
- The field map of the current PDDP drift field configuration particularly if one want to study SCE (space-charge effect)

Current DP SimChannel extractor service

Very much the same since Jan 2016 when DP detector was first introduced into larsoft for the detector optimization task force

```
int DPhaseSimChannelExtractService::
extract(const sim::SimChannel* psc, AdcSignalVector& sigs) const {

  // clear and resize temporary ADC buffer
  sigs.clear();
  sigs.resize(m_ntick, 0.0);


  std::vector<double> sigs_original;
  sigs_original.resize(m_ntick, 0.0);

  if ( psc == nullptr ) return 0;

  // get the channel number
  unsigned int chan = psc->Channel();

  //CLHEP::RandGaussQ rGauss(*m_pran, 0.0, fRedENC);

  for ( size_t itck=0; itck<sigs.size(); ++itck )
    {
      sigs[itck] = fDPGainPerView * psc->Charge(itck);
    }

  // perform convolution

  m_psss->Convolute(chan, sigs);

  return 0;
}
```

Creates waveforms on each channel from the simulated charge depositions on the wires

Fixed gain per view right now for all CRPs

➔ Would be good to have a tool to get getCRPViewGain(…) that can be configured in a more flexible way (include blind channel regions as well as variation in LEM-to-LEM gain variations)

Need 2D information of the projected charge

“Imperfect” solution for LEM effects

- The “drift” of charge is done in larsoft SimDriftElectrons__module
  - The charge is assigned to channel/tdc from XYZ of deposit and taking into account LAr purity, diffusion, drift velocity, quenching effects
- The position of the projected charge on the readout planes are not stored
- However, XYZ the energy deposit in the world coordinates is currently available via *SimChannel::TrackIDEs(TDC_t startTDC, TDC_t endTDC)*
- Can do 2D mapping to LEMs
  - But this would ignore the diffusion effects as well as space-charge effects on the drifted charges → not the best solution

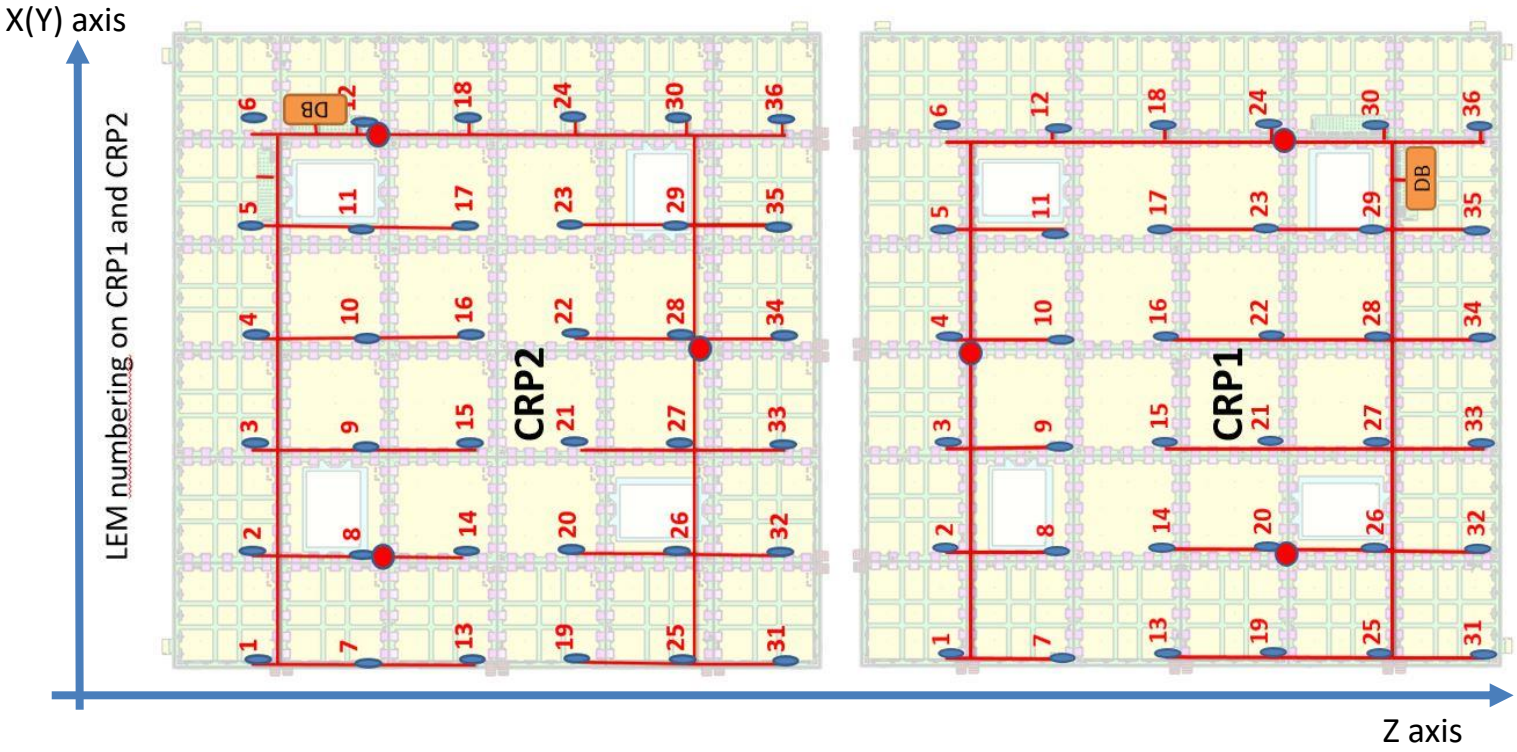
```
struct IDE{

  typedef int TrackID_t;

  IDE();


  IDE(IDE const& ide, int offset);

  IDE(TrackID_t tid,
      float nel,
      float e,
      float xpos,
      float ypos,
      float zpos)
  : trackID     (tid)
  , numElectrons(nel)
  , energy      (e)
  , x           (xpos)
  , y           (ypos)
  , z           (zpos)
  {}


  TrackID_t trackID;
  float numElectrons;
  float energy;
  float x;
  float y;
  float z;
}; // struct IDE
```

- Expand IDE structure in sim::simChannel to include a minimum doublet float[2] of projected position of the cluster on the readout plane
- Add transverse part of the projected position, fDriftClusterPos, in SimDriftElectrons_module
- This would take care of any diffusion (and space-charge) effects when mapping to CRP LEMs

➔Requires modifying core larsoft module/objects

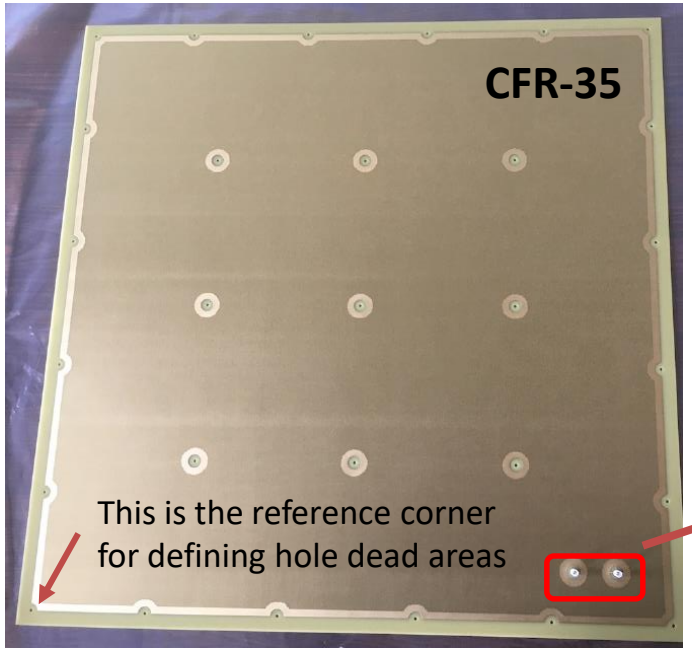
LEM numbering convention in sim

Convention to for LEM numbering that will be followed, when the LEM gain is specified for each unit: CRP# LEM# <gain value>

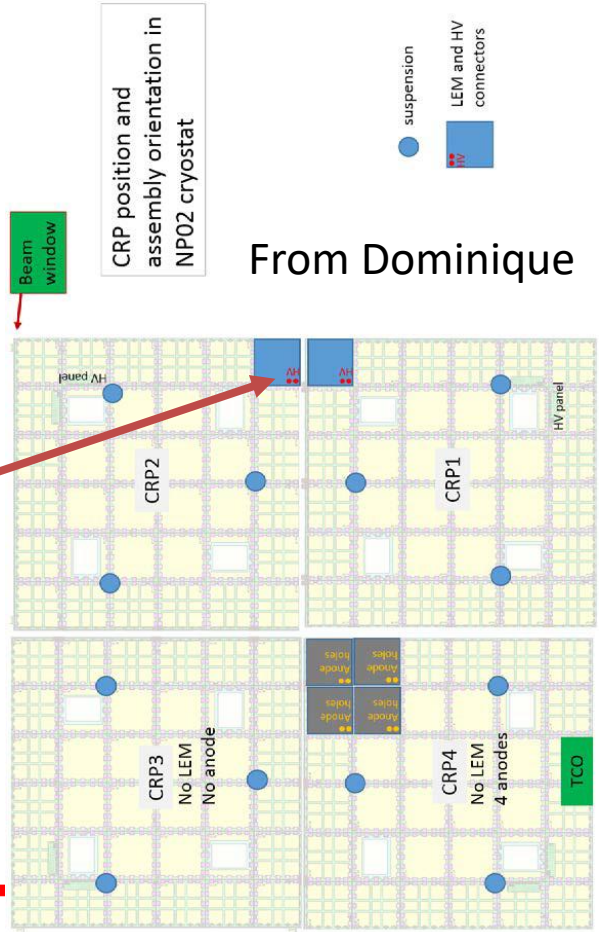


Need to take care of the actual position of the HV connections in these coordinates to correctly specify the dead area due to these utility holes

LEM dead areas



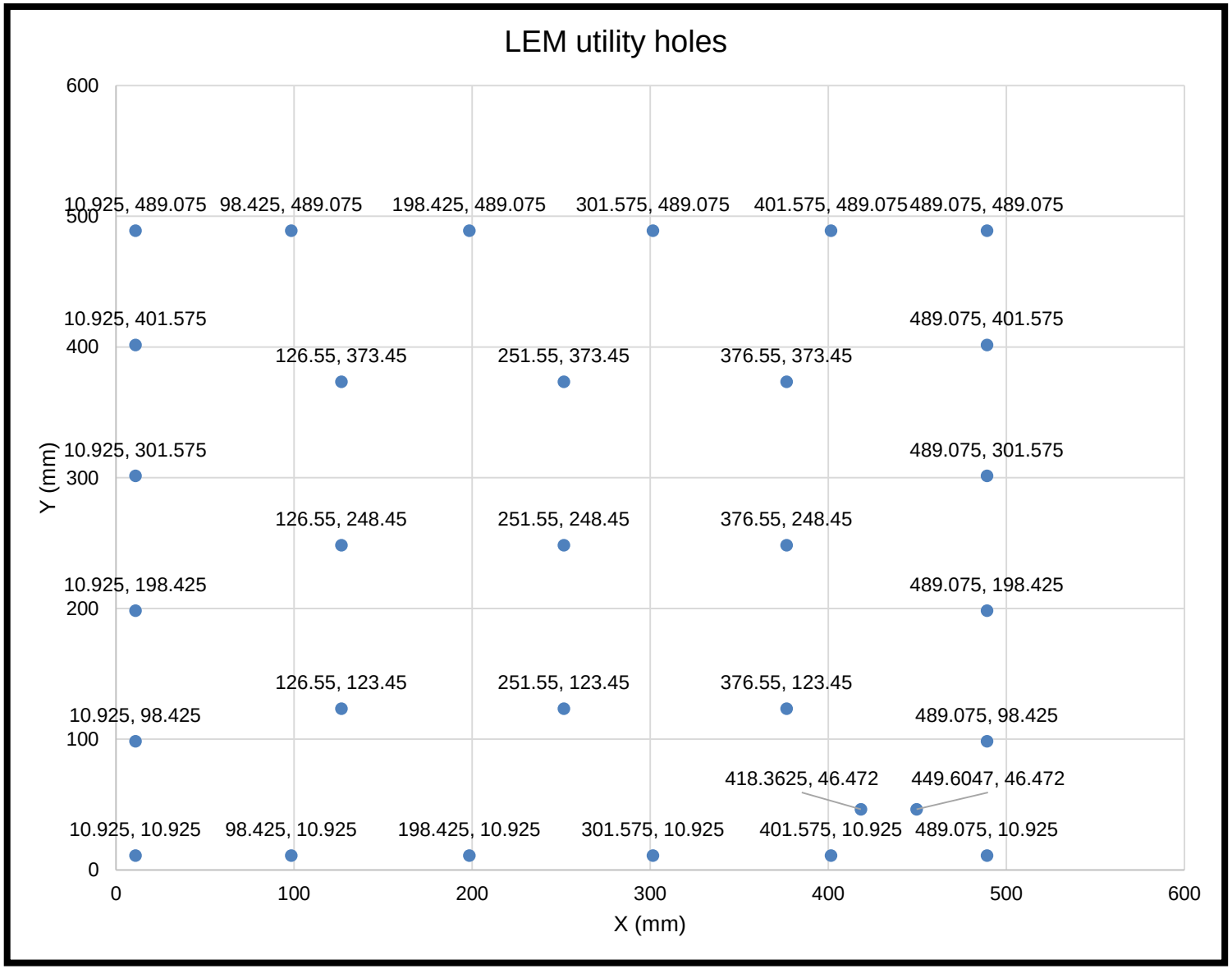
Need to know to mask the correct sections of the anode



		LEM borders		Screw holes			
LEM design	% Active area	FR4	copper guard ring	FR4 ring Φ	copper guard ring Φ	FR4 ring Φ	copper guard ring Φ
CFR-34	96.2	2 mm	2 mm	4.2 mm	6 mm	10 mm	12 mm
CFR-35	85.8	10 mm	5 mm	10 mm	20 mm	10 mm	20 mm
CFR-36	92.1	2 mm	5 mm	10 mm	20 mm	10 mm	20 mm

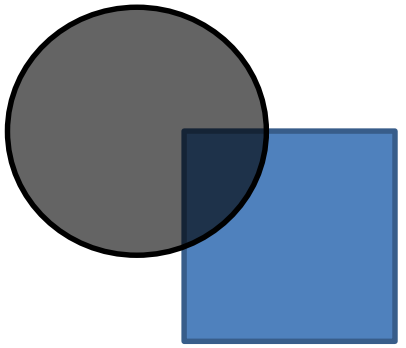
LEM utility hole positions extracted from GERBER

These positions will determine the location of dead areas due to utility holes when mapping signals to anode channels



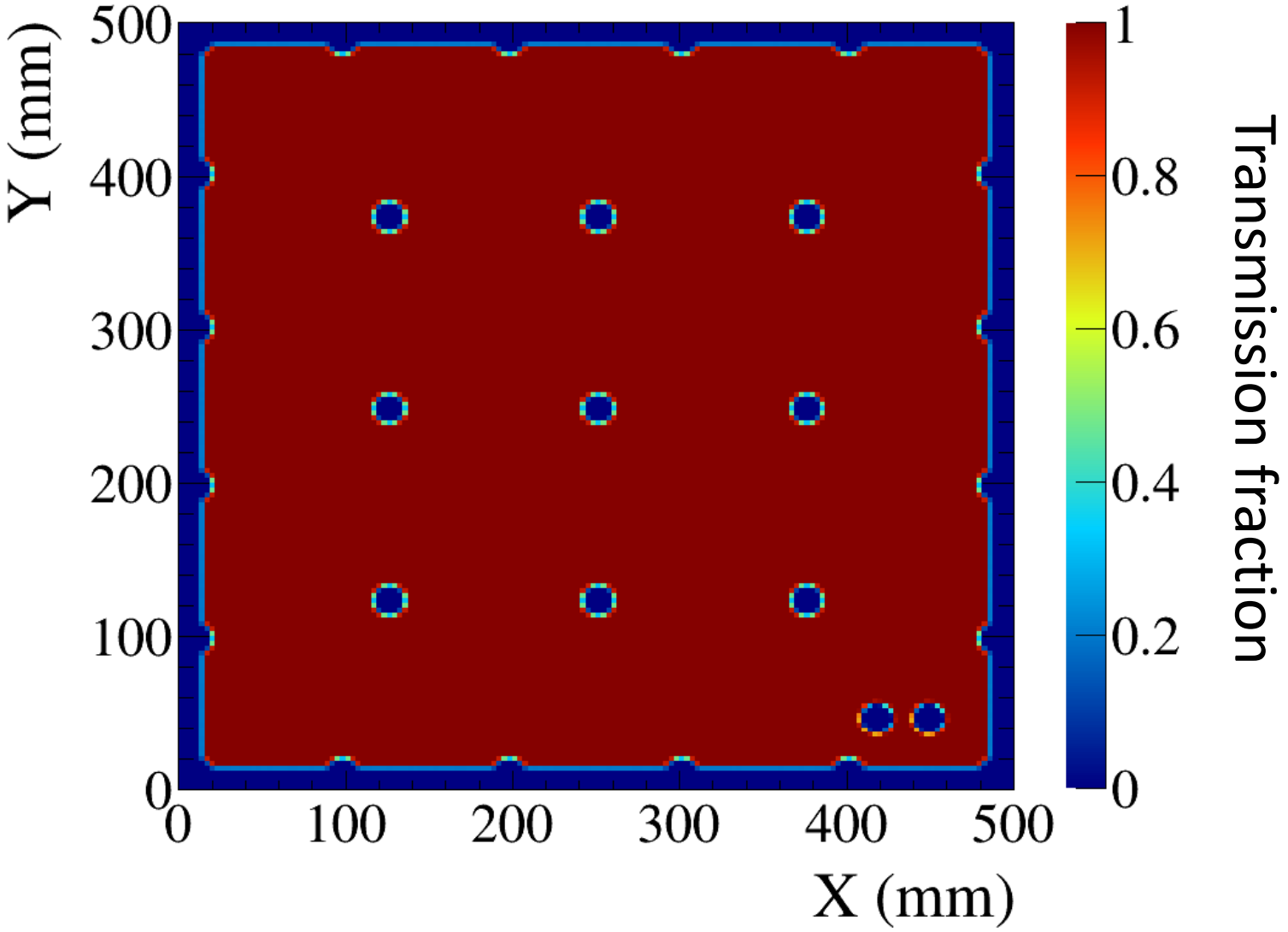
LEM-Anode transmission map

- The transmission coefficient for each channel is calculated simply as a fraction of an area overlap between LEM dead region and a square pixel of 3.125 mm x 3.125 mm



- The area overlap is calculated using Monte Carlo (fall-in hits/total throws)
  - Calculating overlaps analytically between circles and squares quickly becomes rather cumbersome

Calculated transmission map



Conclusions

- The decoder for ProtoDUNE-DP data and the channel map service to go from online to offline channel numbering have been available since summer
- The data prep stage runs smoothly on the unpacked data (D. Adams from BNL fixed the issue #23283 with errors in pedestal fits)
- There is still an issue with vertical drift geometry, so need to use fake horizontal drift geometry
- The track reconstruction requires tuning to deal with LEM dead area effects
  - Even at the level of a single events see a number of mismatches between views
  - Needs to be fixed if one wants to get dQ/dx for example to look at LEM gain
- The tracking performance should be checked and tuned on MC
  - It looks to be feasible to include LEM dead areas in the simulation
  - The transmission map given the actual LEM geometry have been prepared
  - Hopefully the implementation in dunetpc should be finalized soon

Protoduneana

- There is an on-going work (managed by T. Junk) to split any relevant protodune analysis code (protoduneana) from dunetpc
- Motivation: dunetpc is slow to recompile
  - frequent code changes when developing analyses → frequent recompilation cycles → a lot of time is waiting for code to compile